PROBABILISTIC REASONING AND LEARNING ON PERMUTATIONS:
EXPLOITING STRUCTURAL DECOMPOSITIONS OF THE SYMMETRIC GROUP

JONATHAN HUANG

CMU-RI-TR-11-28

Submitted in partial fulfillment of the requirements for the degree of Doctor of Philosophy in Robotics

The Robotics Institute
Carnegie Mellon University
Pittsburgh, Pennsylvania 15213

July 2011

Thesis committee
Carlos Guestrin, CMU (chair)
John Lafferty, CMU,
Drew Bagnell, CMU,
Alex Smola, Yahoo! Research,
Leonidas Guibas, Stanford

ABSTRACT

Probabilistic reasoning and learning with permutation data arises as a fundamental problem in myriad applications such as modeling preference rankings over objects (such as webpages), tracking multiple moving objects, reconstructing the temporal ordering of events from multiple imperfect accounts, and more. Since the number of permutations scales factorially with the number of objects being ranked or tracked, however, it is not feasible to represent and reason with arbitrary probability distributions on permutations. Consequently, many approaches to probabilistic reasoning problems on the group of permutations have been either ad-hoc, unscalable, and/or relied on rigid and unrealistic assumptions. For example, common factorized probability distribution representations, such as graphical models, are inefficient due to the mutual exclusivity constraints that are typically associated with permutations.

This thesis addresses problems of scalability for probabilistic reasoning with permutations by exploiting a number of methods for decomposing complex distributions over permutations into simpler, smaller component parts. In particular, we explore two general and complementary approaches for decomposing distributions over permutations: (1) *additive decompositions* and (2) *multiplicative decompositions*. Our additive approach is based on the idea of projecting a distribution onto a group theoretic generalization of the Fourier basis. Our multiplicative approach assumes a factored form for the underlying probability distribution based on a generalization of the notion of probabilistic independence which we call *riffled independence*.

We show that both probabilistic decompositions lead to compact representations for distributions over permutations and that one can formulate efficient probabilistic inference algorithms by taking advantage of the combinatorial structure of each representation. An underlying theme throughout is the idea that both kinds of structural decompositions can be employed in tandem to relax the apparent intractability of probabilistic reasoning over the space of permutations.

From the theoretical side, we address a number of problems in understanding the consequences of our approximations. For example, we present results in this thesis which illuminate the nature of error propagation in the Fourier domain and propose methods for mitigating their effects.

Finally, we apply our decompositions to multiple application domains. For example, we show how the methods in the thesis can be used to solve challenging camera tracking scenarios as well as to reveal latent voting patterns and structure in Irish political elections and food preference surveys.

To summarize, the main contributions of this thesis can be categorized into the following three broad categories:

- Principled and compactly representable approximations of probability distributions over the group of permutations,
- Flexible probabilistic reasoning and learning algorithms which exploit the structure of these compact representations for running time efficiency, and
- Theoretical analyses of approximation quality as well as algorithmic and sample complexity.

ACKNOWLEDGMENTS

As I write the final paragraphs of my thesis at a sunny coffeeshop terrace in Barcelona, I realize that I have enjoyed a fantastic life over the last few years of graduate school. I have many people to thank for this, both from Pittsburgh and far away! First I must thank my thesis advisor, Carlos Guestrin, for his constant support, encouragement, and probing questions. I will always be grateful to him for taking me from being a student with no research publications to being a confident researcher. And hopefully I will one day be able to tell my own students that their papers and presentations need more Oomph! *Muito obrigado!*

I am also particularly grateful to Leo Guibas — for initially proposing some of the ideas upon which this thesis is based, for collaborating on a number of projects throughout the last several years, for his stories over dinner at various conferences, and finally for giving me my next job. I have always admired Leo's applications of beautiful mathematical ideas to solve a number of different applications, and I know that I have much more to learn from working with him.

I want to thank the remaining members of my thesis committee: Alex Smola, Drew Bagnell, and John Lafferty, each for taking the time to discuss my research progress, to read my papers, and provide feedback.

Michelle Martin, Suzanne Lyons Muth and Diane Stidle were crucial for making my graduate experience smooth by making sure everything happened the way they were supposed to happen.

Additionally, there have been a number of researchers in the community with whom I have enjoyed discussions and collaborations. Xiaoye Jiang has been a close collaborator, even coming to Pittsburgh for a semester. I am grateful to Kyle Heath for helping to collect and process data the week before my first ever NIPS deadline! I have learned so much from Risi Kondor, from our discussions, from reading his beautifully written papers, and his unwavering campaign to bring group theoretical methods into the machine learning community. I am very grateful for my discussions with Marina Meilă, which led to the development of one of the primary topics of this thesis (riffled independence).

I was fortunate to enjoy several summer internships during graduate school. My first one was at Intel Research, thanks to Ali Rahimi, who gave me my first (and hopefully last) chance to use a real oscilloscope and soldering iron. I am also grateful to Ashish Kapoor for mentoring me during a second Seattle internship at Microsoft Research, and for taking me out in his airplane to see Seattle from a whole new point of view. Many thanks also to Eric Horvitz for setting up the wonderful opportunity for interning at MSR and for patiently listening to me talk about Fourier analyis. Going even further back in time, I am grateful to Li Deng for a fascinating summer internship at IPAM-UCLA. And still further, I thank

Rob Rosenberg from the Naval Research Labs for mentoring me in my first ever research experience and getting me interested in computing research. Because of Rob, I will probably always have a fond interest in computer graphics.

I have learned so much from being a member of the SELECT lab run by Carlos Guestrin and Geoff Gordon. Learning seems to happen via osmosis whenever I am in the same room as Geoff, who seems to know everything. I am also thankful to labmates (past and present): Danny Bickson, Byron Boots, Joseph Bradley, Anton Chechetka, Kit Chen, Miroslav Dudík, Khalid El-Arini, Stanislav Funiak, Joseph Gonzalez, Arthur Gretton, Sue Ann Hong, Andreas Krause, Aapo Kyrola, Yucheng Low, Dafna Shahaf, Sajid Siddiqi, Ajit Singh, Le Song, Gaurav Veda, Yisong Yue, and Brian Ziebart. There were a number of undergraduates that I enjoyed meeting with over the years (even if I'm not sure if the enjoyment was mutual). They were: Hui-Han Chin, Jonathan Leonard Long, and Jeremy Maitin-Shepard.

In addition to my labmates, I have had the pleasure of getting to know many fantastic friends from CMU. I learned a lot about vision, learning, psychology, philosophy, and many other subjects in the hundreds of coffeeshop conversations that I shared with Tomasz Malisiewicz. I would like to personally thank Jack Bauer for giving me opportunities to spend time with Boris and Dana Sofman, Mark Palatucci, Hanns Tappeiner, and Dan Munoz. Great times were always had whenever my RI friends were around, including: Pete Barnum, Lillian Chang, Eakta Jain, Jenney and Matt McNaughton, Ram Ravichandran, Kristina Rohlin, Vini Varadharajan, Kevin Yoon, and Ling Xu. I also enjoyed spending time with my MLD/LTI/CS friends, including Mary McGlohon, Austin McDonald, Purna Sarkar, Emily Stiehl, and Yang Xu.

Despite the workload, I have also managed to keep in touch with some of my closest friends from before graduate school. They did not help me to finish my thesis any faster, but I would like to still thank them for good times and friendship. They are: Dustin Arendt, Eric Chase, Matt Falkenhagen, Fontaine Lam, Karen Law, Laura Savidge, as well as my drawmates: Arturo Caballero, Jamie Funamura, Karl Goodman, Marisa Juarez, Carlos Nunez, and Dave Singh. Hopefully I will see many of you before our next reunion!

In addition to my friends from school, I had been fortunate to be part of a larger spiritual family in Pittsburgh. There are way too many people to name here, so I'll just name a few to whom I am particularly appreciative: Spencer Ying, Frank Lin, Sunny Shen, Doris Chen, Jun Li, Arnold Saha, Mannie Chiu, Jin Hyuk Hong, Salvador and Alice Tarun, Charles Guo, Christine Zhao, Gigi Li, Abraham and Jennifer Lu, Rebecca Lim, Ming-kai Hsieh and Ming-yu Lin, Wilson Sung, Carl and So-young Paradis, Kwan and Young Lee. I am especially thankful for Tom and Ellen Piccone who have always welcomed us into their home. And I want to thank Leon Wang for putting up with me as a roommate for five years.

Many of my happiest moments in Pittsburgh have been shared with Lucia Castellanos Pérez-Bolde who has been my best friend throughout the

last several years. *Muchísimas gracias*, from the depths of my heart, for your care, encouragement, warmth, and laughter. I am excited about sharing many more happy times in the future.

In addition to my immediate family, I am blessed to be a member of a very large extended family that happens to be uncommonly tight-knit. They are also far too numerous to name here, but here is a big thank-you to my grandparents, aunts, uncles, and cousins — particularly to my Uncle Hsi-Zen, who recently passed away. I would like to thank my brother, who is also moving out to California soon, for being my brother, and for keeping me balanced by telling me about music and snowboards. And last but not least, I would like to express gratitude to my parents, Billy and Janet Huang, for working hard to put both of us through school, and for providing a lifetime of love and encouragement. From them I have learned the value of hard work and perseverance, but more importantly, that in the end, there are far more important things than work.

CONTENTS

Part I

INTRODUCTION AND BACKGROUND

1 INTRODUCTION

PERMUTATIONS are all around us. From the configuration of a deck of cards [11], to preference rankings of sushi [70], even to the ordering of rings in a bell tower [14], permutations play a role in many aspects of our lives and it is no surprise that these combinatorial objects arise as a central topic of study in modern mathematics.

For all the attention that has been rendered to the study of permutations over the last centuries, however, it has not been until recent decades that mathematicians and statisticians have devoted significant effort to studying probability distributions on permutations, which arise nearly as frequently as permutations themselves. Since the 1980s, statisticians (such as Persi Diaconis) have tackled a broad range of issues on probabilistic modeling, learning and hypothesis testing for distributions over permutations.

This thesis comes at similar questions about probabilistic reasoning and learning on permutations from both the statistical *and* computational perspectives of machine learning. We address problems of how one might tractably represent, reason with, and estimate distributions over the space of permutations in both the statistical *and* algorithmic senses. In particular, we propose:

- Principled and compactly representable approximations of probability distributions over the group of permutations,
- Flexible probabilistic reasoning and learning algorithms which exploit the structure of these compact representations for efficiency, and
- Theoretical analysis of approximation quality as well as algorithmic and sample complexity.

1.1 APPLICATIONS

This thesis also focuses on a number of real permutation datasets, taken from diverse application domains. Below we outline a number of scenarios in which permutations can arise in real problems.

1.1.1 *Card games*

Historically, distributions over permutations were perhaps first studied in the context of card games and gambling. In the context of cards, a permutation can be thought of as a configuration of a deck of cards ("King of Hearts on top, 3 of Spades second, etc.") and with uncertainty emerging due to partial observability and some number of shuffles that typically precede a card trick or game. Some questions that typically arise are: "What

suite does the top card in the deck most likely belong to?", or "What is the probability that an opponent has a full house?", or "How many shuffles are required to sufficiently randomize a deck?". Bayer and Diaconis [11] analyzed the commonly used *riffle shuffle* (see Figure 2a) and remarkably found that seven typical shuffles are sufficient to bring the distribution over configurations of a standard 52 card deck to be close to uniform (and that further shuffles do not help very much). The card shuffling problem has been tackled in a number of papers over the last two decades. See, for example, Aldous and Diaconis [4], Diaconis [29], and Bayer and Diaconis [11].

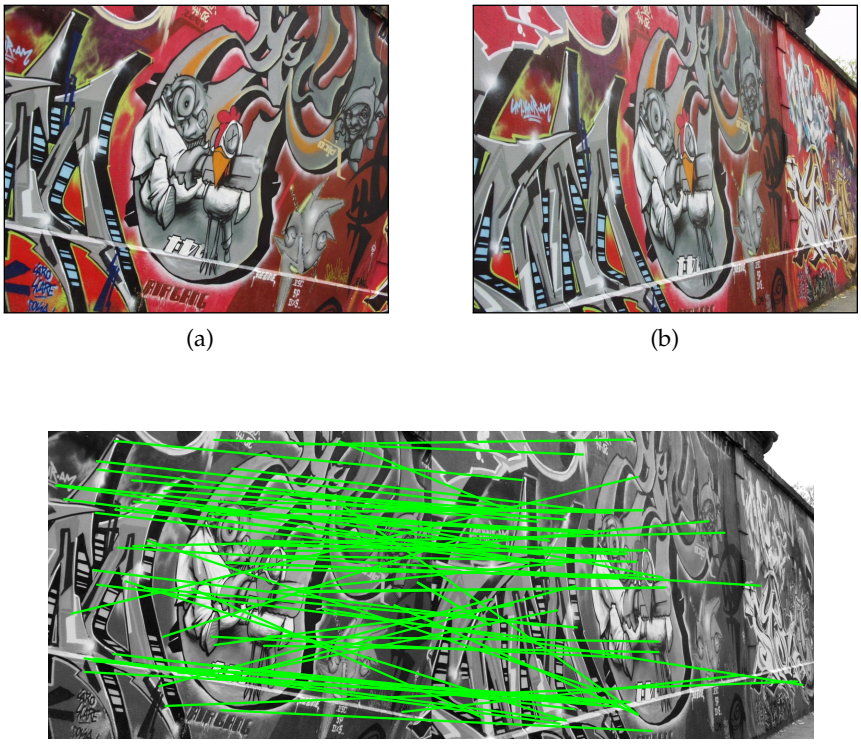
1.1.2 *Preference Rankings*

Distributions over rankings (and votes) arise in a multitude of information retrieval tasks. Based on user information, prior site visits, mouseclicks, and other possible information, one would like to produce a ranking of websites, preferred films or books for a given search query. Distributions over rankings arise due to variations in preferences among different people. For example, some people prefer to rent romantic comedies, while others enjoy action movies. Such population effects are not captured by a single ranking but rather, an entire probability distribution over rankings. Distributions can also arise due to uncertainty over the preference relations of a particular person. For example, ranking data is more often than not only available in partial form, where users provide *partially* specified rankings ("My five favorite movies in no particular order are..."). Data are sometimes available in the form of *ratings* rather than direct rankings and it often makes sense to convert them to rankings before data analysis since ratings across people are often incompatible ("My perfect 10 might not be your perfect 10"). Such ranking problems have been studied by many statisticians over the past decades including: [30, 26, 36, 91, 84, 86, 126, 28]

Similarly, rankings occur in a number of political election settings. While many elections such as plurality voting in the Unites States are based on 'first-past-the-post' rules, voting systems based on ranking candidates in order of preference are also common in a number of organizations, from countries such as Ireland and Malta, to smaller groups such as the American Psychological Association and the American Academy of Motion Picture Arts and Sciences (which decides the winners of the Oscar awards).

1.1.3 *Tracking, matching and identity management*

Consider the problem of tracking n objects (vehicles or people, for example) based on a set of noisy measurements of identity and position. A typical tracking system might attempt to manage a set of n tracks along with an identity corresponding to each track, in spite of ambiguities from imperfect identity measurements. When the objects are well separated, the problem is easily decomposed and measurements about each individual object can be clearly associated with a particular track. When objects pass near each



(a) (b)

(c)

Figure 1: Feature point matching example: (a) and (b) are images of the same scene taken from two similar viewpoints. (c) shows seventy detected *SIFT* feature points [89] in each image and a possible match (correspondence) between the two sets of points.

other, however, confusion can arise as their signal signatures may mix; see Figure 2c. After the individual objects separate again, their positions may be clearly distinguishable, but their identities can still be confused, resulting in identity uncertainty which must be propagated forward in time with each object, until additional observations allow for disambiguation. This task of maintaining a belief state for the correct association between object tracks and object identities while accounting for local mixing events and sensor observations, was introduced in [120] and is called the *identity management problem*. Identity management and the closely related problem of date association have been addressed in a number of previous works, including: [8, 9, 21, 25, 106, 105, 109, 103, 112, 115, 116, 121, 47].

1.2 DEALING WITH FACTORIAL POSSIBILITIES: TWO RELATED PROBLEMS

In each of the above problems, one must deal the fact that there are factorially many possible rankings, which poses a number of significant challenges

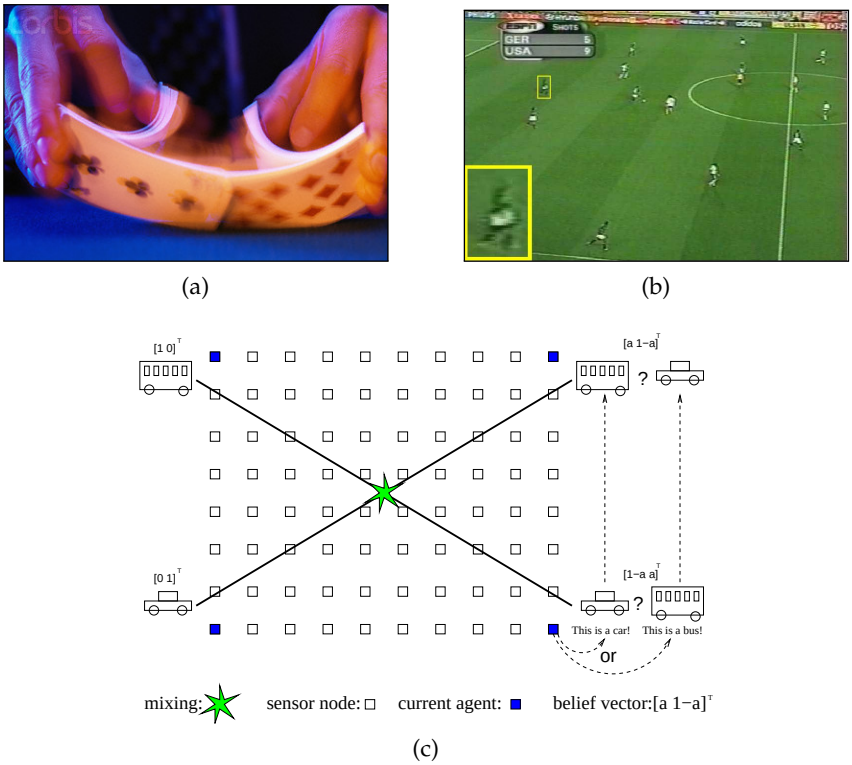


Figure 2: Card shuffling: (a) demonstrates the standard "riffle-shuffle" on a deck of cards; Identity management examples: (b) illustrates a multiperson tracking scenario (soccer game) where identity information is weak due to low-resolution imagery (image from [34]), and (c) (image from [47]) illustrates a "mixing" event in which two tracks swap identity with some probability.

for learning and inference. We address two related problems in this thesis: *representation* and *inference*.

REPRESENTATION Since there are $n!$ permutations of n items, simple tabular representations of distributions over permutations are intractable for even moderately sized n. Storing an array of 12! doubles, for example, requires roughly 14 gigabytes of storage, which is beyond the RAM capacity of a typical modern PC. A tabular representation of distributions over a standard deck of 52 cards requires storing over 8×10^{67} probabilities.

By the *representation problem*, we refer to the challenge of identifying realistic learning biases, model simplifications, and problem decompositions which allow for a distribution over permutations to be efficiently represented in polynomial space. More than being just a storage problem, we also desire representations which can be learned efficiently with polynomial training examples. For nontrivial n, it is impractical to hope that each of the $n!$ possible permutations would appear even once in a training set. It is said (Su [123]), for example, that any configuration of a deck of 52 cards achieved through random shuffling has most likely never been seen before in the history of card shuffling! The only existing datasets in

which every possible permutation is realized are those for which $n \leqslant 5$, and in fact, the APA dataset (American Psychological Association), which we discuss at various points throughout the thesis is the only such dataset for $n = 5$ that we are aware of.

INFERENCE. Even if we knew how to solve the representation problem, however, we would still need to contend with the fact that the naive algorithmic complexity of common probabilistic operations is also intractable for such distributions. Computing the marginal probability in a preference ranking problem, $h(\sigma(\mathtt{Corn}) < \sigma(\mathtt{Peas}))$, that Corn is preferred over Peas, for example, requires summing h over the rankings which place Corn before Peas (the collection over which has $O((n-2)!)$ elements).

With respect to the *inference problem*, we are interested in finding ways to exploit any underlying structure within our chosen probabilistic representations to formulate accurate and efficient algorithms for performing a number of probabilistic operations such as marginalization, conditioning, and normalization, etc.

1.3 EXPLOITING STRUCTURAL DECOMPOSITIONS: TWO RELATED APPROACHES

A pervasive technique in machine learning for making large problems tractable is to exploit conditional independence structures for decomposing large problems into much smaller ones. It is the structure of conditional independence which has made *graphical models* (such as Bayesian networks and conditional random fields) so ubiquitous in machine learning and AI [74]. Unfortunately, graphical models over permutations do not lead to decompositions which can be tractably represented in storage and therefore necessitate alternative methodologies.

Instead, in this thesis, we address the representation and inference problems by decomposing distributions over permutations via two alternative approaches: *additive (Fourier transform based) decompositions* and *multiplicative (probabilistic independence based) decompositions*.

- In the *additive decomposition*, a distribution h is approximated by a weighted linear combination of Fourier basis functions defined over permutations. These Fourier basis functions are ordered by complexity, from low-frequency functions capturing large-scale, global effects, to high-frequency functions capturing more localized, small-scale effects. Bandlimited approximations, which maintain weights only for a fixed set of low-frequency basis functions can therefore be used as a compact way of representing distributions over permutations.

- In the *multiplicative decomposition*, a distribution h is approximated as a product of factors defined over rankings of small sets of items. While conditional independence assumptions are typically unrealistic for, say, ranking applications, we propose a simple alternative generalization of independence (called *riffled independence*) which is

more effective and realistic as a compact approximation. Explicitly restricting the possible probabilistic dependencies among variables via such a decomposition yields interpretable models as well as exponential reductions in storage complexity, since each factor can often be represented compactly.

As we discuss, certain probabilistic inference operations are often easier to perform with respect to one representation than the other. An underlying theme throughout, however, is the idea that both kinds of structural decompositions can often be employed in tandem to relax the apparent intractability of probabilistic reasoning over the space of permutations. Using these decompositions, we develop general, efficient techniques applicable to multiple domains. For example, we apply the methods developed in this thesis both to solving challenging camera tracking scenarios as well as to reveal voting patterns in Irish elections which had never been identified before.

1.4 THESIS CONTRIBUTIONS

This thesis is about developing efficient representations for probability distributions over permutations, and inference algorithms which exploit the structure of these representations for efficiency. Specifically, we study two complementary approaches which decompose functions over permutations additively or multiplicatively into simpler, more compact functions.

EFFICIENT FOURIER BASED REPRESENTATIONS. Taking inspiration from signal processing, this thesis extensively explores the idea of additively decomposing distributions over permutations into a *weighted sum of low frequency Fourier basis functions*. Unlike the common discrete Fourier transform appearing ubiquitously in digital applications, Fourier transforms on permutations are a modern development based on group theory and until recently, have only been studied theoretically [29]. The work of this thesis, represents one of the first successful applications of these modern Fourier analytic methods to machine learning.

EFFICIENT FOURIER BASED INFERENCE. The Fourier theoretic perspective offers a new approach for approximate probabilistic inference — in particular, it suggests ignoring high-frequency coefficients and running probabilistic inference operations by working only with the low frequency Fourier coefficients. To this end, we introduce fast algorithms for performing probabilistic inference using these Fourier based representations. As an example, the prediction operation of hidden Markov model inference can be written as a convolution over permutations — written in the Fourier domain, however, it becomes a pointwise product of Fourier coefficients.

Performing probabilistic reasoning algorithms with a truncated Fourier transform can, unsurprisingly, lead to errors. For a number of common probabilistic reasoning operations, we provide theoretical results illuminating the nature of error propagation in the Fourier domain. Finally, a

surprising and theoretically significant result is that many of our algorithms can be applied, essentially unchanged, to performing probabilistic reasoning operations over *any finite or compact Lie group*. For example, it may seem that representing distributions over permutations should require substantially different mathematics compared to representing distributions over the rotation matrices or quaternions (which are fundamental to applications in computer vision and robotics). However, our work shows that all of these exotic but useful spaces can really be treated within the same mathematical framework.

EXPLOITING FULLY INDEPENDENT DECOMPOSITIONS FOR EFFICIENT INFERENCE. Though polynomial in size, the complexity of Fourier based representations can grow rapidly, thus posing an obstacle to efficiency that becomes especially prominent for scenarios in which one must maintain reasonable estimates for peaked distributions. When the distribution is sharp, it sometimes makes more sense to decompose the problem into smaller problems over subsets of objects and to reason about these disjoint subsets of objects independently of each other. Such independence based decompositions are what we refer to as *multiplicative decompositions*.

An even better strategy is to simultaneously take advantage of additive *and* multiplicative structure. To do this, we propose algorithms which operate entirely in the Fourier domain for combining factors to form a joint distribution and factoring a distribution, respectively. We discuss a method for detecting probabilistic independence using the Fourier coefficients of a distribution and apply our methods to adaptively decompose large identity management problems into much smaller ones, improving previous methods both in scalability and approximation quality.

RIFFLE INDEPENDENT FACTORIZATIONS. While full independence assumptions can sometimes be appropriate for modeling moving groups of objects, we show that independence assumptions impose strong sparsity constraints on distributions and are unsuitable for modeling ranking data. We identify a novel class of independence structures generalizing full independence, called riffled independence. Compared to fully independent assumptions, riffled independence captures a more expressive family of distributions while retaining many of the properties necessary for performing efficient inference and reducing sample complexity Experimentally, we show that while items in real ranking datasets are never independent in the ordinary sense, there are a number of datasets in which items exhibit approximate *riffled independence*.

As with full independence, we provide algorithms that can be used in conjunction with Fourier based decompositions, allowing for both types of structure to be exploited in conjunction. Specifically, we propose algorithms for joining riffle independent factors and for teasing apart the riffle independent factors from a joint distribution

HIERARCHICAL FACTORIZATIONS AND STRUCTURE LEARNING. Taking inspiration from graphical models, which factor according to conditional independence relationships, we introduce a class of probabilistic models for rankings that factor hierarchically based on riffled independence relationships. Given a training set of rankings, we propose an automated method for learning the structure of these *hierarchical riffle independent models* from a training set of rankings, and show that our clustering-like algorithms can be used to discover meaningful latent coalitions from real ranking datasets. In Irish election datasets, for example, where voters provide rank-orderings of candidates, we show that the many Irish political factions are often near riffle independent of each other, thus revealing a hidden voting pattern that had previously gone unnoticed.

EXPLOITING RIFFLED INDEPENDENCE FOR EFFICIENT INFERENCE. Ranking data is often collected from humans in the form of elections and preference surveys. Due to the fact that it is typically difficult and inconvenient for humans to specify long preference lists, however, it is common for ranking data to come in the form of *partial rankings*. Consequently, an important class of inference problems arising in ranking is that of inferring a distribution over items given a partial ranking of the items — for example, we may desire an estimate of a user's preference profile over movies given his top ten favorite movies.

We establish a fundamental connection between the notion of riffled independence and partial ranking, proving first that it is possible to exploit riffled independence relationships among items to condition on partial rankings in time linear in the number of model parameters. We show that general inference queries which would ordinarily be amenable to efficient inference under ordinary independence assumptions are not efficiently handled under riffled independence assumptions. And in fact, we are able to establish in a rigorous sense that partial rankings are the *only* observations for which one can exploit riffled independence for efficient condition.

1.5 THESIS ORGANIZATION

The thesis is organized into four parts. See Figure 3 for a roadmap of the thesis, showing how each chapter addresses one or both of our two main problems of representation and inference using either additive or multiplicative decompositions.

PART I. In the remainder of the first part of the thesis, we introduce notation and many of the basic properties of permutations that will be used throughout this thesis (**Chapter 2**). We introduce the symmetric group, distributions over the symmetric group as well as some of the basic probabilistic operations that one might want to perform using these distributions (**Chapter 3**).

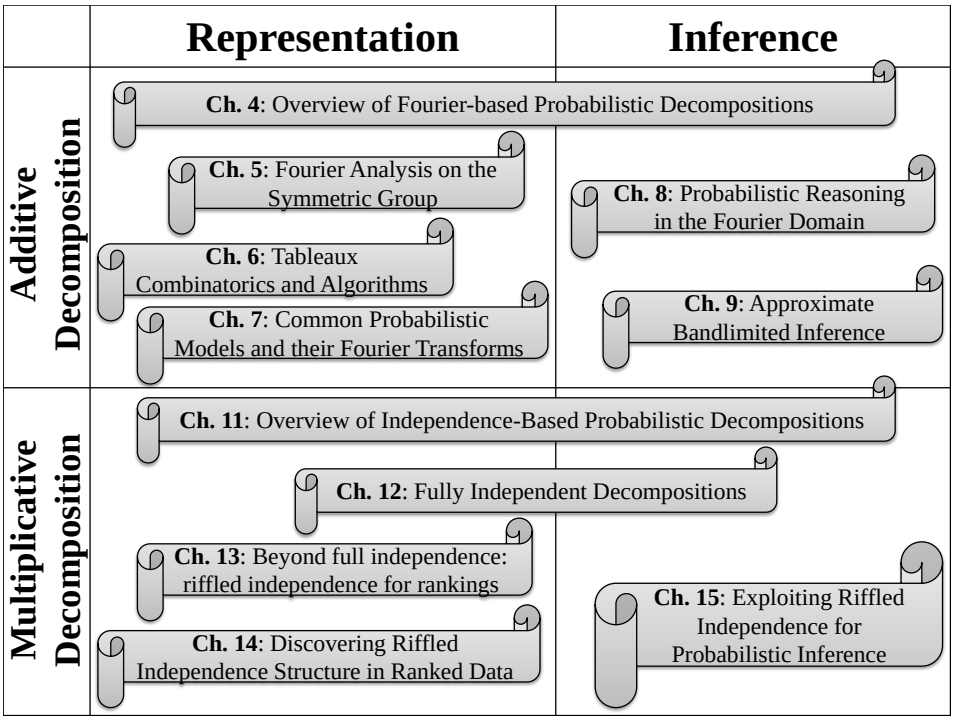


Figure 3: A roadmap of this thesis, outlining how each chapter addresses one or both of our two main problems (*representation* and *inference* with distributions over permutations) using one of the two main decompositions (*additive* and *multiplicative*).

PART II. We discuss *additive decompositions* of distributions over permutations. By projecting a distribution to low-frequency Fourier basis functions, we are able to achieve compact representations of probability distributions over permutations. We provide an accessible overview of group representation theory, the foundation upon which Fourier analysis for the symmetric group is based (**Chapter 5**), as well as specific representation theoretic algorithms for the symmetric group (Chapter 6). Using the tools of representation theory, we construct Fourier transforms of common probabilistic models (**Chapter 7**) and formulate efficient Fourier theoretic counterparts of the probabilistic operations described in Part I (**Chapter 8**). We discuss issues concerning approximation quality (**Chapter 9**), where we also present an empirical evaluation of our methods on challenging multi-target tracking problems.

PART III. We discuss how *multiplicative decompositions*, based on the notion of probabilistic independence, can additionally be used as a complementary approach in obtaining even more efficient probabilistic representations and inference algorithms (**Chapter 12**). We introduce a novel generalization of probabilistic independence called *riffled independence* (**Chapter 13**). By exploiting riffle independent relationships among ranked items, we obtain structure and parameter learning algorithms that are efficient both with respect to running time and sample complexity (**Chapter 14**). By es-

tablishing a fundamental theoretical connection between partial ranking and riffled independence, we are able to extend our algorithms to learn models from partially ranked data, which is far more common than fully ranked data (**Chapter 15**). For both ordinary probabilistic independence and riffled independence, we design efficient Fourier theoretic algorithms capable of computing factors and joint distributions from low-order Fourier coefficients.

PART IV. Finally we discuss a number of remaining open problems (**Chapter 17**). We conclude by summarizing the major contributions of this thesis and reiterating the major mathematical themes that buttress the thesis (**Chapter 18**).

2 PERMUTATIONS AND THE SYMMETRIC GROUP

In this chapter, we provide background on the group of permutations and introduce many of the preliminaries as well as terminology for this thesis. In particular, we introduce a number of the basic algorithms associated with the symmetric group. We remark that all of the definitions as well as theoretical results in this chapter appear in most modern and classical algebra textbooks such as [83, 32], but are here presented from a perhaps more algorithmic viewpoint.

2.1 PERMUTATIONS AND THEIR MANY NOTATIONS

A permutation on n elements is a one-to-one mapping of the set $\Omega = \{1, \ldots, n\}$, into itself and can be written as a tuple,

$$\sigma = [\sigma(1) \;\; \sigma(2) \;\; \ldots \;\; \sigma(n)],$$

where $\sigma(i)$ denotes where the ith element is mapped under the permutation. For example, $\sigma = [2\ 3\ 1\ 4\ 5]$ means that $\sigma(1) = 2$, $\sigma(2) = 3$, $\sigma(3) = 1$, $\sigma(4) = 4$, and $\sigma(5) = 5$. The tuple $[1\,1\,2\,5]$, on the other hand, is *not* a proper permutation since it is not one-to-one. This way of notating permutations is called *one-line notation*, and is perhaps the most convenient notation for programming.

The set of all permutations on n elements forms a group[1] under the operation of function composition — that is, if σ_1 and σ_2 are permutations, then

$$\sigma_1\sigma_2 = [\sigma_1(\sigma_2(1)) \;\; \sigma_1(\sigma_2(2)) \;\; \ldots \;\; \sigma_1(\sigma_2(n))],$$

is itself a permutation. Additionally, for every permutation σ, there is an inverse permutation σ^{-1} since permutations are one-to-one mappings. Finally, the identity permutation is the permutation ϵ which maps every element to itself (i.e., $\epsilon = [1\,2\,\ldots\,n]$). See Algorithms 2.1 and 2.2 for pseudocode implementing permutation composition and inversion, respectively.

Definition 1. The set of all $n!$ permutations is called the *symmetric group*, or just S_n. In addition to the set $\{1, \ldots, n\}$, we will also consider permutations of a general set Ω, in which case we notate the corresponding group as S_Ω.

Example 2. *There are* $4! = 24$ *elements in the group* S_4*. Consider two elements of* S_4, $\sigma_1 = [1\,3\,4\,2]$ *and* $\sigma_2 = [3\,1\,2\,4]$*. Then* $\sigma_1\sigma_2 = [4\,1\,3\,2]$*. The inverse of* σ_1 *is* $\sigma_1^{-1} = [1\,4\,2\,3]$.

Notably, the group operation for permutation is noncommutative*, and so* S_n *is not an abelian group for* $n \geqslant 3$*. With* σ_1 *and* σ_2 *set as above, We have, for example,* $\sigma_1\sigma_2 = [4\,1\,3\,2]$*, but* $\sigma_2\sigma_1 = [3\,2\,4\,1]$.

1 See Appendix A for a list of the basic group theoretic definitions used in this thesis.

Algorithm 2.1: Algorithm for composing two permutations (one-line notation). Input: two elements $\sigma_1, \sigma_2 \in S_n$, Output: $\sigma_1\sigma_2$.

```
COMPOSE(σ1, σ2):

    Initialize result to be an integer array of length n;
    for i = 1, ..., n do
        result ← σ1[σ2[i]];
    end
    return result ;
```

Algorithm 2.2: Algorithm for inverting a permutation (one-line notation). Input: $\sigma \in S_n$, Output: σ^{-1}.

```
INVERT(σ):

    Initialize result to be an integer array of length n;
    for i = 1, ..., n do
        result[σ[i]] ← i;
    end
    return result ;
```

2.1.1 *Cycle notation*

Another common way to notate the elements of S_n uses the perhaps more standard *cycle notation*, in which a *cycle* $(i, j, k, \ldots, \ell)$ refers to the permutation which maps i to j, j to k, $\ldots$, and finally ℓ to i. Though not every permutation can be written as a single cycle, any permutation can always be written as a product of disjoint cycles. For example, the permutation $\sigma = [2\,3\,1\,4\,5]$ written in cycle notation is $\sigma = (1,2,3)(4)(5)$. The number of elements in a cycle is called the *cycle length* and we typically drop the length 1 cycles in cycle notation when it creates no ambiguity — in our example, $\sigma = (1,2,3)(4)(5) = (1,2,3)$. Items falling in the length 1 cycles of a permutation σ are referred to as the *fixed points* of σ. The *cycle type* of a permutation σ is the (unordered) tuple of cycle lengths of σ. For example, the cycle type of $\sigma = (1,2,3)(4,5)$ is $(3,2)$. See Algorithm 2.3 for a simple algorithm which converts one-line notation to cycle notation.

There are several advantages which sometimes make cycle notation more convenient than one-line notation. *Transpositions*, or *swaps* of two elements i and j, for example, can simply be denoted as a two-cycle (i, j). It is also particularly simple to invert a permutation which is written in cycle notation by simply "flipping" each cycle. For example, if $\sigma = (1,2,3)(4,5)$, then its inverse is $\sigma^{-1} = (3,2,1)(5,4)$.

2.1.2 *Ordering/Vertical bar notation for rankings*

For many ranking applications it will be more convenient to refer to a permutation σ by specifying its corresponding inverse as $\sigma^{-1}(1)|\sigma^{-1}(2)|\ldots|\sigma^{-1}(n)$ in what is typically called *ordering* or *vertical bar notation*. We say that σ ranks item a *before (or over)* item b if the rank of a is less than the rank of b ($\sigma(a) < \sigma(b)$). The reason for using both notations is due to the fact that

Algorithm 2.3: Algorithm for converting a permutation from one-line notation to cycle notation. The output is a disjoint array of tuples (cycles) whose product $((allcycles[0])(allcycles[1])\dots(allcycles[\ell]))$ results in the given permutation σ. Note that since the cycles are disjoint, the order of multiplication does not matter here, though it will matter for later related algorithms.

toCycle($\sigma \in S_n$):

```
Initialize integer numcovered to be zero;
Initialize array covered to be a length n array of zeros;
Initialize (ordered) list allcycles to be empty;
while numcovered < n do
    Initialize integer start to be zero ;
    while covered[start] == 1 do
        start ← start + 1;
    end
    Initialize array newcycle to be a length 1 array with newcycle[1] = start;
    covered[start] ← 1;
    numcovered ← numcovered + 1;
    Initialize pos to be σ(start);
    while pos != start do
        Append pos to the end of newcycle;
        covered[pos] ← 1 ;
        numcovered ← numcovered + 1;
        pos ← σ(pos);
    end
    Append newcycle to the end of allcycles;
end
return allcycles ;
```

certain concepts will be more intuitive to express using either the ranking or ordering notation. For example

Example 3. *As one of the running example in this thesis, we will consider ranking a small list of 6 items consisting of foods enumerated below:*

1. Corn (**C**)	2. Peas (**P**)	3. Lemons (**L**)
4. Oranges (**O**)	5. Figs (**F**)	6. Grapes (**G**)

Written in one-line notation, the ranking $\sigma = [3\,1\,5\,6\,2\,4]$ *means, for example, that Corn is ranked third, Peas is ranked first, Lemons is ranked fifth, and so on. In vertical bar notation, the same ranking is expressed as:*

$$\sigma = \text{Peas}\,|\,\text{Figs}\,|\,\text{Corn}\,|\,\text{Grapes}\,|\text{Lemons}\,|\,\text{Oranges}.$$

Finally we will use $\sigma(3) = \sigma(\mathbf{L}) = 5$ *to denote the rank of the third item, Lemons.*

2.1.3 *Mappings between distinct input/ouput domains*

In many applications, we will abuse notation by using 'permutation' to refer to a one-to-one mapping between two distinct domains of equal cardinality (instead of mappings of a domain to itself). For example, in tracking, a permutation may map a set of identities to a set of tracks. Or a permutation may equivalently map a set of tracks to a set of identities. In ranking, on the other hand, a permutation may map a collection of items or candidates to

Algorithm 2.4: Algorithm for lexicographically enumerating all elements of S_n. Input: sorted array of items, X (to enumerate S_n, $X = [1, 2, \ldots, n]$). Output: All permutations of items in X.

```
enumeratePerms(X):

    Initialize result to be an empty array (of permutations);
    if |X| == 1 then
        Append X to result;
        return result;
    end
    for i = |X|,|X|-1,...,1 do
        tmp ← enumeratePerms([X[1],...,X[i−1],X[i+1],...,X[n]]);
        Append X[i] to the end of each element of tmp;
        Append each element of tmp to result;
    end
    return result ;
```

a set of ranks. The ranking of the item set $\{\text{Corn}, \text{Peas}, \text{Lemons}\}$ for which $\pi(\text{Corn}) = 1$, $\pi(\text{Peas}) = 2$, and $\pi(\text{Lemons}) = 3$, for example, designates Corn as the most preferred item.

Since the input and output domains of a ranking are not the same in any of the above applications, it does not make sense to, for example, compose two rankings. For this reason, the collection of rankings (as well as mappings between identities and tracks) does not technically form a group. To treat one-to-one mappings between distinct domains, one can simply fix some *reference ranking* (or a reference numbering) of the input and output domains. With respect to this reference ranking, it is possible to associate every ranking with a unique permutation which can be thought of as the 'deviation' of the ranking from the original reference ranking. In the parlance of group theory, there is a faithful group action of S_n on the collection of rankings.

Example 4. *If we fix a reference ranking* π_0 *such that* $\pi_0(\text{Corn}) = 1$, $\pi_0(\text{Peas}) = 2$, *and* $\pi_0(\text{Lemons}) = 3$ *then the permutation* $\sigma = [3\,2\,1]$ *is associated to the ranking which maps* Corn *to* $\sigma(\pi_0(\text{Corn})) = 3$, Peas *to* $\sigma(\pi_0(\text{Peas})) = 2$, *and* Lemons *to* $\sigma(\pi(\text{Lemons})) = 1$.

In the examples which can be found in this thesis, we will typically fix the reference ranking of a set of items to be alphabetical. For example, the reference ranking of the set $\Omega = \{\text{Corn}, \text{Peas}\}$ is always assumed to be $\pi_0(\text{Corn}) = 1$ and $\pi_0(\text{Peas}) = 2$. This issue of the reference ranking will arise in a number of applications as it will be necessary in many algorithms to 'reorder' the items, or change the underlying reference ranking.

2.2 ENUMERATION AND INDEXING

In this thesis we will typically enumerate permutations with respect to a reverse lexicographical ordering obtained when reading the one-line notation of each permutation from right to left. For simplicity, we will refer

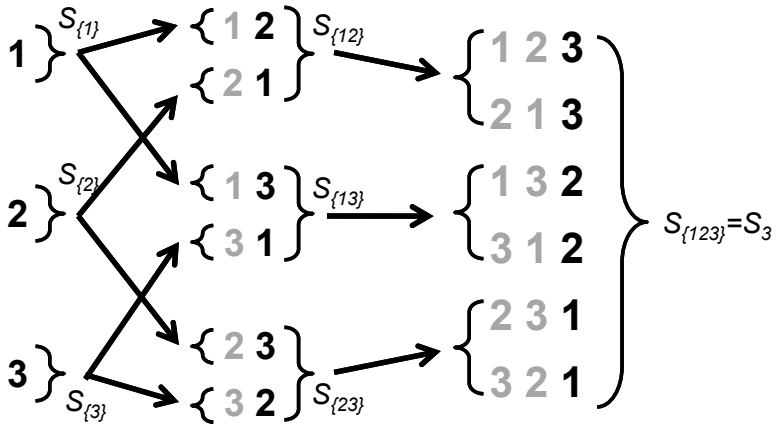


Figure 4: Structure of symmetric group allows for recursive enumeration.

to the ordering simply as the "lex" ordering. For example, the lex ordering of S_3 is:

$$[1\,2\,3],\ [2\,1\,3],\ [1\,3\,2],\ [3\,1\,2],\ [2\,3\,1],\ [3\,2\,1].$$

There is a simple recursive method for enumerating elements of S_n with respect to lexicographical order (given in Algorithm 2.4). The recursion is based on fixing item n to map to $n, n-1, \ldots, 1$, and each for each fixed target, enumerating the permutations of a smaller group isomorphic to S_{n-1}. For example, to enumerate the elements of S_3, one fixes $\sigma(3) = 3$ and enumerates the permutations of $\{1, 2\}$, then fixes $\sigma(3) = 2$, and enumerates the permutations of $\{1, 3\}$ then finally fixes $\sigma(3) = 1$ and enumerates the permutations of $\{2, 3\}$. See Figure 4 for a diagram of the recursive structure in the lexigraphical enumeration algorithm.

Note that since there are $n!$ factorial permutations, one rarely uses Algorithm 2.4 to enumerate over all permutations (except for very small n). Instead, it is more useful for debugging and 'sanity check' purposes. Algorithm 2.5 provides a related algorithm which, given any element of S_n, returns its index within the lex ordering of S_n. We have also chosen to present the enumeration algorithm here because the same recursive structure in which one reduces a problem over S_n to n problems over S_{n-1} is mirrored in many algorithms throughout the thesis. In particular, the FFT (Fast Fourier transform which we present in later chapters) will rely on a very similar algorithmic decomposition.

2.3 GENERATING SETS OF THE SYMMETRIC GROUP

The fundamental computational challenge that this thesis addresses is that of dealing with the factorial number of possible permutations. One of the main insights which we rely upon repeatedly throughout to work with such a large collection of objects is the idea that permutations can be always be decomposed into the composition of a number of simpler 'parts'.

Algorithm 2.5: Algorithm for computing the index of a permutation with respect to lexicographical ordering. Here end refers to the last element of σ (viewed as an array), and $\sigma_{1:end-1}$ is the array obtained by dropping the last element of σ.

```
permToIndex(σ ∈ S_n):

    if |σ| == 1 then
        return 1;
    end
    j ← (|σ| − 1)! · |{i : σ(i) > σ(end)}| ;
    return j+permToIndex (σ_{1:end−1}) ;
```

Formally, we accomplish these decompositions through the use of *generating sets* of a group.

Definition 5. A finite group G (think S_n) is *generated* by a *generating set* $X = \{\sigma_1, \ldots, \sigma_m\}$ if every element of G can be written as a finite product of elements in X and their inverses; i.e., every element $g \in G$ can be written:

$$g = x_1^{\epsilon_1} x_2^{\epsilon_2} \cdots x_\ell^{\epsilon_\ell},$$

where $x_i \in X$ and $\epsilon_i \in \{+1, -1\}$ for all i.

Of course, the entire group S_n is a valid generating set, but we are especially interested in small generating sets, in which we can understand each permutation as being constructed from a set of only, say, a handful of permutations. In particular, we will argue that the following three sets are all generating sets for S_n:[2]

- **All pairwise transpositions**:

$$X_a = \{(i,j) : 1 \leqslant i < j \leqslant n\},$$

- **Adjacent pairwise transpositions**:

$$X_b = \{(i,i+1) : 1 \leqslant i < n\},$$

- **Two-element generating set**:

$$X_c = \{(1,2), (1,2,3,\ldots,n)\}.$$

Example 6. *As an example, we note that the permutation $\sigma = (1,4,2,3)$ can be factored with respect to each of the above generating sets:*

$$\begin{aligned}(1,4,2,3) &= (2,3)(1,3)(1,4), \text{ (pairwise transpositions)}\\ &= (2,3)(1,2)(3,4)(2,3)(1,2), \text{ (adjacent transpositions)}\\ &= \beta\alpha\beta^3\alpha\beta^2\alpha\beta^3\alpha\beta^3\alpha. \text{ (where } \alpha = (1,2), \text{ and } \beta = (1,2,3,4))\end{aligned}$$

2 Note that for our discussion of generating sets, we will rely mostly on cycle notation for convenience.

Algorithm 2.6: Algorithm for decomposing a permutation σ into a product of transpositions of the form (i, j).

```
permToSwaps(σ ∈ S_n)

(a_1^(1), ..., a_{m^(1)}^(1))(a_1^(2), ..., a_{m^(2)}^(2)) ... (a_1^(ℓ), ..., a_{m^(ℓ)}^(ℓ)) ← toCycle(σ) ;
Initialize (ordered) list allcycles to be empty;
for i = 1, ..., ℓ do
    for j = m^i, ..., 2 do
        Append the transposition (a_1^(i), a_j^(i)) to the end of allcycles;
    end
end
return allcycles ;
```

The idea that permutations can be factored into a sequence of elements in a generating set is so useful that we will discuss each of the above generating sets in some detail. In the following, we present factoring algorithms which express a given permutation with respect to each generating set above.

We also present bounds on the number of generating elements required to construct a given permutation. We note that there are often many ways to express any given permutation in terms of a generating set and so the algorithms given below only produce one of many valid decompositions. The minimum such number of terms plays a role in the analysis of complexity for some of the algorithms in the thesis.

Definition 7. Given a generating set X of a group G, the *wordlength* of an element $\sigma \in G$ with respect to X, $L_X(\sigma)$, is the minimum ℓ for which σ is expressible as $g = x_1^{\epsilon_1} x_2^{\epsilon_2} \cdots x_\ell^{\epsilon_\ell}$, where $x_i \in X$ and $\epsilon_i \in \{+1, -1\}$ (see Definition 5).

2.3.1 *From pairwise transpositions*

We first show an easy algorithm for writing any permutation as product of pairwise transpositions based on the following lemma, which shows that any length m cycle can be written as a product of $m-1$ transpositions.

Lemma 8. *The cycle $(a_1, a_2, \ldots, a_m)$ factors as a product of $m-1$ transpositions.*

Proof.

$$(a_1, a_2, \ldots, a_m) = (a_1, a_m)(a_1, a_{m-1})(a_1, a_{m-2}) \cdots (a_1, a_2).$$

□

To decompose a given permutation σ as a finite product of cycles, one can simply use Lemma 8 (Algorithm 2.3) to replace each cycle by a finite product of transpositions. See Algorithm 2.6 for pseudocode. How many transpositions are required to express any permutation? We now show that the answer is: no more than $n-1$.

Algorithm 2.7: Algorithm for decomposing a permutation σ into a product of *adjacent* transpositions of the form $(i, i+1)$. Note that this algorithm is very similar to the well-known Bubblesort algorithm.

```
permToAdjSwaps(σ ∈ S_n):

    for i = 1, 2, ..., n do
        for j = n, n − 1 ..., i + 1 do
            if σ[j] < σ[j − 1] then
                swap (σ[j], σ[j − 1]);
                Append transposition (j, j + 1) to the beginning of result;
            end
        end
    end
    return result ;
```

Proposition 9. *The maximum wordlength with respect to X_a (the set of pairwise transpositions) bounded above by $n-1$ ($\max_{\sigma\in S_n} L_{X_a}(\sigma) \leqslant n-1$).*

Proof. The length of the decomposition of a permutation σ produced by Algorithm 2.6 is an upper bound on the wordlength of σ and cannot exceed $n-1$ since the cycle decomposition of Algorithm 2.3 produces disjoint cycles, and the reduction of a length m cycle to transpositions by Lemma 8 results in $m-1$ terms. On the other hand $n-1$ is the wordlength of the worst case which occurs when σ takes the form of an n-cycle, in which case the number of transpositions produced by the decomposition is exactly $n-1$. □

2.3.2 *From adjacent swaps*

Lemma 10. *For any distinct $i, j \in \{1, \ldots, n\}$, the transposition (i, j) factors as a product of adjacent transpositions.*

Proof. Without loss of generality, assume that $i < j$. Then we have:

$$(i,j) = (i,i+1)(i+1,i+2)\cdots(j-2,j-1)(j-1,j)$$
$$(j-2,j-1)\cdots(i+1,i+2)(i,i+1).$$

□

One way to write an arbitrary permutation as product of adjacent transpositions is, as before, to call Algorithm 2.6 to write a permutation σ as a product of (i, j) transpositions, then use Lemma 10 to replace each transposition as a product of adjacent transpositions. In Algorithm 2.7 we present an alternative method based on an idea similar to the *Bubblesort* algorithm for sorting.

The maximum wordlength in the case of adjacent transpositions is $O(n^2)$.

Proposition 11. *The maximum wordlength with respect to X_b is bounded above by $\frac{n(n-1)}{2}$.*

Proof. To compute an upper bound on $\max_{\sigma \in S_n} L_{X_a}(\sigma)$, it is enough to consider the worst case for any algorithm which factors permutations into adjacent transpositions. Algorithm 2.7, for example, adds at most one transposition to the factorization per iteration and terminates after $\frac{n(n-1)}{2}$ iterations, and therefore the wordlength of any permutation σ can be no greater than $\frac{n(n-1)}{2}$. □

KENDALL'S TAU DISTANCE METRIC ON RANKINGS. The number of transpositions produced by Algorithm 2.7 can be used as a distance measure on the space of permutations. Given two permutations $\sigma_1, \sigma^2 \in S_n$, the size of the decomposition of $\sigma_1 \sigma_2^{-1}$ into adjacent transpositions via Algorithm 2.7 can be thought of as the number of swaps (of adjacent elements) necessary to transform σ_1 into σ_2. Interpreting the result as a distance metric yields the well known *Kendall's tau distance metric* [71] (denoted henceforth as d_K), which is sometimes also called the *Bubblesort distance* since it reflects the number of iterations required to "Bubblesort" σ_1 to become σ_2.

Example 12. *Let* $\sigma_1 = [3\,1\,4\,2]$ *and* $\sigma_2 = [1\,2\,3\,4]$. *To compute the Kendall's tau distance between* σ_1 *and* σ_2, *we need to use Algorithm 2.7 to decompose the permutation* $\sigma_1 \sigma_2^{-1} = [3\,1\,4\,2]$, *which yields the factorization:* $[3\,1\,4\,2] = (2,3)(1,2)(3,4)$, *and therefore* $d_K(\sigma_1, \sigma_2) = 3$.

2.3.3 *A generator set with two elements*

The smallest generating set for the symmetric group, X_c, has only two elements, a transposition $(1,2)$, and an n-cycle. To show that X_c generates the entire symmetric group, it is enough to show that every adjacent transposition can be written in terms of elements in X_c:

Lemma 13. *The adjacent transposition* $(i, i+1)$ *can be written as a product of elements in* X_c.

Proof.

$$(i, i+1) = (1, 2, \ldots, n)^{i-1}(1,2)(1, 2, \ldots, n)^{n-i+1}.$$

□

While the generating set X_c has only two elements, however, it has a maximum wordlength of $O(n^3)$.

Proposition 14. *The maximum wordlength with respect to* X_c *is bounded above by* $\frac{n(n-1)(n+1)}{2}$.

Proof. There are $n+1$ elements of X_c necessary to generate an arbitrary adjacent transposition. Multiplying by at most $\frac{n(n-1)}{2}$ adjacent transpositions necessary to generate an arbitrary permutation (as guaranteed) by Proposition 11, we see that $\max_{\sigma \in S_n} L_{X_c}(\sigma) \leqslant \frac{n(n-1)(n+1)}{2}$. □

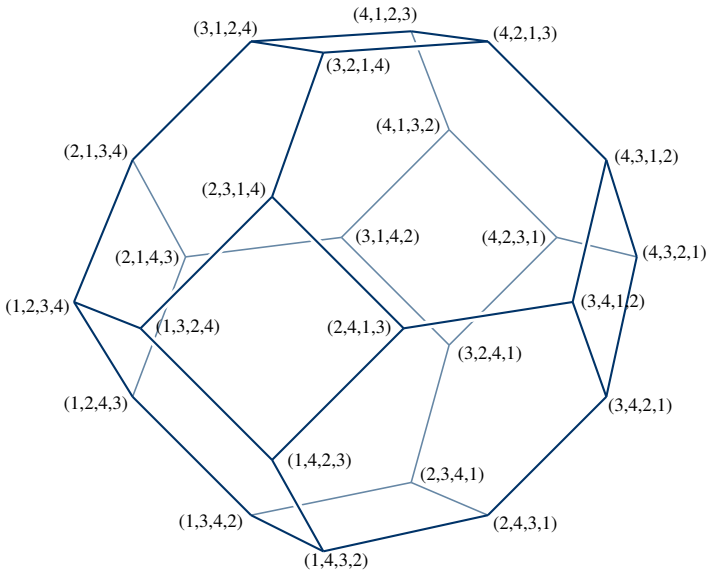


Figure 5: The Cayley graph on the group S_4 with the adjacent transpositions as generators.

2.3.4 *The Cayley graph of the symmetric group.*

A simple way to visualize the symmetric group (for very low n) is via its *Cayley graph*. Given a generating set X, the Cayley graph associates each permutation $\sigma \in S_n$ with a vertex v_σ. Additionally, for each $\sigma \in S_n$ and generator $\pi_X \in X$, there is an edge connecting σ with $\sigma\pi_X$. Figure 5 plots the Cayley graph for the symmetric group on 4 items, S_4, when the generating set consists of adjacent permutations. We will revisit Cayley graphs in our discussion of Fourier analysis in Chapter 5.

2.4 SUBGROUPS AND COSETS

Another way to factor or decompose permutations into simpler parts is to use *coset decompositions* of the symmetric group, in which the group of permutations is partitioned into 'translations' of some subgroup. There are two subgroups in particular that we rely on.

SUBGROUPS OF THE FORM $S_p \subset S_n$. We will identify S_p, the symmetric group on p items, isomorphically with the following subgroup of S_n in which all but the first p items are fixed to map to themselves:

$$S_p = \{\sigma \in S_n \ : \ \sigma(i) = i, \text{ for all } i > p\}.$$

For example, S_2 can be viewed as the subgroup of permutations in S_3 which fix item 3 to map to 3.

Left-cosets of S_p in S_n then take the form:

$$\pi S_p = \{\pi\sigma_p \ : \ \sigma_p \in S_p \subset S_n\},$$

$$\left\{\begin{array}{l}1|2|3|4,\\1|2|4|3,\\2|1|3|4,\\2|1|4|3\end{array}\right\},\quad \left\{\begin{array}{l}1|3|2|4,\\2|3|1|4,\\1|4|2|3,\\2|4|1|3\end{array}\right\},\quad \left\{\begin{array}{l}1|3|4|2,\\2|3|4|1,\\1|4|3|2,\\2|4|3|1\end{array}\right\},$$
$$\left\{\begin{array}{l}3|1|4|2,\\3|2|4|1,\\4|1|3|2,\\4|2|3|1\end{array}\right\},\quad \left\{\begin{array}{l}3|1|2|4,\\3|2|1|4,\\4|1|2|3,\\4|2|1|3\end{array}\right\},\quad \left\{\begin{array}{l}3|4|1|2,\\3|4|2|1,\\4|3|1|2,\\4|3|2|1\end{array}\right\}$$

Figure 6: The collection of left $S_2 \times S_2$-cosets of S_4. To make the interleaving intuition clearer, we write the permutations here using the vertical bar notation explained in Section 2.1.2. Items from set $A = \{1, 2\}$ are colored in red and items from set $B = \{3, 4\}$ are colored in blue.

where π is some fixed element of S_n, and can be thought of as the subset of permutations which agree with π on the last $n - p$ items of the input domain. For example, if $\pi = [3\,2\,1]$, the coset πS_2 is the collection of permutations which map item 3 to 1 since $\pi(3) = 1$:

$$\pi S_2 = \{[3\,2\,1][1\,2\,3], [3\,2\,1][2\,1\,3]\} = \{[3\,2\,1], [2\,3\,1]\} = \{\sigma \in S_n \,:\, \sigma(3) = 1\}.$$

Similarly, right-cosets of S_p take the form:

$$\pi S_p = \{\pi\sigma_p \,:\, \sigma_p \in S_p \subset S_n\},$$

where π is again some fixed element of S_n, and can be thought of as the subset of permutations which agree with π on the last $n - p$ items of the output domain. For example, if $\pi = [3\,2\,1]$, then the coset $S_2\pi$ is the collection of permutations which map item 1 to 3 to blah since $\pi(1) = 3$.

SUBGROUPS OF THE FORM $S_p \times S_q \subset S_n$. We will also refer to the subgroup $S_p \times S_q \subset S_n$ (where $p + q = n$) defined as:

$$S_p \times S_q = \{\sigma \in S_n \,:\, \sigma(i) \leqslant p \text{ for all } i \leqslant p \text{ (and } \sigma(i) > p \text{ for all } i > p)\}. \tag{2.1}$$

$S_p \times S_q$ can be thought of as the subgroup of permutations which permute the first p items amongst themselves and the remaining q items amongst themselves, but do not allow for items from one subset to map to the other subset. For example, $S_2 \times S_2$ consists of the permutations $[1\,2\,3\,4]$, $[2\,1\,3\,4]$, $[1\,2\,4\,3]$, and $[2\,1\,4\,3]$. Going beyond just two subsets, it can be generalized to $S_{p_1} \times S_{p_2} \times \cdots \times S_{p_\ell}$, where $\sum_{i=1}^{\ell} p_i = n$.

The left $S_p \times S_q$-cosets are:

$$\tau(S_p \times S_p) = \{\tau\sigma_{pq} \,:\, \sigma_{pq} \in S_p \times S_q\},$$

where again τ is some fixed element of S_n. We are particularly interested in left $S_p \times S_q$-cosets (see Chapter 13 for example) since they correspond to the collection of ways to *interleave* the two sets $A = \{1, \ldots, p\}$ and $B = \{p+1, \ldots, n\}$.

We say that two permutation τ_1 and τ_2 *interleave* A and B in the same way if the sets $\{\tau_1(1), \ldots, \tau_1(p)\}$ and $\{\tau_2(1), \ldots, \tau_2(p)\}$ (and $\{\tau_1(p+1), \ldots, \tau_1(n)\}$ and $\{\tau_2(p+1), \ldots, \tau_2(n)\}$) are equal. For example, the permutations $\tau_1 = [1\,3\,4\,2]$ and $\tau_2 = [3\,1\,2\,4]$ can be said to interleave the sets $A = \{1, 2\}$ and $B = \{3, 4\}$ in the same way. We now show that the left $S_p \times S_q$-cosets partition the symmetric group into equivalence classes in which each coset corresponds to a permutation which interleave A and B in the same way. Consequently, each left $S_p \times S_q$-coset as defined above can be identified with a unique interleaving of A and B.

Proposition 15. *Consider any two permutations $\tau_1, \tau_2 \in S_n$ which interleave A and B in the same way. Then the corresponding left $S_p \times S_q$-cosets, $\tau_1(S_p \times S_q)$ and $\tau_2(S_p \times S_q)$ must be the same.*

Proof. Let $\pi = \tau_1^{-1}\tau_2$. Since τ_1 and τ_2 interleave A and B in the same way, we have $1 \leqslant \pi(i) \leqslant p$ for all $1 \leqslant i \leqslant p$ and similarly that $p+1 \leqslant pi(i) \leqslant n$ for all $p+1 \leqslant pi(i) \leqslant n$. Therefore π is an element of $S_p \times S_q$. Finally, we have that $\tau_2(S_p \times S_q) = \tau_1\pi(S_p \times S_q) = \tau_1(S_p \times S_q)$. □

Example 16. *In Figure 6, we present the left $S_2 \times S_2$-cosets of S_4. To make the interleaving intuition clearer, we write the permutations here using the vertical bar notation explained in Section 2.1.2. As can be seen, S_4, which consists of 24 elements is partitioned into 6 $S_2 \times S_2$-cosets of 4 elements each. Each coset corresponds to a unique way of interleaving sets $A = \{1, 2\}$ and $B = \{3, 4\}$. The cosets can therefore be summarized as:*

$$\{\mathbf{A}|\mathbf{A}|\mathbf{B}|\mathbf{B}, \mathbf{A}|\mathbf{B}|\mathbf{A}|\mathbf{B}, \mathbf{A}|\mathbf{B}|\mathbf{B}|\mathbf{A}, \mathbf{B}|\mathbf{A}|\mathbf{B}|\mathbf{A}, \mathbf{B}|\mathbf{A}|\mathbf{A}|\mathbf{B}, \mathbf{B}|\mathbf{B}|\mathbf{A}|\mathbf{A}\}.$$

2.5 SUMMARY

In this chapter, we have presented the elementary operations for the symmetric group as well as different ways for notating permutations. One of the main themes that we emphasize in this chapter is that permutations can often be decomposed into simpler pieces. For example, we discussed how one can factor any permutation into a product of adjacent transpositions. In later chapters, we will extend this theme of decomposability not just for permutation but also to *distributions over permutations*, which is the focus of our next chapter.

3

PROBABILITY DISTRIBUTIONS ON THE SYMMETRIC GROUP

REASONING with probability distributions over the symmetric group is the central theme of this dissertation. In this chapter, we introduce many of the basic probabilistic concepts and operations related to permutations. Many of the computations in this chapter scale factorially in the number of items being ranked, or objects being tracked, necessitating approximate approaches. These computational challenges will form the main body of our work in later chapters, but we begin by introducing exact probabilistic operations, while ignoring complexity.

3.1 DISTRIBUTIONS ON THE SYMMETRIC GROUP

A distribution $h(\sigma)$, defined over the group of permutations can be viewed as a joint distribution over the n variables $(\sigma(1), \ldots, \sigma(n))$ (where $\sigma(j) \in \{1, \ldots, n\}$), subject to *mutual exclusivity constraints* which stipulate that two objects cannot simultaneously map to the same rank, or alternatively, that two ranks cannot simultaneously be occupied by the same object. Thus, we have that $h(\sigma(i) = \sigma(j)) = 0$ whenever $i \neq j$. It is due to this fact that all of the $\sigma(i)$ are coupled in the joint distribution, that typical approaches from the machine learning community such as graphical models, which might have otherwise exploited an underlying conditional independence structure, are ineffective.

We will commonly refer to the *uniform* and *delta* distributions over permutations, defined as:

$$U(\sigma) = \frac{1}{n!}, \text{ for all } \sigma \in S_n, \text{ and } \delta_{\pi_0}(\sigma) = \begin{cases} 1 & \text{if } \sigma = \pi_0 \\ 0 & \text{otherwise} \end{cases}. \tag{3.1}$$

The fact that there are factorially many possible rankings poses a number of significant challenges for learning and inference. First, there is no way to tractably represent arbitrary distributions over rankings for large n. Storing an array of 12! doubles, for example, requires roughly 14 gigabytes of storage, which is beyond the capacity of a typical modern PC. Second, the naive algorithmic complexity of common probabilistic operations is also intractable for such distributions. As we discuss later, computing the marginal probability, $h(\sigma(i) < \sigma(j))$, that item i is preferred to item j, for example, requires a summation over $O((n-2))!)$ elements. Finally, even if storage and computation issues were resolved, one would still have sample complexity issues to contend with — for nontrivial n, it is impractical to hope, for example, that each of the $n!$ possible rankings would appear even once in a training set of rankings. The only existing datasets in which every possible ranking is realized are those for which $n \leqslant 5$, and in fact, the APA

dataset (discussed below in Section 3.4) is the only such dataset for $n = 5$ that we are aware of.

3.2 PROBABILISTIC REASONING WITH PERMUTATIONS

Ignoring tractability for now, we introduce in this section a number of common and useful probabilistic manipulations that arise in permutation problems. As a prelude, we set up a simple probabilistic inference problem that commonly arises in multitarget tracking called the *identity management problem*.

3.2.1 *The identity management problem*

In this problem, we observe a stream of localization data from three people walking inside a room (illustrated in Figure 7). Except for a camera positioned at the entrance, however, there is no way to distinguish between identities once they are inside. In this example, an internal tracker declares that two tracks have 'mixed' whenever they get too close to each other and announces the identity of any track that enters or exits the room.

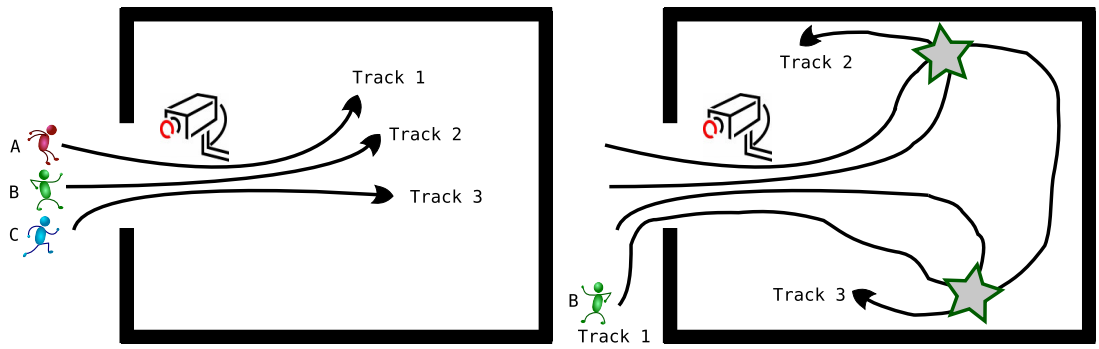
In our particular example, three people, Alice, Bob and Cathy, enter a room separately, walk around, and we observe Bob on Track 1 as he exits the room. The events for our particular example in the figure are recorded in Table 1. The *inference problem* that we must solve is: *after the third event, who is at Tracks 1 and 2*? In fact, in our toy example, there is a simple solution. Since Tracks 2 and 3 never mix, we know that Cathy cannot be in Track 2 in the end, and furthermore, since we observe Bob to be in Track 1 when he exits, we can deduce that Cathy must have been in Track 3, and therefore Alice must have been in Track 2. Our simple example illustrates the combinatorial nature of the problem — in particular, reasoning about the mixing events allows us to exactly decide where Alice and Cathy were even though we only made an observation about Bob at the end.

Event #	*Event Type*
1	*Tracks 1 and 2 mixed*
2	*Tracks 1 and 3 mixed*
3	*Observed Identity Bob at Track 1*

Table 1: Table of Mixing and Observation events logged by the tracker.

3.2.2 *Hidden Markov models on the symmetric group*

In identity management, a permutation σ represents a joint assignment of identities to internal tracks, with $\sigma(i)$ being the track belonging to the ith identity. When people walk too closely together, their identities can be confused, leading to uncertainty over σ. To model this uncertainty,



(a) Before (b) After

Figure 7: Identity Management example. Three people, Alice, Bob and Charlie enter a room and we receive a position measurement for each person at each time step. With no way to observe identities inside the room, however, we are confused whenever two tracks get too close. In this example, Track 1 crosses with Track 2, then with Track 3, then leaves the room, at which point it is observed that the identity at Track 1 is in fact Bob.

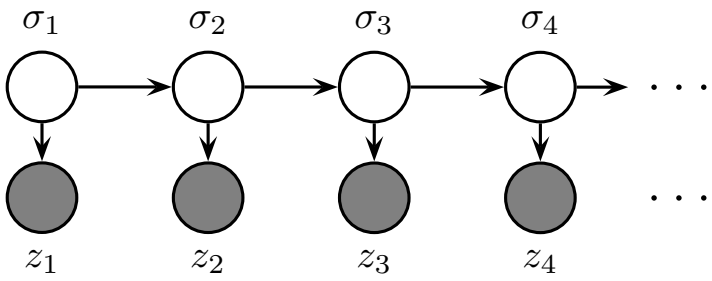


Figure 8: Hidden Markov model over permutations.

we use a *Hidden Markov Model (HMM)* on permutations, which is a joint distribution over latent permutations $\sigma^{(1)}, \ldots, \sigma^{(T)}$, and observed variables $z^{(1)}, \ldots, z^{(T)}$ which factors as:

$$h(\sigma^{(1)}, \ldots, \sigma^{(T)}, z^{(1)}, \ldots, z^{(T)}) = h(\sigma^{(1)}) h(z^{(1)} | \sigma^{(1)}) \prod_{t=2}^{T} h(z^{t} | \sigma^{(t)}) \cdot h(\sigma^{(t)} | \sigma^{(t-1)}).$$

Figure 8 is an independence diagram illustrating the conditional independence relationships implied by the factorization. The conditional probability distribution $h(\sigma^{(t)} | \sigma^{(t-1)})$ is called the *transition model*, and might reflect, for example, that the identities belonging to two tracks were swapped with some probability by a mixing event. The distribution $P(h^{(t)} | \sigma^{(t)})$ is called the *observation model*, which might, for example, capture a distribution over the color of clothing for each individual.

We will be particularly concerned with the *filtering problem*, in which one queries the HMM for the posterior distribution at some time step, condi-

tioned on all past observations. Given the distribution $h(\sigma^{(t)}|z^{(1)},\dots,z^{(t)})$, we recursively compute the posterior at time $t+1$, $h(\sigma^{(t+1)}|z^{(1)},\dots,z^{(t+1)})$, in two steps: a *prediction/rollup* step and a *conditioning* step. Taken together, these two steps form the well known *Forward Algorithm* [111]. The prediction/rollup step multiplies the distribution by the transition model and marginalizes out the previous time step:

$$h(\sigma^{(t+1)}|z^{(1)},\dots,z^{(t)}) = \sum_{\sigma^{(t)}} h(\sigma^{(t+1)}|\sigma^{(t)})h(\sigma^{(t)}|z^{(1)},\dots,z^{(t)}). \quad (3.2)$$

The conditioning step conditions the distribution on an observation $z^{(t+1)}$ using Bayes rule,

$$h(\sigma^{(t+1)}|z^{(1)},\dots,z^{(t+1)}) \propto h(z^{(t+1)}|\sigma^{(t+1)})h(\sigma^{(t+1)}|z^{(1)},\dots,z^{(t)}). \quad (3.3)$$

These probabilistic inference operations which we have introduced in the context of filtering for HMMs in fact appear ubiquitously in more general probabilistic inference settings. Below we introduce the probabilistic operations that will be considered in this thesis.

3.2.3 *Probabilistic operations*

Our first two operations are the prediction/rollup and conditioning operations from above which are applicable in settings beyond hidden Markov models.

Prediction/Rollup operation:

$$h(\sigma^{(t+1)}) = \sum_{\sigma^{(t)}} h(\sigma^{(t+1)}|\sigma^{(t)})h(\sigma^{(t)}). \quad (3.4)$$

Since there are $n!$ permutations, a single iteration of the algorithm requires $O((n!)^2)$ flops and is consequently intractable for all but very small n. Bayesian conditioning, on the other hand, which requires a pointwise multiplication of a likelihood function and a prior distribution, has a running time of $O(n!)$ in general:

Bayesian conditioning operation:

$$h(\sigma|z) \propto h(z|\sigma)h(\sigma). \quad (3.5)$$

Additionally, it is often the case that one desires a *normalized* posterior distribution which requires summing over $n!$ terms:

Normalization:

$$Z = \sum_{\sigma \in S_n} h(\sigma). \quad (3.6)$$

In other cases, one simply wants to *maximize* the posterior, finding the ranking or identity-to-track permutation which attains the highest likelihood under the posterior distribution, which also has $O(n!)$ running time in general:

Maximization:

$$\hat{\sigma} = \arg\max_{\sigma \in S_n} h(\sigma). \tag{3.7}$$

Below, we present several more operations which require some more explanation: *convolution*, *shifting*, *restriction*, and *embedding*.

CONVOLUTION. Instead of focusing on prediction/rollup operations (Equation 3.4) in full generality, we will typically consider one particularly useful class of transition models — that of random walks over a group, which assumes that $\sigma^{(t+1)}$ is generated from $\sigma^{(t)}$ by drawing a random permutation $\pi^{(t)}$ from some distribution $q^{(t)}$ and setting $\sigma^{(t+1)} = \pi^{(t)}\sigma^{(t)}$.[1] In our identity management example, $\pi^{(t)}$ represents a random identity permutation that might occur among tracks when they get close to each other (what we call a *mixing event*). For example, $q((1,2)) = 1/2$ means that Tracks 1 and 2 swapped identities with probability $1/2$. The random walk model also appears in many other applications such as modeling card shuffles [29], which we will consider again in Chapter 13.

The motivation behind the random walk transition model is that it allows us to write the prediction/rollup operation as a *convolution* of distributions on a group. The extension of the familiar notion of convolution to groups simply replaces additions and subtractions by analogous group operations (function composition and inverse, respectively):

Definition 17. Let f and g be probability distributions on a group G. Define the *convolution* of f and g to be the function:

Convolution:

$$[f * g](\sigma_1) = \sum_{\sigma_2} f(\sigma_1 \sigma_2^{-1}) g(\sigma_2). \tag{3.8}$$

Compare Definition 17 with the more familiar definition of convolution of two functions $\phi_1, \phi_2 : \mathbb{R} \to \mathbb{R}$, defined as: $\phi_1 * \phi_2(t) = \int_{-\infty}^{\infty} \phi_1(t-\tau)\phi_2(\tau)$. Note that this definition of convolution on groups is *strictly* a generalization of convolution of functions on the real line, and is a non-commutative operation for non-Abelian groups. Thus the distribution $f * g$ is not necessarily the same as $g * f$.

Using Definition 17, we see that the prediction/rollup step can be rewritten as:

$$\begin{aligned} h(\sigma^{(t+1)}) &= \sum_{\sigma^{(t)}} h(\sigma^{(t+1)}|\sigma^{(t)}) \cdot h(\sigma^{(t)}), \\ &= \sum_{\{(\sigma^{(t)}, \pi^{(t)}) \,:\, \sigma^{(t+1)} = \pi^{(t)} \cdot \sigma^{(t)}\}} q^{(t)}(\pi^{(t)}) \cdot h(\sigma^{(t)}), \end{aligned}$$

1 In the context of identity management, we place π on the left side of the multiplication because we want it to permute tracks and not identities. Had we defined π to map from tracks to identities (instead of identities to tracks), then π would be multiplied from the right. Besides left versus right multiplication, there are no differences between the two conventions.

$$
\begin{aligned}
&\text{(Right-multiplying both sides of}\\
&\qquad \sigma^{(t+1)} = \pi^{(t)}\sigma^{(t)} \text{ by } (\sigma^{(t)})^{-1}\text{, we see that}\\
&\qquad \pi^{(t)} \text{ can be replaced by } \sigma^{(t+1)}(\sigma^{(t)})^{-1}\text{),}\\
&= \sum_{\sigma^{(t)}} q^{(t)}(\sigma^{(t+1)} \cdot (\sigma^{(t)})^{-1}) \cdot h(\sigma^{(t)}),\\
&= \left[q^{(t)} * h\right](\sigma^{(t+1)}).
\end{aligned}
$$

THE SHIFT OPERATION FOR DISTRIBUTIONS. Just as in linear algebra, where it is often more convenient to work with respect to a particular basis set, it is often more convenient to work with a particular labeling of the input or output set in many permutation applications. Consider a random permutation σ_1 with distribution h, representing a 1-1 mapping from an input set $X_{in} = \{1, \ldots, n\}$ to an output set $X_{out} = \{1, \ldots, n\}$ (here we abuse notation by using the word 'permutation' to refer to a mapping that is not strictly from a set to itself). We would like to know (1) what σ_1 should be if the reference ordering of the input and output spaces, X_{in} and X_{out}, were relabeled, and (2), how to modify the distribution h accordingly.

Suppose now that the input and output spaces, X_{in} and X_{out}, are respectively relabeled as $\pi_{in} : X_{in} \to X_{in}$ and $\pi_{out} : X_{out} \to X_{out}$. Rewriting σ_1 with respect to the new labeling is the same as changing the basis for a linear operator:

$$\sigma_2 = \pi_{out}\sigma_1\pi_{in}^{-1}. \tag{3.9}$$

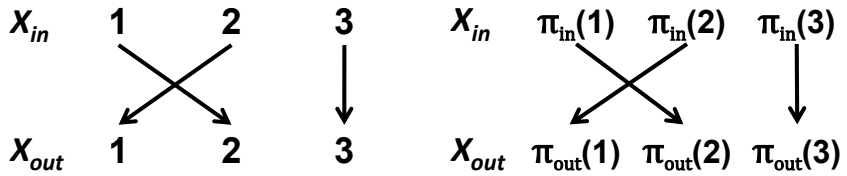
If $X_{in} = X_{out}$ and their labelings are the same (i.e., $\pi_{in} = \pi_{out}$), then we have $\sigma_2 = \pi_{in}\sigma_1\pi_{in}^{-1}$. Equation 3.9 can be understood if we try to see where σ_2 maps an item in X_{in}, labeled as i with respect to the relabeled set. With respect to the original labeling of X_{in}, item i becomes $\pi_{in}^{-1}(i)$, which is then mapped under σ_1 with respect to the original labelings of X_{in} and X_{out} to $\sigma_1(\pi_{in}^{-1}(i))$. Finally, with respect to the relabeling of X_{out}, this result is $\pi_{out}(\sigma_1(\pi_{in}^{-1}(i)))$. Notationally, we will write: $\sigma_2 = \textsc{Shift}[\sigma_1, \pi_{in}, \pi_{out}]$.

Example 18. *As an example, let* $\sigma_1 = [2\,1\,3]$ *and let* $\pi_{in} = [2\,3\,1]$, $\pi_{out} = [3\,2\,1]$. *A diagram is shown in Figure 9. In (a), we show* σ_1 *represented according to the original labeling. In (b), we show the same permutation where we've relabeled* X_{in} *using* π_{in} *and* X_{out} *using* π_{out}. *Writing (b) in one-line notation, we see that* $\sigma_2(1) = 1$, $\sigma_2(2) = 2$, *and* $\sigma_2(3) = 3$, *and hence that* σ_2 *should be* $[1\,2\,3]$. *We can also try computing* σ_2 *using Equation 3.9. Note that* $\sigma_{in}^{-1} = [3\,1\,2]$. *Then:*

$$\sigma_2 = \pi_{out} \cdot \sigma_1 \cdot \pi_{in}^{-1} = [3\,2\,1] \cdot [2\,1\,3] \cdot [3\,1\,2] = [1\,2\,3],$$

which matches the computation from above..

Having now defined shifting for permutations, we would like to define a corresponding shift for a distribution over permutations. If σ_1 is distributed according to a distribution $h(\sigma_1)$ we would like to have a



(a) Original labels, $\sigma_1 = [2\,1\,3]$ (b) New labels, $\sigma_2 = [1\,2\,3]$

Figure 9: Diagram for Example 18

function $h' = \textsc{Shift}[h, \pi_{in}, \pi_{out}]$ which, with respect to the relabeled input and output spaces, is equal to f in the original space. Thus, we want $h'(\pi_{out} \cdot \sigma_1 \cdot \pi_{in}^{-1}) = h(\sigma_1)$. Rewriting, we have the following *shift operation* for distributions over permutations:

Shift:

$$h'(\sigma) = \textsc{Shift}[h, \pi_{in}, \pi_{out}] = h(\pi_{out}^{-1} \cdot \sigma \cdot \pi_{in}). \tag{3.10}$$

RESTRICT/EMBED OPERATIONS. Finally, it is sometimes necessary in certain applications to increase the size of the underlying item set. For example, in a multitarget tracking problem with n identities, a new person may walk into the building, requiring us to extend our current distribution over S_n to a distribution over S_{n+1}.

In the simplest case, where we are given the distribution h (defined over S_n) and know the identity-to-track association of this new $(n+1)^{th}$ person, the distribution after adding the new person is defined on S_{n+1} and we will denote it as $\textsc{Embed}[h]$:

Embed:

$$\textsc{Embed}[h](\sigma) = \begin{cases} h(\sigma) & \text{if } \sigma(n+1) = n+1 \\ 0 & \text{otherwise} \end{cases}, \text{ for all } \sigma \in S_{n+1}. \tag{3.11}$$

Similarly, we have a corresponding $\textsc{Restrict}$ functional which takes a function or distribution defined over S_{n+1} and returns a function on S_n:

Restrict:

$$\textsc{Restrict}[h](\sigma) = h([\sigma(1)\,\sigma(2)\,\ldots\,\sigma(n)\ \underbrace{n+1}_{\text{fixed}}]), \text{ for all } \sigma \in S_n. \tag{3.12}$$

We will use the notation $\textsc{Restrict}^k[h]$ and $\textsc{Embed}^k[h']$ to denote iterated operations. For example, $\textsc{Restrict}^2[h]$ is the same as $\textsc{Restrict}[\textsc{Restrict}[h]]$ and returns the following function on S_{n-2}:

$$h'(\sigma') = h([\sigma'(1), \ldots, \sigma'(n-2), \underbrace{n-1, n}_{\text{fixed}}]).$$

To summarize, we have introduced a number of useful operations that occur frequently in probabilistic reasoning scenarios. All of the following are computationally intractable for all by very small n:

Convolution:	$\mathcal{O}((n!)^2)$	**Shift:**	$\mathcal{O}(n \cdot n!)$
Conditioning:	$\mathcal{O}(n!)$	**Embed:**	$\mathcal{O}((n+1)!)$
Normalization:	$\mathcal{O}(n!)$	**Restrict:**	$\mathcal{O}(n!)$
Maximization:	$\mathcal{O}(n!)$		

In later chapters, however, we will show that when a distribution over permutations can be decomposed additively or multiplicatively, there often exist efficient exact or approximate inference routines.

3.3 COMPACT SUMMARY STATISTICS OF DISTRIBUTIONS

Since there are $n!$ permutations, it is infeasible for all but very small n to consider storing full distributions over S_n, much less to even consider the probabilistic inference operations mentioned in the previous section.

In this section, we consider a few ideas for compactly summarizing distributions over permutations. These compact summaries are useful for visualizing and communicating the main features of a probability distribution using broad strokes. Perhaps surprisingly however, some of our deceptively simple summary statistics will go on to form the basis for the powerful machinery of Fourier analysis which we will leverage in later chapters.

MODE. In the following, we assume that a distribution $h(\sigma)$ (which could simply be an empirical histogram) is provided. While continuous distributions like Gaussians are typically summarized using moments (like mean and variance), or more generally, expected features, it is not immediately obvious how one might, for example, compute the 'mean' of a distribution over permutations. The simplest statistic that one might report instead, is the *mode*, the permutation with the highest probability:

$$\textbf{Mode}: \qquad \arg\max_{\sigma \in S_n} h(\sigma).$$

CONSENSUS RANKING. Another common statistic to report, particularly among ranking applications, is what is known as the *consensus ranking* [97], the ranking which minimizes the expected distance to a ranking drawn from h. More precisely, let $d_K : S_n \times S_n \to \mathbb{R}$ be the *Kendall's tau distance metric* (see Section 2.3.2), The consensus ranking is defined as:

$$\textbf{Consensus ranking}: \qquad \arg\min_{\sigma \in S_n} \sum_{\pi \in S_n} d_K(\pi, \sigma) h(\pi).$$

FIRST-ORDER SUMMARIES. Yet another common summary that we might use as a replacement for the 'mean' is to think of the permutations as *permutation matrices* and to average the matrices.

Example 19. *For example, consider the two permutations* $\epsilon, (1,2) \in S_3$ *(*ϵ *is the identity and* $(1,2)$ *swaps 1 and 2). We can associate the identity permutation* ϵ *with the* 3×3 *identity matrix, and similarly, we can associate the permutation* $(1,2)$ *with the matrix:*

$$(1,2) \mapsto \begin{bmatrix} 0 & 1 & 0 \\ 1 & 0 & 0 \\ 0 & 0 & 1 \end{bmatrix}.$$

The 'average' of ϵ *and* $(1,2)$ *is therefore:*

$$\frac{1}{2}\begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix} + \frac{1}{2}\begin{bmatrix} 0 & 1 & 0 \\ 1 & 0 & 0 \\ 0 & 0 & 1 \end{bmatrix} = \begin{bmatrix} 1/2 & 1/2 & 0 \\ 1/2 & 1/2 & 0 \\ 0 & 0 & 1 \end{bmatrix}.$$

As we will later discuss in more detail, computing the 'mean' (as described above) of a distribution over permutations, h, compactly summarizes h by storing a marginal distribution over each of $\sigma(1), \sigma(2), \ldots, \sigma(n)$, which requires storing only $O(n^2)$ numbers rather than the full $O(n!)$ for the exact distribution. Thus we can write the first-order summary as:

$$\textbf{First-order summary}: H_{ij} = \sum_{\sigma\,:\,\sigma(j)=i} h(\sigma), \text{ for all } 1 \leqslant i, j \leqslant n.$$

As an identity management based example, one possible summary might look like:

$$H = \left[\begin{array}{l|ccc} & \text{Alice} & \text{Bob} & \text{Cathy} \\ \hline \text{Track 1} & 2/3 & 1/6 & 1/6 \\ \text{Track 2} & 1/3 & 1/3 & 1/3 \\ \text{Track 3} & 0 & 1/2 & 1/2 \end{array}\right],$$

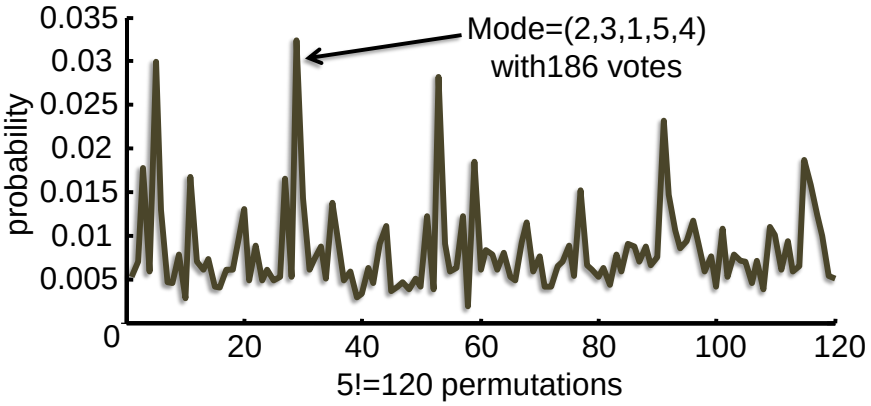
and says, for example, that

$$h(\text{Alice is at Track 1}) = 2/3, \text{ and } h(\text{Bob is at Track 3}) = 1/2.$$

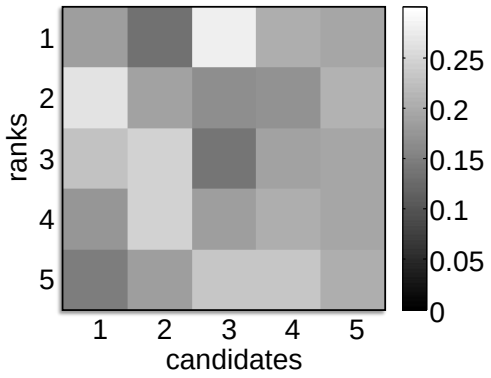
Such doubly stochastic "first-order summaries" have been studied in various settings [120, 49].

3.4 APA ELECTION DATA

As a more elaborate running example throughout the thesis, we will analyze the well known APA election dataset that was first used by [29] and has since been analyzed in a number of ranking studies. The APA dataset is a collection of 5738 ballots from a 1980 presidential election of the American Psychological Association where members rank ordered five candidates from favorite to least favorite. The names of the five candidates that year



(a)



(b)

Figure 10: APA (American Psychological Association) election data. (a) vote distribution: percentage of votes for each of $5! = 120$ possible rankings — the mode of the distribution is $\sigma = (2,3,1,5,4)$. (b) Matrix of first order marginals: the $(i,j)^{th}$ entry reflects the number of voters who ranked candidate j in the i^{th} rank.

were (1) William Bevan, (2) Ira Iscoe, (3) Charles Kiesler, (4) Max Siegle, and (5) Logan Wright [92].

Since there are five candidates, there are $5! = 120$ possible rankings. In Figure 10a we plot the proportion of votes that each ranking received. Interestingly, instead of concentrating at just a small set of rankings, the vote distribution in the APA dataset is fairly diffuse with every ranking receiving some number of votes. The mode of the vote distribution occurs at the ranking $\sigma = (2,3,1,5,4) = [\![\text{C. Kiesler, W. Bevan, I. Iscoe, L. Wright, M. Siegle}]\!]$ with 186 votes.

For interpretability, we also visualize the *first-order matrix* in which the (i,j) entry represents the number of voters who assigned rank i to candidate j.

$$\left[\sum_{\sigma:\sigma(j)=i} h(\sigma)\right]_{i,j} =$$

	W. Bevan	I. Iscoe	C. Kiesler	M. Siegle	L. Wright
Rank 1	1053	775	1609	1172	1129
Rank 2	1519	1077	960	972	1210
Rank 3	1313	1415	793	1089	1128
Rank 4	1002	1416	1050	1164	1106
Rank 5	851	1055	1326	1341	1165

.

Figure 10b also represents the first-order matrix using grayscale levels to represent numbers of voters. What can be seen is that overall, candidate 3 (C. Kiesler) received the highest number of votes for rank 1 (and incidentally, won the election). The vote distribution gives us a story that goes far deeper than simply telling us who the winner was, however. [29], for example, noticed that candidate 3 also had a significant "hate" vote — a good number of voters placed him in the last rank. Throughout this thesis, we will let this story unfold via a series of examples based on the APA dataset.

3.5 CONCLUSION

In this chapter we have introduced the basics of probabilistic reasoning in the context of permutations as well as some of the motivating examples from tracking and elections that we will use throughout the thesis. We introduced a number of common inference operations such as convolutions, conditioning, maximization and normalization, as well as several simple summary statistics for permutation data.

Due to the factorial size of the symmetric group, one must work with compact probabilistic representations. At the same time, one must design inference algorithms that can harness the structure of these compact representations for efficiency. Solving these twin challenges of *compact representation* and *efficient inference* form the central themes of Parts II and III of this thesis, where we propose novel methodologies for decomposing large intractable distributions over permutations into collections of smaller, easier-to-manipulate component functions.

Part II

ADDITIVE DECOMPOSITIONS: FOURIER BASED REPRESENTATIONS AND INFERENCE

4

OVERVIEW OF PART II: ADDITIVE DECOMPOSITIONS

We have introduced the symmetric group and the fundamental algorithms associated with manipulating permutations, as well as the two main problems associated with probabilistic reasoning with permutations — finding *compact representations* and *efficient inference algorithms*. In Part II of this thesis, we approach these problems of efficient representation and inference by decomposing probability distributions over permutations into a weighted sums of simpler Fourier basis functions.

Recall that the ordinary Fourier series decomposition allows one to write any function $h : [0, 1] \to \mathbb{C}$ as a linear combination of trigonometric basis functions [93]:

$$h(x) = \sum_{m=-\infty}^{\infty} \alpha_m e^{i2\pi mx},$$

where $\alpha_m \in \mathbb{C}$ for each m. The squared magnitude of each α_m ($|\alpha_m|^2$) measures the "energy" of h contained at frequency m — for example, if $h(x)$ is purely sinusoidal with one frequency, then only one of the α_m is nonzero. The collection of trigonometric functions $e^{i2\pi m}$ forms a *complete orthogonal basis* for the space of functions on the interval $[0, 1]$; thus any collection of α_m's corresponds to some function h and moreover, h is uniquely determined. Since the collection of frequency responses, $\{\alpha_m\}_{m=-\infty}^{\infty}$ can themselves be thought of as a function in "frequency space", we call the mapping between the functions $(h \mapsto \{\alpha_m\})$ the *Fourier transform*.

Fourier analytic methods are now widely employed in almost all science and engineering disciplines due in part to the development of efficient algorithms for computing Fourier transforms such as the FFT [23]. One of the classical applications of Fourier methods, for example, has been for "compressing" signals by dropping high frequency terms, typically resulting in a smoother approximation to the original signal:

$$h(x) \approx \sum_{m=-B}^{B} \alpha_m e^{i2\pi mx},$$

in an operation known as *bandlimiting*.

In Part II, we rely on a generalized notion of the Fourier transform as well as generalized notions of frequency and bandlimiting that apply to functions defined over the symmetric group. We show in particular that by exploiting the Fourier structure of probabilistic reasoning problems, one can obtain principled and compact approximate representations of a distribution over permutations. Important questions that we answer in Part II include:

- What does it mean for a function defined on permutations to be low or high frequency?

- In what sense does the Fourier transform generalize the ordinary DFT (discrete Fourier transform)?
- How can probabilistic inference operations be performed using Fourier coefficients, without working explicitly with the approximate probability vector?
- What kind of errors can arise during probabilistic inference by using a bandlimited approximation?

The following is an outline of the main contributions as well as a roadmap of the chapters ahead in Part II.

- In Chapter 5, we provide a gentle introduction to the theory of group representations and noncommutative Fourier analyis. While the results of Chapters 5 and 6 are not novel, and have indeed been studied by mathematicians for decades [30, 127, 136, 18], noncommutative Fourier analysis is still fairly new to the machine learning and artificial intelligence communities, which have just begun to discover some of its exciting applications [57, 80, 79].
- While Chapter 5 stresses probabilistic connections and intuitions, Chapter 6 is a is concerned with the implementation of basic combinatorial algorithms, such as the generalized FFT (Fast Fourier transform), that are necessary as subroutines for the methods of Part II. Readers who are more interested in probabilistic modeling and inference can continue to the next chapter with little loss of continuity.

  Together, Chapters 5 and 6 form tutorial chapters which are targeted specifically at the machine learning and artificial intelligence communities and describe the connections between noncommutative Fourier analysis and probabilistic inference problems that involve permutations.
- In Chapter 7, we present a collection of useful probabilistic models for which we *can* efficiently compute low-order Fourier coefficients or even provide a closed-form expression. Viewing these models as a collection of atoms, we show that they can be combined via scale/shift/convolution operations to form a more complex and rich set of models.
- In Chapter 8, we discuss performing probabilistic inference operations in the Fourier domain. In particular, we present Fourier theoretic algorithms for the probabilistic inference operations which we presented in Chapter 3, which appear in filtering applications and beyond, such as prediction/rollup, conditioning with Bayes rule, normalization, and maximization. Our most important contribution in this chapter is a novel and conceptually simple algorithm, called *Kronecker Conditioning*, which performs all Bayesian conditioning operations completely in the Fourier domain, allowing for a principled tradeoff between computational complexity and approximation accuracy. Our

approach generalizes upon previous work in two ways — first, in the sense that it can address any transition model or likelihood function that can be represented in the Fourier domain, and second, in the sense that many of our results hold for arbitrary finite groups.

- Finally, in Chapter 9, we analyze the errors which can be introduced by bandlimiting a probability distribution and show how they propagate with respect to inference operations. We argue that approximate conditioning based on bandlimited distributions can sometimes yield Fourier coefficients which do not correspond to any valid distribution, even returning negative "probabilities" on occasion. We address possible negative and inconsistent probabilities by presenting a method for projecting the result back into the polytope of coefficients which correspond to nonnegative and consistent marginal probabilities using a simple quadratic program. We empirically evaluate the accuracy of approximate inference on simulated data drawn from our model and further demonstrate the effectiveness of our approach on a real camera-based multi-person tracking scenario.

The contributions of Part II have also appeared (or will soon appear) in publication in the following articles:

[1] Jonathan Huang, Carlos Guestrin, and Leonidas Guibas. Efficient inference for distributions on permutations. In J.C. Platt, D. Koller, Y. Singer, and S. Roweis, editors, *Advances in Neural Information Processing Systems 20*, NIPS '07, pages 697–704. MIT Press, Cambridge, MA, 2008.

[2] Jonathan Huang, Carlos Guestrin, and Leonidas J. Guibas. Fourier theoretic probabilistic inference over permutations. *Journal of Machine Learning Research (JMLR)*, 10:997–1070, 2009.

[3] Xiaoye Jiang, Jonathan Huang, and Leonidas Guibas. Fourier-information duality in the identity management problem. In *The European Conference on Machine Learning and Principles and Practice of Knowledge Discovery in Databases (ECML 2011)*, ECML '11, Athens, Greece, September 2011.

5

FOURIER ANALYSIS ON THE SYMMETRIC GROUP

OVER the last fifty years, the Fourier Transform has been ubiquitously applied to everything digital, particularly with the invention of the Fast Fourier Transform [23, 113]. On the real line, the Fourier Transform is a well-studied method for decomposing a function into a sum of sine and cosine terms over a spectrum of frequencies. Perhaps less familiar to the machine learning community though, is its group theoretic generalization. In this section we review group theoretic generalizations of the Fourier transform with an eye towards approximating functions on S_n. Noncommutative generalizations of the Fourier transform have been studied quite extensively throughout the last century from both the mathematics [83] and physics communities [18]. Applications to permutations were first pioneered by Persi Diaconis who studied problems in card shuffling and since then, there have been many papers on related topics in probability and statistics. For further information, see [29] and [127].

5.1 GROUP REPRESENTATION THEORY

The generalized definition of the Fourier Transform relies on the theory of group representations,[1] which formalize the concept of associating permutations with matrices and are used to construct a complete basis for the space of functions on a group G, thus also playing a role analogous to that of sinusoids on the real line.

Definition 20 (Group Representation)**.** A *representation* of a group G is a map ρ from G to a set of invertible $d_\rho \times d_\rho$ (complex) matrix operators ($\rho : G \to \mathbb{C}^{d_\rho \times d_\rho}$) which preserves algebraic structure in the sense that for all $\sigma_1, \sigma_2 \in G$, we have:

$$\rho(\sigma_1 \sigma_2) = \rho(\sigma_1) \cdot \rho(\sigma_2).$$

The matrices which lie in the image of ρ are called the *representation matrices*, and we will refer to d_ρ as the *degree* of the representation.

1 The term 'group representation', which is standard throughout all of modern algebraic literature, should not be confused with our problem of efficient 'representation' in which we design tractable methods for storing the parameters or summary statistics of a probability distribution.

The requirement that $\rho(\sigma_1\sigma_2) = \rho(\sigma_1)\cdot\rho(\sigma_2)$ is analogous to the property that $e^{i(\theta_1+\theta_2)} = e^{i\theta_1}\cdot e^{i\theta_2}$ for the conventional sinusoidal basis. Each matrix entry, $\rho_{ij}(\sigma)$ defines some function over S_n:

$$\rho(\sigma) = \begin{bmatrix} \rho_{11}(\sigma) & \rho_{12}(\sigma) & \cdots & \rho_{1d_\rho}(\sigma) \\ \rho_{21}(\sigma) & \rho_{22}(\sigma) & \cdots & \rho_{2d_\rho}(\sigma) \\ \vdots & \vdots & \ddots & \vdots \\ \rho_{d_\rho 1}(\sigma) & \rho_{d_\rho 2}(\sigma) & \cdots & \rho_{d_\rho d_\rho}(\sigma) \end{bmatrix}, \tag{5.1}$$

and consequently, each representation ρ simultaneously defines a set of d_ρ^2 functions over S_n. We will eventually think of group representations as the set of Fourier basis functions onto which we can project arbitrary functions.

Before moving onto examples, we make several remarks about the generality of these chapters. First, while this thesis is primarily focused on the symmetric group, many of the results from Part II hold for arbitrary finite groups. For example, there are a variety of finite groups that have been studied in applications, like metacyclic and dihedral groups [136], wreath product groups [37], etc. However, while some of these results will even extend with minimal effort to more general cases, such as locally compact groups, the assumption in all of the following results will be that the group G is finite, even if it is not explicity stated. Secondly, given an arbitrary finite group G, some of the algebraic results that we use require that the underlying field be the complex numbers. For the particular case of the symmetric group, however, we can in fact assume that the representations are real-valued matrices. Thus, throughout the thesis, we will explicitly assume that the representations are real-valued.[2]

Example 21. *We begin by showing three examples of representations on the symmetric group.*

1. *The simplest example of a representation is called the* trivial representation $\rho_{(n)} : S_n \to \mathbb{R}^{1\times 1}$*, which maps each element of the symmetric group to 1, the multiplicative identity on the real numbers. The trivial representation is actually defined for every group, and while it may seem unworthy of mention, it plays the role of the constant basis function in the Fourier theory.*

2. *The* first-order permutation representation *of S_n, which we alluded to in Example 19, is the degree n representation, $\tau_{(n-1,1)}$ (we explain the terminology in Section 5.2) , which maps a permutation σ to its corresponding permutation matrix given by $[\tau_{(n-1,1)}(\sigma)]_{ij} = \mathbb{1}\{\sigma(j) = i\}$. For example, the first-order permutation representation on S_3 is given by:*

2 To recover similar results for more complex-valued representations, one would have to replace matrix transposes by adjoints, etc.

$$\tau_{(2,1)}(\epsilon) = \begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix} \qquad \tau_{(2,1)}(1,2) = \begin{bmatrix} 0 & 1 & 0 \\ 1 & 0 & 0 \\ 0 & 0 & 1 \end{bmatrix}$$

$$\tau_{(2,1)}(2,3) = \begin{bmatrix} 1 & 0 & 0 \\ 0 & 0 & 1 \\ 0 & 1 & 0 \end{bmatrix} \qquad \tau_{(2,1)}(1,3) = \begin{bmatrix} 0 & 0 & 1 \\ 0 & 1 & 0 \\ 1 & 0 & 0 \end{bmatrix}$$

$$\tau_{(2,1)}(1,2,3) = \begin{bmatrix} 0 & 0 & 1 \\ 1 & 0 & 0 \\ 0 & 1 & 0 \end{bmatrix} \qquad \tau_{(2,1)}(1,3,2) = \begin{bmatrix} 0 & 1 & 0 \\ 0 & 0 & 1 \\ 1 & 0 & 0 \end{bmatrix}$$

3. *The* alternating representation *of* S_n*, maps a permutation* σ *to the determinant of* $\tau_{(n-1,1)}(\sigma)$*, which is* $+1$ *if* σ *can be equivalently written as the composition of an even number of pairwise swaps, and* -1 *otherwise. We write the alternating representation as* $\rho_{(1,\ldots,1)}$ *with* n *1's in the subscript. For example, on* S_4*, we have:*

$$\rho_{(1,1,1,1)}((1,2,3)) = \rho_{(1,1,1,1)}((13)(12)) = +1.$$

The alternating representation, as we shall discuss further, can be interpreted as the 'highest frequency' basis function on the symmetric group, intuitively due to its high sensitivity to swaps. For example, if $\tau_{(1,\ldots,1)}(\sigma) = 1$*, then* $\tau_{(1,\ldots,1)}((12)\sigma) = -1$*. In identity management, it may be reasonable to believe that the joint probability over all* n *identity labels should only change by a little if just two objects are mislabeled due to swapping — in this case, ignoring the basis function corresponding to the alternating representation should still provide an accurate approximation to the joint distribution.*

In general, a representation corresponds to an overcomplete set of functions and therefore does not constitute a valid basis for any subspace of functions. For example, the set of nine functions on S_3 corresponding to $\tau_{(2,1)}$ span only four dimensions, because there are six normalization constraints (three on the row sums and three on the column sums), of which five are independent — and so there are five redundant dimensions. To find a valid complete basis for the space of functions on S_n, we will need to find a family of representations whose basis functions are independent, and span the entire $n!$-dimensional space of functions.

In the following two definitions, we will provide two methods for constructing a new representation from old ones such that the set of functions on S_n corresponding to the new representation is linearly *dependent* on the old representations. Somewhat surprisingly, it can be shown that dependencies which arise amongst the representations can always be recognized in a certain sense, to come from the two possible following sources [117].

Definition 22.

1. **Equivalence.** Given a representation ρ_1 and an invertible matrix C, one can define a new representation ρ_2 by "changing the basis" for ρ_1:

$$\rho_2(\sigma) \triangleq C^{-1} \cdot \rho_1(\sigma) \cdot C. \tag{5.2}$$

   We say, in this case, that ρ_1 and ρ_2 are *equivalent* as representations (written $\rho_1 \equiv \rho_2$, as opposed to $\rho_1 = \rho_2$), and the matrix C is known as the *intertwining operator*. Note that $d_{\rho_1} = d_{\rho_2}$.

   It can be checked that the functions corresponding to ρ_2 can be reconstructed from those corresponding to ρ_1. For example, if C is a permutation matrix, the matrix entries of ρ_2 are exactly the same as the matrix entries of ρ_1, only permuted.

2. **Direct Sum.** Given two representations ρ_1 and ρ_2, we can always form a new representation, which we will write as $\rho_1 \oplus \rho_2$, by defining:

$$\rho_1 \oplus \rho_2(\sigma) \triangleq \left[\begin{array}{c|c} \rho_1(\sigma) & 0 \\ \hline 0 & \rho_2(\sigma) \end{array}\right]. \tag{5.3}$$

   $\rho_1 \oplus \rho_2$ is called the *direct sum representation*. For example, the direct sum of two copies of the trivial representation is:

$$\rho_{(n)} \oplus \rho_{(n)}(\sigma) = \begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix},$$

   with four corresponding functions on S_n, each of which is clearly dependent upon the trivial representation itself.

Most representations can be seen as being equivalent to a direct sum of strictly smaller representations. Whenever a representation ρ can be decomposed as $\rho \equiv \rho_1 \oplus \rho_2$, we say that ρ is *reducible*. As an example, we now show that the first-order permutation representation is a reducible representation.

Example 23. *Instead of using the standard basis vectors $\{e_1, e_2, e_3\}$, the first-order permutation representation for S_3, $\tau_{(2,1)} : S_3 \to \mathbb{C}^{3\times 3}$, can be equivalently written with respect to a new basis $\{v_1, v_2, v_3\}$, where:*

$$\begin{aligned} v_1 &= \frac{e_1 + e_2 + e_3}{|e_1 + e_2 + e_3|}, \\ v_2 &= \frac{-e_1 + e_2}{|-e_1 + e_2|}, \\ v_3 &= \frac{-e_1 - e_2 + 2e_3}{|-e_1 - e_2 + 2e_3|}. \end{aligned}$$

To 'change the basis', we write the new basis vectors as columns in a matrix C:

$$C = \begin{bmatrix} | & | & | \\ v_1 & v_2 & v_3 \\ | & | & | \end{bmatrix} = \begin{bmatrix} \frac{1}{\sqrt{3}} & -\frac{\sqrt{2}}{2} & -\frac{1}{\sqrt{6}} \\ \frac{1}{\sqrt{3}} & \frac{\sqrt{2}}{2} & -\frac{1}{\sqrt{6}} \\ \frac{1}{\sqrt{3}} & 0 & \frac{2}{\sqrt{6}} \end{bmatrix},$$

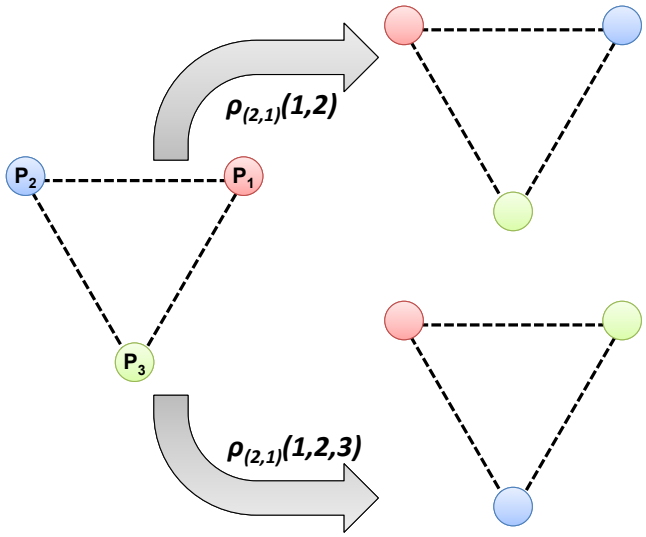


Figure 11: The action of the permutations $(1,2)$ and $(1,2,3)$ on the equilateral triangle with vertices P_1, P_2, and P_3 via the representation $\rho_{2,1}$. The transposition $(1,2)$ reflects the triangle onto itself across the y-axis and the 3-cycle $(1,2,3)$ rotates the triangle by a $\pi/3$ counter-clockwise rotation onto itself (this figure is best viewed in color)

and conjugate the representation $\tau_{(2,1)}$ by C (as in Equation 5.2) to obtain the equivalent representation $C^{-1} \cdot \tau_{(2,1)}(\sigma) \cdot C$:

$$C^{-1} \cdot \tau_{(2,1)}(\epsilon) \cdot C = \left[\begin{array}{c|cc} 1 & 0 & 0 \\ \hline 0 & 1 & 0 \\ 0 & 0 & 1 \end{array}\right],$$

$$C^{-1} \cdot \tau_{(2,1)}(1,2) \cdot C = \left[\begin{array}{c|cc} 1 & 0 & 0 \\ \hline 0 & -1 & 0 \\ 0 & 0 & 1 \end{array}\right],$$

$$C^{-1} \cdot \tau_{(2,1)}(2,3) \cdot C = \left[\begin{array}{c|cc} 1 & 0 & 0 \\ \hline 0 & 1/2 & \sqrt{3}/2 \\ 0 & \sqrt{3}/2 & -1/2 \end{array}\right],$$

$$C^{-1} \cdot \tau_{(2,1)}(1,3) \cdot C = \left[\begin{array}{c|cc} 1 & 0 & 0 \\ \hline 0 & 1/2 & -\sqrt{3}/2 \\ 0 & -\sqrt{3}/2 & -1/2 \end{array}\right],$$

$$C^{-1} \cdot \tau_{(2,1)}(1,2,3) \cdot C = \left[\begin{array}{c|cc} 1 & 0 & 0 \\ \hline 0 & -1/2 & -\sqrt{3}/2 \\ 0 & \sqrt{3}/2 & -1/2 \end{array}\right],$$

$$C^{-1} \cdot \tau_{(2,1)}(1,3,2) \cdot C = \left[\begin{array}{c|cc} 1 & 0 & 0 \\ \hline 0 & -1/2 & \sqrt{3}/2 \\ 0 & -\sqrt{3}/2 & -1/2 \end{array}\right].$$

The interesting property of this particular basis is that the new representation matrices all appear to be the direct sum of two smaller representations, a trivial representation, $\rho_{(3)}$ as the top left block, and a degree 2 representation in the bottom right which we will refer to as $\rho_{(2,1)}$.

Geometrically, the representation $\rho_{(2,1)}$ can also be thought of as the group of rigid symmetries of the equilateral triangle (see Figure 11) with vertices:

$$P_1 = \begin{bmatrix} \sqrt{3}/2 \\ 1/2 \end{bmatrix}, P_2 = \begin{bmatrix} -\sqrt{3}/2 \\ 1/2 \end{bmatrix}, P_3 = \begin{bmatrix} 0 \\ -1 \end{bmatrix}.$$

The matrix $\rho_{(2,1)}(1,2)$ acts on the triangle by reflecting about the y-axis, and $\rho_{(2,1)}(1,2,3)$ by a $\pi/3$ counter-clockwise rotation.

In general, there are infinitely many reducible representations. For example, given any dimension d, there is a representation which maps every element of a group G to the $d \times d$ identity matrix (the direct sum of d copies of the trivial representation). However, for any finite group, there exists a finite collection of atomic representations which can be used to build up any other representation (up to equivalence) using the direct sum operation. These representations are referred to as the *irreducibles* of a group, and they are defined simply to be the collection of representations (up to equivalence) which are not reducible. It can be shown that any (complex) representation of a finite group G is equivalent to a direct sum of irreducibles [29], and hence, for any representation τ, there exists a matrix C for which

$$C^{-1} \cdot \tau \cdot C = \bigoplus_{\rho} \bigoplus_{j=1}^{z_\rho} \rho, \tag{5.4}$$

where ρ ranges over all distinct irreducible representations of the group G, and the inner $\oplus$ refers to some finite number (z_ρ) of copies of each irreducible ρ.

As it happens, there are only three irreducible representations of S_3 [29], up to equivalence: the trivial representation $\rho_{(3)}$, the degree 2 representation $\rho_{(2,1)}$, and the alternating representation $\rho_{(1,1,1)}$. The complete set of irreducible representation matrices of S_3 are shown in Table 2. Unfortunately, the analysis of the irreducible representations for $n > 3$ is far more complicated and we postpone this more general discussion for Section 5.2.

5.1.1 *The Fourier transform*

The link between group representation theory and Fourier analysis is given by the celebrated *Peter-Weyl theorem* ([29, 127, 114]) which says that the matrix entries of the irreducibles of G form a *complete* set of *orthogonal* basis functions on G.[3] The space of functions on S_3, for example, is orthogonally spanned by the 3! functions $\rho_{(3)}(\sigma)$, $[\rho_{(2,1)}(\sigma)]_{1,1}$, $[\rho_{(2,1)}(\sigma)]_{1,2}$,

3 Technically the Peter-Weyl result, as stated here, is only true if all of the representation matrices are unitary. That is, $\rho(\sigma)^*\rho(\sigma) = I$ for all $\sigma \in S_n$, where the matrix A^* is the conjugate transpose of A. For the case of real-valued (as opposed to complex-valued) matrices, however, the definitions of unitary and orthogonal matrices coincide.
While most representations are not unitary, there is a standard result from representation theory which shows that for *any* representation of G, there exists an equivalent unitary representation.

σ	$\rho_{(3)}$	$\rho_{(2,1)}$	$\rho_{(1,1,1)}$
ϵ	1	$\begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix}$	1
$(1,2)$	1	$\begin{bmatrix} -1 & 0 \\ 0 & 1 \end{bmatrix}$	-1
$(2,3)$	1	$\begin{bmatrix} 1/2 & \sqrt{3}/2 \\ \sqrt{3}/2 & -1/2 \end{bmatrix}$	-1
$(1,3)$	1	$\begin{bmatrix} 1/2 & -\sqrt{3}/2 \\ -\sqrt{3}/2 & -1/2 \end{bmatrix}$	-1
$(1,2,3)$	1	$\begin{bmatrix} -1/2 & -\sqrt{3}/2 \\ \sqrt{3}/2 & -1/2 \end{bmatrix}$	1
$(1,3,2)$	1	$\begin{bmatrix} -1/2 & \sqrt{3}/2 \\ -\sqrt{3}/2 & -1/2 \end{bmatrix}$	1

Table 2: The irreducible representation matrices of S_3.

$[\rho_{(2,1)}(\sigma)]_{2,1}$, $[\rho_{(2,1)}(\sigma)]_{2,2}$ and $\rho_{(1,1,1)}(\sigma)$, where $[\rho(\sigma)]_{ij}$ denotes the (i,j) entry of the matrix $\rho(\sigma)$.

As a replacement for projecting a function h onto a complete set of sinusoidal basis functions (as one would do on the real line), the Peter-Weyl theorem suggests instead to project onto the basis provided by the irreducibles of G. As on the real line, this projection can be done by computing the inner product of h with each element of the basis, and we define this operation to be the generalized form of the Fourier Transform.

Definition 24. Let $h : G \to \mathbb{R}$ be any function on a group G and let ρ be any representation on G. The *Fourier Transform* of h at the representation ρ is defined to be the matrix of coefficients:

$$\hat{h}_\rho = \sum_\sigma h(\sigma)\rho(\sigma). \tag{5.5}$$

The collection of Fourier Transforms at all irreducible representations of G form *the Fourier Transform of* h.

There are two important points which distinguish this Fourier Transform from its familiar formulation on the real line — first, the outputs of the transform are matrix-valued, and second, the inputs to $\hat{h}$ are *representations* of G rather than real numbers. As in the familiar formulation, the Fourier Transform is invertible and the inversion formula is explicitly given by the Fourier Inversion Theorem.

Theorem 25 (Fourier Inversion Theorem)**.**

$$h(\sigma) = \frac{1}{|G|} \sum_\lambda d_{\rho_\lambda} Tr\left[\hat{h}_{\rho_\lambda}^T \cdot \rho_\lambda(\sigma)\right], \tag{5.6}$$

where λ *indexes over the collection of irreducibles of* G.

Note that the trace term in the inverse Fourier Transform is just the 'matrix dot product' between $\hat{h}_{\rho_\lambda}$ and $\rho_\lambda(\sigma)$, since $\mathrm{Tr}\left[A^T \cdot B\right] = \langle \mathrm{vec}(A), \mathrm{vec}(B) \rangle$, where by vec we mean mapping a matrix to a vector on the same elements arranged in column-major order.

We now provide several examples for intuition. For functions on the real line, the Fourier Transform at zero frequency gives the DC component of a signal. The same holds true for functions on a group; If $h : G \to \mathbb{R}$ is any function, then since $\rho_{(n)} = 1$, the Fourier Transform of h at the trivial representation is constant, with $\hat{h}_{\rho_{(n)}} = \sum_\sigma h(\sigma)$. Thus, for any probability distribution, we have $\hat{h}_{\rho_{(n)}} = 1$. If U is the uniform distribution, then $\hat{U}_\rho = 0$ at every irreducible ρ except at the trivial representation.

The Fourier Transform at $\tau_{(n-1,1)}$ also has a simple interpretation:

$$\begin{aligned}
[\hat{h}_{\tau_{(n-1,1)}}]_{ij} &= \sum_{\sigma \in S_n} h(\sigma)[\tau_{(n-1,1)}(\sigma)]_{ij}, \\
&= \sum_{\sigma \in S_n} h(\sigma) \mathbb{1}\{\sigma(j) = i\}, \\
&= \sum_{\sigma : \sigma(j) = i} h(\sigma).
\end{aligned}$$

If h is a probability distribution, then $\hat{h}_{\tau_{(n-1,1)}}$ is a matrix of *first-order* marginal probabilities, where the (i, j)-th element is the marginal probability that a random permutation drawn from h maps element j to i.

Example 26. *Consider the following probability distribution on S_3:*

σ	ϵ	$(1,2)$	$(2,3)$	$(1,3)$	$(1,2,3)$	$(1,3,2)$
$h(\sigma)$	$1/3$	$1/6$	$1/3$	0	$1/6$	0

The set of all first order marginal probabilities is given by the Fourier transform at $\tau_{(2,1)}$:

$$\widehat{h}_{\tau_{(2,1)}} = \left[\begin{array}{c|ccc} & \mathbf{A} & \mathbf{B} & \mathbf{C} \\ \hline \mathbf{1} & 2/3 & 1/6 & 1/6 \\ \mathbf{2} & 1/3 & 1/3 & 1/3 \\ \mathbf{3} & 0 & 1/2 & 1/2 \end{array}\right].$$

In the above matrix, each column j represents a marginal distribution over the possible tracks that identity j can map to under a random draw from h. We see, for example, that Alice is at Track 1 with probability 2/3, or at Track 2 with probability 1/3. Simultaneously, each row i represents a marginal distribution over the possible identities that could have been mapped to track i under a random draw from h. In our example, Bob and Cathy are equally likely to be in Track 3, but Alice is definitely not in Track 3. Since each row and each column is itself a distribution, the matrix $\widehat{h}_{\tau_{(2,1)}}$ must be doubly stochastic. We will elaborate on the consequences of this observation later.

The Fourier transform of the same distribution at all irreducibles is:

$$\widehat{h}_{\rho_{(3)}} = 1, \quad \widehat{h}_{\rho_{(2,1)}} = \begin{bmatrix} 1/4 & \sqrt{3}/4 \\ \sqrt{3}/4 & 1/4 \end{bmatrix}, \quad \widehat{h}_{\rho_{(1,1,1)}} = 0.$$

The first-order permutation representation, $\tau_{(n-1,1)}$, captures the statistics of how a random permutation acts on a single object irrespective of where all of the other $n-1$ objects are mapped, and in doing so, compactly summarizes the distribution with only $O(n^2)$ numbers. Unfortunately, the Fourier transform at the first-order permutation representation cannot capture more complicated statements like:

$$P(\text{Alice and Bob occupy Tracks 1 and 2}) = 0.$$

To avoid collapsing away so much information, we might define richer summary statistics that might capture 'higher-order' effects. We define the *second-order unordered permutation representation* by:

$$[\tau_{(n-2,2)}(\sigma)]_{\{i,j\},\{k,\ell\}} = \mathbb{1}\{\sigma(\{k,\ell\}) = \{i,j\}\},$$

where we index the matrix rows and columns by unordered pairs $\{i,j\}$. The condition inside the indicator function states that the representation captures whether the pair of objects $\{k,\ell\}$ maps to the pair $\{i,j\}$, but is indifferent with respect to the ordering; i.e., either $k \mapsto i$ and $\ell \mapsto j$, *or*, $k \mapsto j$ and $\ell \mapsto i$.

Example 27. *For $n = 4$, there are six possible unordered pairs:* $\{1,2\}$, $\{1,3\}$, $\{1,4\}$, $\{2,3\}$, $\{2,4\}$, *and* $\{3,4\}$. *The matrix representation of the permutation* $(1,2,3)$ *is:*

$$\tau_{(2,2)}(1,2,3) = \left[\begin{array}{c|cccccc} & \{1,2\} & \{1,3\} & \{1,4\} & \{2,3\} & \{2,4\} & \{3,4\} \\ \hline \{1,2\} & 0 & 0 & 0 & 1 & 0 & 0 \\ \{1,3\} & 1 & 0 & 0 & 0 & 0 & 0 \\ \{1,4\} & 0 & 0 & 0 & 0 & 1 & 0 \\ \{2,3\} & 0 & 1 & 0 & 0 & 0 & 0 \\ \{2,4\} & 0 & 0 & 0 & 0 & 0 & 1 \\ \{3,4\} & 0 & 0 & 1 & 0 & 0 & 0 \end{array}\right].$$

The *second order ordered permutation representation*, $\tau_{(n-2,1,1)}$, is defined similarly:

$$[\tau_{(n-2,1,1)}(\sigma)]_{(i,j),(k,\ell)} = \mathbb{1}\{\sigma((k,\ell)) = (i,j)\},$$

where (k,ℓ) denotes an *ordered* pair. Therefore, $[\tau_{(n-2,1,1)}(\sigma)]_{(i,j),(k,\ell)}$ is 1 if and only if σ maps k to i *and* ℓ to j.

As in the first-order case, the Fourier transform of a probability distribution at $\tau_{(n-2,2)}$, returns a matrix of marginal probabilities of the form: $P(\sigma : \sigma(\{k,\ell\}) = \{i,j\})$, which captures statements like, "Alice and Bob occupy Tracks 1 and 2 with probability $1/2$". Similarly, the Fourier transform at $\tau_{(n-2,1,1)}$ returns a matrix of marginal probabilities of the form

$P(\sigma : \sigma((k,\ell)) = (i,j))$, which captures statements like, "Alice is in Track 1 *and* Bob is in Track 2 with probability 9/10".

We can go further and define third-order representations, fourth-order representations, and so on. In general however, the permutation representations as they have been defined above are reducible, intuitively due to the fact that it is possible to recover lower order marginal probabilities from higher order marginal probabilities. For example, one can recover the normalization constant (corresponding to the trivial representation) from the first order matrix of marginals by summing across either the rows or columns, and the first order marginal probabilities from the second order marginal probabilities by summing across appropriate matrix entries. To truly leverage the machinery of Fourier analysis, it is important to understand the Fourier transform at the irreducibles of the symmetric group, and in the next section, we show how to derive the irreducible representations of the Symmetric group by first defining permutation representations, then "subtracting off the lower-order effects".

5.2 REPRESENTATION THEORY OF THE SYMMETRIC GROUP

In this section, we provide a brief introduction to the representation theory of the symmetric group. Rather than giving a fully rigorous treatment of the subject, our goal is to give some intuition about the kind of information which can be captured by the irreducible representations of S_n. Roughly speaking, we will show that Fourier transforms on the symmetric group, instead of being indexed by frequencies, are indexed by *partitions* of n (tuples of numbers which sum to n), and certain partitions correspond to more complex basis functions than others. For proofs, we point the reader to consult: [30, 65, 114, 133].

Instead of the singleton or pairwise marginals which were described in the previous section, we will now focus on using the Fourier coefficients of a distribution to query a much wider class of marginal probabilities. As an example, we will be able to compute the following (more complicated) marginal probability on S_6 using Fourier coefficients:

$$\hat{h}\left(\sigma \;:\; \sigma\left(\left\{\begin{array}{|c|c|c|}\hline 1&2&3\\\hline 4&5\\\cline{1-2} 6\\\cline{1-1}\end{array}\right\}\right) = \left\{\begin{array}{|c|c|c|}\hline 1&2&6\\\hline 4&5\\\cline{1-2} 3\\\cline{1-1}\end{array}\right\}\right), \tag{5.7}$$

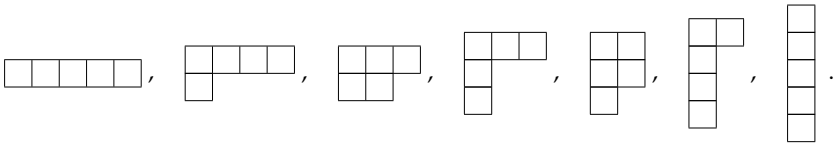
which we interpret as the joint marginal probability that the rows of the diagram on the left map to corresponding rows on the right as unordered sets. In other words, Equation 5.7 is the joint probability that *unordered* set $\{1,2,3\}$ maps to $\{1,2,6\}$, the unordered pair $\{4,5\}$ maps to $\{4,5\}$, and the singleton $\{6\}$ maps to $\{3\}$.

The diagrams in Equation 5.7 are known as *Ferrer's diagrams* and are commonly used to visualize *partitions* of n, which are defined to be unordered tuples of positive integers, $\lambda = (\lambda_1, \ldots, \lambda_\ell)$, which sum to n. For example, $\lambda = (3,2)$ is a partition of $n = 5$ since $3+2=5$. Usually we write partitions

as weakly decreasing sequences by convention, so the partitions of $n = 5$ are:

$$(5),\ (4,1),\ (3,2),\ (3,1,1),\ (2,2,1),\ (2,1,1,1),\ (1,1,1,1,1),$$

and their respective Ferrers diagrams are:



A *Young tabloid* is an assignment of the numbers $\{1,\ldots,n\}$ to the boxes of a Ferrers diagram for a partition λ, where each row represents an unordered set. There are 6 Young tabloids corresponding to the partition $\lambda = (2,2)$, for example:

$$\left\{\begin{array}{|c|c|}\hline 1&2\\\hline 3&4\\\hline\end{array}\right\},\left\{\begin{array}{|c|c|}\hline 1&3\\\hline 2&4\\\hline\end{array}\right\},\left\{\begin{array}{|c|c|}\hline 1&4\\\hline 2&3\\\hline\end{array}\right\},\left\{\begin{array}{|c|c|}\hline 2&3\\\hline 1&4\\\hline\end{array}\right\},\left\{\begin{array}{|c|c|}\hline 2&4\\\hline 1&3\\\hline\end{array}\right\},\left\{\begin{array}{|c|c|}\hline 3&4\\\hline 1&2\\\hline\end{array}\right\}.$$

The Young tabloid, $\left\{\begin{array}{|c|c|}\hline 1&2\\\hline 3&4\\\hline\end{array}\right\}$, for example, represents the two underordered sets $\{1,2\}$ and $\{3,4\}$, and if we were interested in computing the joint probability that $\sigma(\{1,2\}) = \{3,4\}$ and $\sigma(\{3,4\}) = \{1,2\}$, then we could write the problem in terms of Young tabloids as:

$$h\left(\sigma\ :\ \sigma\left(\left\{\begin{array}{|c|c|}\hline 1&2\\\hline 3&4\\\hline\end{array}\right\}\right) = \left\{\begin{array}{|c|c|}\hline 3&4\\\hline 1&2\\\hline\end{array}\right\}\right).$$

In general, we will be able to use the Fourier coefficients at irreducible representations to compute the marginal probabilities of Young tabloids. As we shall see, with the help of the *James Submodule theorem* [65], the marginals corresponding to "simple" partitions will require very few Fourier coefficients to compute, which is one of the main strengths of working in the Fourier domain.

Example 28. *Imagine three separate rooms containing two tracks each, in which Alice and Bob are in room 1 occupying Tracks 1 and 2; Cathy and David are in room 2 occupying Tracks 3 and 4; and Eric and Frank are in room 3 occupying Tracks 5 and 6, but we are not able to distinguish which person is at which track in any of the rooms. Then*

$$h\left(\sigma\ :\ \left(\left\{\begin{array}{|c|c|}\hline A&B\\\hline C&D\\\hline E&F\\\hline\end{array}\right\}\right) \to \left\{\begin{array}{|c|c|}\hline 1&2\\\hline 3&4\\\hline 5&6\\\hline\end{array}\right\}\right) = 1.$$

It is in fact, possible to recast the first-order marginals which were described in the previous section in the language of Young tabloids by

noticing that, for example, if 1 maps to 1, then the unordered set $\{2,\dots,n\}$ must map to $\{2,\dots,n\}$ since permutations are one-to-one mappings. The marginal probability that $\sigma(1) = 1$, then, is equal to the marginal probability that $\sigma(1) = 1$ *and* $\sigma(\{2,\dots,n\}) = \{2,\dots,n\}$. If $n = 6$, then the marginal probability written using Young tabloids is:

$$h\left(\sigma \,:\, \sigma\left(\left\{\begin{array}{|c|c|c|c|c|}\hline 2&3&4&5&6\\\hline 1\\\cline{1-1}\end{array}\right\}\right) = \left\{\begin{array}{|c|c|c|c|c|}\hline 2&3&4&5&6\\\hline 1\\\cline{1-1}\end{array}\right\}\right).$$

The first-order marginal probabilities correspond, therefore, to the marginal probabilities of Young tabloids of shape $\lambda = (n-1,1)$.

Likewise, the second-order unordered marginals correspond to Young tabloids of shape $\lambda = (n-2,2)$. If $n = 6$ again, then the marginal probability that $\{1,2\}$ maps to $\{2,4\}$ corresponds to the following marginal probability for tabloids:

$$h\left(\sigma \,:\, \sigma\left(\left\{\begin{array}{|c|c|c|c|c|}\hline 3&4&5&6\\\hline 1&2\\\cline{1-2}\end{array}\right\}\right) = \left\{\begin{array}{|c|c|c|c|c|}\hline 1&3&5&6\\\hline 2&4\\\cline{1-2}\end{array}\right\}\right).$$

The second-order *ordered* marginals are captured at the partition $\lambda = (n-2,1,1)$. For example, the marginal probability that $\{1\}$ maps to $\{2\}$ *and* $\{2\}$ maps to $\{4\}$ is given by:

$$h\left(\sigma \,:\, \sigma\left(\left\{\begin{array}{|c|c|c|c|c|}\hline 3&4&5&6\\\hline 1\\\cline{1-1}2\\\cline{1-1}\end{array}\right\}\right) = \left\{\begin{array}{|c|c|c|c|c|}\hline 1&3&5&6\\\hline 2\\\cline{1-1}4\\\cline{1-1}\end{array}\right\}\right).$$

And finally, we remark that the $(1,\dots,1)$ partition of n recovers all original probabilities since it asks for a joint distribution over $\sigma(1),\dots,\sigma(n)$. The corresponding matrix of marginals has $n! \times n!$ entries (though there will only be $n!$ distinct probabilities.

To see how the marginal probabilities of Young tabloids of shape λ can be thought of as Fourier coefficients, we will define a representation (which we call the *permutation representation*) associated with λ and show that the Fourier transform of a distribution at a permutation representation gives marginal probabilities. We begin by fixing an ordering on the set of possible Young tabloids, $\{t_1\},\{t_2\},\dots$, and define the permutation representation $\tau_\lambda(\sigma)$ to be the matrix:

$$[\tau_\lambda(\sigma)]_{ij} = \begin{cases} 1 & \text{if } \sigma(\{t_j\}) = \{t_i\} \\ 0 & \text{otherwise} \end{cases}. \tag{5.8}$$

It can be checked that the function τ_λ is indeed a valid representation of the symmetric group, and therefore we can compute Fourier coefficients at τ_λ. If $h(\sigma)$ is a probability distribution, then

$$\begin{aligned}\left[\widehat{h}_{\tau_\lambda}\right]_{ij} &= \sum_{\sigma\in S_n} h(\sigma)\,[\tau_\lambda(\sigma)]_{ij}\,,\\ &= \sum_{\{\sigma\,:\,\sigma(\{t_j\})=\{t_i\}\}} h(\sigma),\\ &= h(\sigma\ :\ \sigma(\{t_j\})=\{t_i\}),\end{aligned}$$

and therefore, *the matrix of marginals corresponding to Young tabloids of shape λ is given* exactly *by the Fourier transform at the representation τ_λ.*

As we showed earlier, the simplest marginals (the zeroth order normalization constant), correspond to the Fourier transform at $\tau_{(n)}$, while the first-order marginals correspond to $\tau_{(n-1,1)}$, and the second-order unordered marginals correspond to $\tau_{(n-2,2)}$. The list goes on and on, with the marginals getting more complicated. At the other end of the spectrum, we have the Fourier coefficients at the representation $\tau_{(1,1,\ldots,1)}$ which exactly recover the original probabilities $h(\sigma)$.

5.3 INTERPRETATIONS OF FREQUENCY ON THE SYMMETRIC GROUP

We have introduced partitions as a way of parameterizing different levels of complexity for marginals in a way that is highly reminiscent of the different frequency levels for real valued signals. In this section, we discuss in more detail how one might think about the notion of 'frequency' on the symmetric group.

5.3.1 *Dominance ordering and the James Submodule Theorem.*

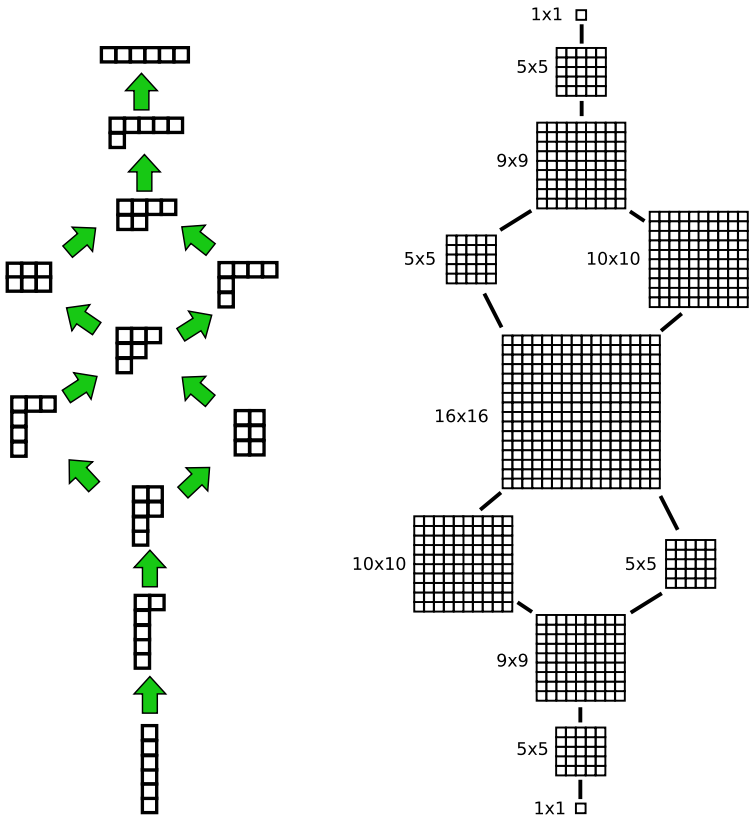
One way to think about frequency for permutations relies on the *dominance ordering* of partitions, which, unlike the ordering on frequencies (from low to high), is not a linear order, but rather, a partial order.

Definition 29 (Dominance Ordering)**.** Let λ, μ be partitions of n. Then $\lambda \trianglerighteq \mu$ (we say λ *dominates* μ), if for each i, $\sum_{k=1}^{i} \lambda_k \geqslant \sum_{k=1}^{i} \mu_k$.

For example, $(4,2) \trianglerighteq (3,2,1)$ since $4 \geqslant 3$, $4+2 \geqslant 3+2$, and $4+2+0 \geqslant 3+2+1$. However, $(3,3)$ and $(4,1,1)$ cannot be compared with respect to the dominance ordering since $3 \leqslant 4$, but $3+3 \geqslant 4+1$. The ordering over the partitions of $n = 6$ is depicted in Figure 12a.

Partitions with fat Ferrers diagrams tend to be greater (with respect to dominance ordering) than those with skinny Ferrers diagrams. Intuitively, representations corresponding to partitions which are *high* in the dominance ordering are 'low frequency', while represention corresponding to partitions which are *low* in the dominance ordering are 'high frequency'[4].

4 The direction of the ordering is slightly counterintuitive given the frequency interpretation, but is standard in the literature.



(a) Dominance ordering for $n = 6$.

(b) Fourier coefficient matrices for S_6.

Figure 12: The dominance order for partitions of $n = 6$ are shown in the left diagram (a). Fat Ferrer's diagrams tend to be higher in the order and long, skinny diagrams tend to be lower. The corresponding Fourier coefficient matrices for each partition (at irreducible representations) are shown in the right diagram (b). Note that since the Fourier basis functions form a complete basis for the space of functions on the symmetric group, there must be exactly $n!$ coefficients in total.

Having defined a family of intuitive permutation representations over the symmetric group, we can now ask whether the permutation representations are irreducible or not: the answer in general, is to the negative, due to the fact that it is often possible to reconstruct lower order marginals by summing over the appropriate higher order marginal probabilities. However, it is possible to show that, for each permutation representation τ_λ, there exists a corresponding irreducible representation ρ_λ, which, loosely, captures all of the information at the 'frequency' λ which was not already captured at lower frequency irreducibles. Moreover, it can be shown that there exists no irreducible representation besides those indexed by the partitions of n. These remarkable results are formalized in the *James Submodule Theorem*, which we state here without proof (see [29, 65, 114]).

Theorem 30 (James' Submodule Theorem)**.**

1. *(Uniqueness) For each partition, λ, of n, there exists an irreducible representation, ρ_λ, which is unique up to equivalence.*
2. *(Completeness) Every irreducible representation of S_n corresponds to some partition of n.*
3. *There exists a matrix C_λ associated with each partition λ, for which*

$$C_\lambda^T \cdot \tau_\lambda(\sigma) \cdot C_\lambda = \bigoplus_{\mu \trianglerighteq \lambda} \bigoplus_{\ell=1}^{K_{\lambda\mu}} \rho_\mu(\sigma), \quad \textit{for all } \sigma \in S_n. \tag{5.9}$$

4. $K_{\lambda\lambda} = 1$ *for all partitions λ.*

In plain English, part (3) of the James Submodule theorem says that we can always reconstruct marginal probabilities of λ-tabloids using the Fourier coefficients at irreducibles which lie *at* λ *and above* in the dominance ordering, if we have knowledge of the matrix C_λ (which can be precomputed using methods detailed in Appendix B), and the multiplicities $K_{\lambda\mu}$. In particular, combining Equation 5.9 with the definition of the Fourier transform, we have that

$$\hat{h}_{\tau_\lambda} = C_\lambda \cdot \left[\bigoplus_{\mu \trianglerighteq \lambda} \bigoplus_{\ell=1}^{K_{\lambda\mu}} \hat{h}_{\rho_\mu} \right] \cdot C_\lambda^T, \tag{5.10}$$

and so to obtain marginal probabilities of λ-tabloids, we simply construct a block diagonal matrix using the appropriate irreducible Fourier coefficients, and conjugate by C_λ. The multiplicities $K_{\lambda\mu}$ are known as the *Kostka numbers* and can be computed using Young's rule [114]. To illustrate using a few examples, we have the following decompositions:

$$\begin{aligned}
\tau_{(n)} &\equiv \rho_{(n)}, \\
\tau_{(n-1,1)} &\equiv \rho_{(n)} \oplus \rho_{(n-1,1)}, \\
\tau_{(n-2,2)} &\equiv \rho_{(n)} \oplus \rho_{(n-1,1)} \oplus \rho_{(n-2,2)}, \\
\tau_{(n-2,1,1)} &\equiv \rho_{(n)} \oplus \rho_{(n-1,1)} \oplus \rho_{(n-1,1)} \oplus \rho_{(n-2,2)} \oplus \rho_{(n-2,1,1)}, \\
\tau_{(n-3,3)} &\equiv \rho_{(n)} \oplus \rho_{(n-1,1)} \oplus \rho_{(n-2,2)} \oplus \rho_{(n-3,3)}, \\
\tau_{(n-3,2,1)} &\equiv \rho_{(n)} \oplus \rho_{(n-1,1)} \oplus \rho_{(n-1,1)} \oplus \rho_{(n-2,2)} \\
&\qquad \oplus \rho_{(n-2,2)} \oplus \rho_{(n-2,1,1)} \oplus \rho_{(n-3,3)} \oplus \rho_{(n-3,2,1)}.
\end{aligned}$$

Intuitively, the irreducibles at a partition λ reflect the "pure" λ^{th}-order effects of the underlying distribution. In other words, the irreducibles at λ form a basis for functions that have "interesting" λ^{th}-order marginal probabilities, but uniform marginals at all partitions μ such that $\mu \rhd \lambda$.

Example 31. *As an example, we demonstrate a "preference" function which is "purely" second-order (unordered) in the sense that its Fourier coefficients are equal to zero at all irreducible representations except $\rho_{(n-2,2)}$ (and the trivial representation). Consider the function $h : S_n \to \mathbb{R}$ defined by:*

$$h(\sigma) = \begin{cases} 1 & \textit{if } |\sigma(1) - \sigma(2)| \equiv 1 \ (\textit{mod } n) \\ 0 & \textit{otherwise} \end{cases}.$$

λ	(n)	$(n-1,1)$	$(n-2,2)$	$(n-2,1,1)$	$(n-3,3)$	$(n-3,2,1)$
$\dim \rho_\lambda$	1	$n-1$	$\frac{n(n-3)}{2}$	$\frac{(n-1)(n-2)}{2}$	$\frac{n(n-1)(n-5)}{6}$	$\frac{n(n-2)(n-4)}{3}$

Table 3: Dimensions of low-order irreducible representation matrices.

Intuitively, imagine seating n people at a round table with n chairs, but with the constraint that the first two people, Alice and Bob, are only happy if they are allowed to sit next to each other. In this case, h can be thought of as the indicator function for the subset of seating arrangements (permutations) which make Alice and Bob happy.

Since h depends only on the destination of the unordered pair $\{1,2\}$, its Fourier transform is zero at all partitions μ such that $\mu \lhd (n-2,2)$ ($\hat{h}_\mu = 0$). On the other hand, Alice and Bob have no individual preferences for seating, so the first-order "marginals" of h are uniform, and hence, $\hat{f}_{(n-1,1)} = 0$. The Fourier coefficients at irreducibles can be obtained from the second-order (unordered) "marginals" using Equation 5.9.

$$C_{(n-2,2)}^T \cdot \hat{h}_{\tau_{(n-2,2)}} \cdot C_{(n-2,2)} = \begin{bmatrix} Z & & \\ & 0 & \\ & & \hat{h}_{\rho_{(n-2,2)}} \end{bmatrix}.$$

5.3.2 *The size of an irreducible*

The sizes of the irreducible representation matrices are typically much smaller than their corresponding permutation representation matrices. In the case of $\lambda = (1,\ldots,1)$ for example, $\dim \tau_\lambda = n!$ while $\dim \rho_\lambda = 1$. There is a simple combinatorial algorithm, known as the *Hook Formula* [114], for computing the dimension of ρ_λ. While we do not discuss it, we provide a few dimensionality computations here (Table 3) to facilitate a dicussion of complexity later. Despite providing polynomial sized function approximations, the Fourier coefficient matrices can grow quite fast, and roughly, one would need $O(n^{2k})$ storage to maintain kth order marginals. For example, we would need to store $O(n^8)$ elements to maintain fourth-order marginals. It is worth noting that since the Fourier transform is invertible, there must be $n!$ Fourier coefficients in total, and so $\sum_\rho d_\rho^2 = |G| = n!$. See Figure 12b for an example of what the matrices of a complete Fourier transform on S_6 would look like.

5.4 SUMMARY

In this chapter we have provided the reader with a brief introduction to Fourier analysis for the symmetric group. The key point that we emphasized is that the problem of *compactly representating a distribution* can be approached by maintaining low frequency Fourier terms which correspond, in a sense, to marginal probabilities over small subsets of items, and can be represented compactly with polynomial storage space. Though these Fourier coefficients correspond to marginals at various frequency levels (ordered with respect to the dominance ordering on partitions), the actual irreducible coefficients that we store represent the "pure" effects of a distribution at a certain frequency level.

In practice, since the irreducible representation matrices are determined only up to equivalence, it is necessary to choose a basis for the irreducible representations in order to explicitly construct the representation matrices. In the next chapter, we visit some of these more numerical and computational issues behind Fourier analysis on the symmetric group, as we discuss algorithms for working with the *Gel'fand-Tsetlin basis* which has several attractive properties, two advantages being that the matrices are real-valued and orthogonal.

6

TABLEAUX COMBINATORICS AND ALGORITHMS

In Chapter 5, we have focused primarily on how to interpret Fourier coefficients for the symmetric group with an emphasis on probabilistic connections and intuitions. In this chapter, we turn to more computational issues and address the following matters:

- We show that the dimensions of the irreducible representation ρ_λ can be indexed by a collection of combinatorial objects known as the *standard Young tableaux*.
- Using the combinatorics of standard tableaux we provide an introduction to algorithms which efficiently compute explicit irreducible representation matrices for the symmetric group (i.e., the basis of functions upon which our additive decomposition methods project).
- Based on our explicit choice for the system of irreducible representation matrices, we introduce *Clausen's FFT (Fast Fourier transform) algorithm* [19] which is able to compute the Fourier transform of a function on S_n in faster than the naive $O(n!^2)$ running time.

Throughout the chapter, we exploit the decompositions described in Chapter 3 in multiple ways in order to derive efficient algorithms.

6.1 THE GEL'FAND-TSETLIN BASIS

In the previous chapter, we have defined the notion of an irreducible representation only up to equivalence, meaning that given explicit representation matrices $\{\rho_\lambda(\sigma) \ : \ \sigma \in S_n\}$, the set $\{C^{-1} \cdot \rho_\lambda(\sigma) \cdot C \ : \ \sigma \in S_n\}$ (obtained by changing the basis using any invertible matrix C) could have been used in its stead. For the purposes of implementation, however, the question of which specific basis we should use for expressing the irreducible representations is an important one.

In this section, we present some standard algorithms for constructing the irreducible representation matrices with respect to the *Gel'fand-Tsetlin (GZ) basis* (constructed with respect to the subgroup chain $S_1 \subset S_2 \subset \cdots \subset S_n$).[1] There are several properties which make the irreducible representation matrices, written with respect to the GZ basis, fairly useful in practice. They are guaranteed to be, for example, real-valued and orthogonal. And as we will show, the matrices have certain useful sparsity properties that can be exploited in implementation for improved running time complexity. We note that the techniques in this chapter are standard in the group

1 The irreducible representation matrices in this chapter are also sometimes referred to as *Young's Orthogonal Representation (YOR)*.

representation theory literature and we present most of the following results without proof. For a more elaborate discussion, we refer the reader to [75, 18, 133].

THE BRANCHING RULE AND THE GEL'FAND-TSETLIN BASIS. To describe the Gel'fand-Tsetlin basis, we introduce a fundamental fact about the representation theory of the symmetric group known as the *branching rule*. It is straightforward to see that any irreducible representation ρ_λ of S_n, can also be seen as a representation of S_{n-1} when restricted to permutations in the subgroup $S_{n-1} \subset S_n$ (recall that in Section 2.4, we defined S_{n-1} to be the subset of permutations in S_n which fix n). Despite the fact that ρ_λ is irreducible as a representation of S_n, as a representation of S_{n-1}, ρ_λ is not necessarily irreducible and might decompose as a direct sum of irreducible representations of S_{n-1}, as Equation 5.4 would dictate. Thus, for all $\sigma \in S_{n-1}$ (i.e., all $\sigma \in S_n$ such that $\sigma(n) = n$), there exists C_λ and multiplicities $z_{\lambda\mu}$ such that:

$$C_\lambda^{-1} \cdot \rho_\lambda(\sigma) \cdot C_\lambda = \bigoplus_{\mu} \bigoplus_{j=1}^{z_{\lambda\mu}} \rho_\mu(\sigma),$$

where μ ranges over the partitions of $n-1$.

The *branching rule* (due originally to Peel [107]) allows us to state the decomposition even more precisely. Given any partition λ of n, let λ^- index over the set of partitions of $n-1$ whose Ferrers diagrams differ from λ in a single box.

Theorem 32 (Branching Rule, see Vershik and Okounkov [133] for a proof)**.** *For each irreducible representation ρ_λ of S_n, there exists a matrix C_λ such that:*

$$C_\lambda^{-1} \cdot \rho_\lambda(\sigma) \cdot C_\lambda = \bigoplus_{\lambda^-} \rho_{\lambda^-}(\sigma) \tag{6.1}$$

holds for any $\sigma \in S_{n-1}$.

Example 33. *If $\lambda = (3,2)$, then its corresponding Ferrers diagram is:*  *, and the Ferrers diagrams corresponding to partitions of 4 which differ from λ in a single box are:*

 , *and*  .

Thus, λ^- indexes over the set $\{(2,2),(3,1)\}$. The branching rule states that given an irreducible matrix representation $\rho_{(3,2)}$ of S_5, there is a matrix $C_{(3,2)}$ such that, for any permutation $\sigma \in S_5$ such that $\sigma(5) = 5$,

$$C_{(3,2)}^{-1} \cdot \rho_{(3,2)}(\sigma) \cdot C_{(3,2)} = \left[\begin{array}{c|c} \rho_{(2,2)}(\sigma) & 0 \\ \hline 0 & \rho_{(3,1)}(\sigma) \end{array}\right].$$

Replacing ρ_λ with an equivalent irreducible representation always results in the same decomposition although with a different matrix C_λ. The

Gel'fand-Tsetlin (GZ) basis for irreducible representations of the symmetric group is chosen exactly so that this conjugation by C is unneccesary (i.e., such that the branching rule holds with all $C_\lambda = I$, the $d_\lambda \times d_\lambda$ identity matrix). Thus the irreducible representation matrices constructed with respect to the GZ basis have the property that the equation:

$$\rho_\lambda(\sigma) = \bigoplus_{\lambda^-} \rho_{\lambda^-}(\sigma) \tag{6.2}$$

holds identically for all $\sigma \in S_{n-1}$ and any partition λ of n. To see why one can always choose a basis so that $C_\lambda = I$, observe that given any irreducible representation ρ_λ such that the branching rule holds with C_λ, there exists an equivalent irreducible representation defined by $\rho'_\lambda = C \cdot \rho_\lambda \cdot C^{-1}$ for which the branching rule must hold with $C'_\lambda = I$.

In the next sections, we explore a consequence of the above Gel'fand-Tsetlin assumption — that the GZ basis elements for each irreducible representation are in bijection with a collection of combinatorial objects known as the *standard Young tableaux*. Using this association allows us to then state a number of concrete algorithms in terms of standard Young tableaux.

6.2 RECURSIVE APPLICATIONS OF THE BRANCHING RULE AND BRANCHING SEQUENCES

We first show that each GZ basis element for an irreducible representation ρ_λ can be associated to a particular sequence of partitions (or Ferrers diagrams). Together, all "legal" sequences (which we define shortly) can be used to index the collection of basis vectors for ρ_λ. More precisely, we assume in the following that ρ_λ operates on a vector space spanned by GZ basis elements $\{b_1, \ldots, b_{d_\lambda}\}$, and we will associate each b_i to a sequence of n partitions

$$b_i : \qquad [\mu^{(1)} \to \mu^{(2)} \to \ldots \mu^{(n)}],$$

where each $\mu^{(k)}$ is a partition of k, and $\mu^{(n)} = \lambda$. Alternatively, one can think of each column of the representation ρ_λ as being associated with a sequence of partitions.

To establish this association between GZ basis elements and sequences of partitions, observe that the branching rule (together with Equation 6.2) allows us to associate each basis element b_i with some partition $\mu^{(n-1)}$ of $n-1$.

Example 34. *By the branching rule, we have that* $\rho_{(3,2)}(\sigma) = \rho_{(2,2)}(\sigma) \oplus \rho_{(3,1)}(\sigma)$ *for all* $\sigma \in S_4 \subset S_5$ *when using the GZ basis (see also Example 33). Therefore, when applied to vectors in the subspace spanned by* $\{b_1, b_2\}$*, the matrix* $\rho_{(3,2)}(\sigma)$ *behaves the same as* $\rho_{(2,2)}(\sigma)$*. Similarly, when applied to vectors in the subspace spanned by* $\{b_3, b_4, b_5\}$*, the matrix* $\rho_{(3,2)}(\sigma)$ *behaves the same as* $\rho_{(3,1)}(\sigma)$*.*

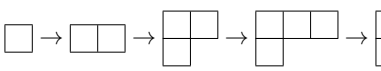
To b_1 *and* b_2*, we therefore associate the partition* $\mu^{(n-1)} = (2,2)$*, and to* b_3*,* b_4*, and* b_5*, we associate the partition* $\mu^{(n-1)} = (3,1)$*.*

We can then recursively apply the branching rule again (thus restricting to the subgroup $S_{n-2} = \{\sigma \in S_n \ : \ \sigma(n-1) = n-1, \sigma(n) = n\}$), to see that the following decomposition holds:

$$\rho_\lambda(\sigma) = \bigoplus_{\lambda^-} \left[\bigoplus_{\lambda^{--}} \rho_{\lambda^{--}}(\sigma) \right], \qquad \text{for any } \sigma \in S_{n-2} \subset S_n,$$

where λ^{--} indexes over partitions which differ from λ^- by a single box. Thus each basis element b_i can be associated with a partition $\mu^{(n)} = \lambda$ of n, a partition $\mu^{(n-1)}$ of $n-1$ *and* a partition $\mu^{(n-2)}$ of $n-2$. Taking this logic even further, we can restrict to S_{n-3}, S_{n-4}, and so on until we can restrict no further, associating each b_i with a sequence of partitions $\mu^{(1)}$ of 1, $\mu^{(2)}$ of 2, ..., $\mu^{(n)}$ of n, where, (1) each partition $\mu^{(i)}$ can be obtained by adding a single box to the Ferrers diagram of $\mu^{(i-1)}$, and (2) $\mu^{(n)} = \lambda$. We will refer to such a sequence as a *branching sequence*. Since the branching rule of Equation 6.1 guarantees multiplicity-free decompositions (that is, $z_{\lambda\mu} = 1$ for all pairs (λ, μ)), it turns out that each GZ basis element b_i is *uniquely* specified by a branching sequence.

Example 35. *A possible branching sequence is:*



or written as partitions, $[(1) \to (2) \to (2,1) \to (3,1) \to (3,2)]$.

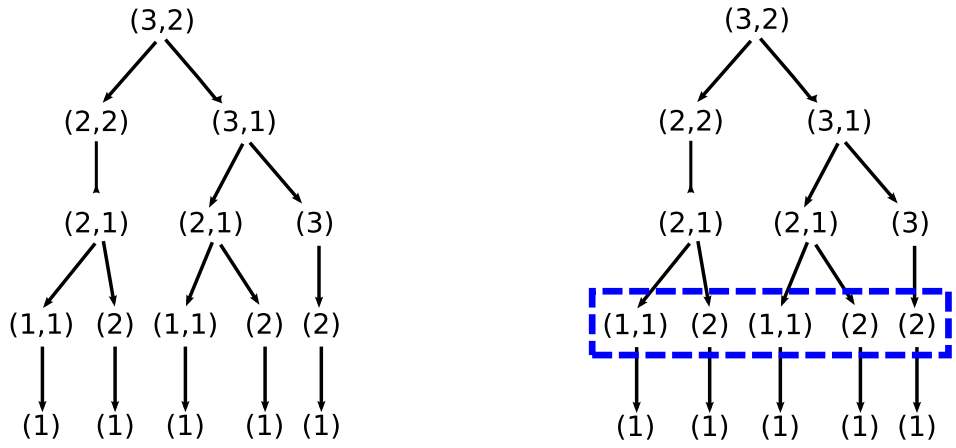
The set of all possible branching sequences ending in λ can be visualized using a *branching tree* (shown for $\lambda = (3,2)$ in Figure 13a), where each branching sequence is a path between the root and some leaf node. We will denote the branching tree corresponding to the partition λ by $\mathcal{T}^\lambda$ and the set of nodes at the r^{th} level of $\mathcal{T}^\lambda$ by $\mathcal{T}^\lambda_r$ (where the root node forms the zeroth level by convention). We can generalize the branching rule to handle recursive restrictions by rewriting the same rule in terms of the branching tree.

Proposition 36. *Let ρ_λ be an irreducible matrix representation of S_n (constructed with respect to the Gel'fand-Tsetlin basis). For any $\sigma \in S_k \subset S_n$, $\rho_\lambda(\sigma)$ decomposes as:*

$$\rho_\lambda(\sigma) = \bigoplus_{\mu \in \mathcal{T}^\lambda_{n-k}} \rho_\mu(\sigma).$$

Example 37. *As an example, consider applying Proposition 36 to $\rho_{(3,2)}$ with $k = 3$. The $(n-k)$th (second) level of the branching tree for $\lambda = (3,2)$, $\mathcal{T}^{(3,2)}_2$ consists of two copies of the partition $(2,1)$ and a single copy of the partition (3). Thus for any element $\sigma \in S_5$ which fixes 4 and 5 (i.e., any $\sigma \in S_5$ such that $\sigma(4) = 4$ and $\sigma(5) = 5$), we have:*

$$\rho_{(3,2)}(\sigma) = \begin{bmatrix} \rho_{(2,1)}(\sigma) & & \\ & \rho_{(2,1)}(\sigma) & \\ & & \rho_{(3)}(\sigma) \end{bmatrix}.$$



(a) $\mathcal{T}^{(3,2)}$ (b) $\mathcal{T}_3^{(3,2)}$

Figure 13: (a) The branching tree for $\lambda = (3,2)$. (b) The 3^{rd} level of $\mathcal{T}^{(3,2)}$ (outlined) is denoted by $\mathcal{T}_3^{(3,2)}$ and consists of two copies of the partition $(1,1)$ and three copies of the partition (2).

6.3 INDEXING BASIS ELEMENTS WITH STANDARD YOUNG TABLEAUX

Branching sequences can be compactly encoded by labeling each box of a Ferrers diagram of λ by the point in the sequence in which that box is added. For example, labeling boxes as they appear in the branching sequence $[(1) \to (2) \to (2,1) \to (3,1) \to (3,2)]$ yields:

$$\boxed{1} \to \boxed{1}\boxed{2} \to \begin{array}{|c|c|}\hline 1 & 2 \\ \hline 3 \\ \cline{1-1}\end{array} \to \begin{array}{|c|c|c|}\hline 1 & 2 & 4 \\ \hline 3 \\ \cline{1-1}\end{array} \to \begin{array}{|c|c|c|}\hline 1 & 2 & 4 \\ \hline 3 & 5 \\ \cline{1-2}\end{array}.$$

The entire sequence can therefore be associated with the following Ferrers diagram with labeled boxes:

$$\begin{array}{|c|c|c|}\hline 1 & 2 & 4 \\ \hline 3 & 5 \\ \cline{1-2}\end{array}.$$

Such labeled Ferrers diagrams are called *Young tableaux* which are like Young tabloids (see Section 5.2) with the distinction that the rows are considered as *ordered tuples* rather than *unordered sets*. For example, the following two diagrams are distinct as Young *tableaux*, but not as Young *tabloids*:

$$\begin{array}{|c|c|c|}\hline 1 & 2 & 3 \\ \hline 4 & 5 \\ \cline{1-2}\end{array} \neq \begin{array}{|c|c|c|}\hline 1 & 3 & 2 \\ \hline 5 & 4 \\ \cline{1-2}\end{array} \qquad \text{(as Young tableaux).}$$

While every branching sequence corresponds to some unique Young tableaux, not every Young tableaux can be constructed from a branching sequence. Instead, branching sequences are associated only with Young tableaux whose entries are increasing to the right along rows and down columns. We call such tableaux *standard*:

Definition 38. A Young tableaux is called *standard* if its labeled entries increase to the right along rows and down columns.

For example, the set of all standard Young Tableaux of shape $\lambda = (3,2)$ is:

$$\left\{ \begin{array}{|c|c|c|}\hline 1&3&5\\\hline 2&4\\\cline{1-2}\end{array}\;,\quad \begin{array}{|c|c|c|}\hline 1&2&5\\\hline 3&4\\\cline{1-2}\end{array}\;,\quad \begin{array}{|c|c|c|}\hline 1&3&4\\\hline 2&5\\\cline{1-2}\end{array}\;,\quad \begin{array}{|c|c|c|}\hline 1&2&4\\\hline 3&5\\\cline{1-2}\end{array}\;,\quad \begin{array}{|c|c|c|}\hline 1&2&3\\\hline 4&5\\\cline{1-2}\end{array} \right\}. \tag{6.3}$$

The significance of the standard tableaux is that the set of all standard tableaux of shape λ can be used to index the set of GZ basis vectors for the irreducible representation ρ_λ. Since there are five total standard tableaux of shape $(3,2)$, we see, for example, that the irreducible corresponding to the partition $(3,2)$ is 5-dimensional. There is a simple recursive procedure for enumerating the set of all standard tableaux of shape λ, which we illustrate for $\lambda = (3,2)$.

Example 39. *If $\lambda = (3,2)$, there are only two possible boxes that the label 5 can occupy so that both rows and columns are increasing. They are:*

$$\begin{array}{|c|c|c|}\hline &&5\\\hline &\\\cline{1-2}\end{array}\;, \text{ and } \begin{array}{|c|c|c|}\hline &&\\\hline &5\\\cline{1-2}\end{array}.$$

To enumerate the set of all standard tableaux of shape $(3,2)$, we need to fill the empty boxes in the above partially filled tableaux with the labels $\{1,2,3,4\}$ so that both rows and columns are increasing. Enumerating the standard tableaux of shape $(3,2)$ thus reduces to enumerating the set of standard tableaux of shapes $(2,2)$ and $(3,1)$, respectively. For $(2,2)$, the set of standard tableaux (which, in implementation would be computed recursively) is:

$$\left\{ \begin{array}{|c|c|}\hline 1&3\\\hline 2&4\\\cline{1-2}\end{array}\;,\quad \begin{array}{|c|c|}\hline 1&2\\\hline 3&4\\\cline{1-2}\end{array} \right\},$$

and for $(3,1)$, the set of standard tableaux is:

$$\left\{ \begin{array}{|c|c|c|}\hline 1&3&4\\\hline 2\\\cline{1-1}\end{array}\;,\quad \begin{array}{|c|c|c|}\hline 1&2&4\\\hline 3\\\cline{1-1}\end{array}\;,\quad \begin{array}{|c|c|c|}\hline 1&2&3\\\hline 4\\\cline{1-1}\end{array} \right\}.$$

The entire set of standard tableaux of shape $(3,2)$ is therefore:

$$\left\{ \begin{array}{|c|c|c|}\hline 1&3&\mathbf{5}\\\hline 2&4\\\cline{1-2}\end{array}\;,\quad \begin{array}{|c|c|c|}\hline 1&2&\mathbf{5}\\\hline 3&4\\\cline{1-2}\end{array} \right\} \bigcup \left\{ \begin{array}{|c|c|c|}\hline 1&3&4\\\hline 2&\mathbf{5}\\\cline{1-2}\end{array}\;,\quad \begin{array}{|c|c|c|}\hline 1&2&4\\\hline 3&\mathbf{5}\\\cline{1-2}\end{array}\;,\quad \begin{array}{|c|c|c|}\hline 1&2&3\\\hline 4&\mathbf{5}\\\cline{1-2}\end{array} \right\}.$$

To summarize the main points of this section, we have discussed how the Gel'fand-Tsetlin (GZ) basis (adapted to the subgroup chain $S_1 \subset \cdots \subset S_n$) is defined so that the branching rule holds as a matrix identity (with no need of a change of basis matrix). Furthermore, each basis vector of the representation space for ρ_λ can be associated with a branching sequence of partitions, or equivalently, a standard tableau. And finally, we showed that using a recursive procedure for enumerating standard tableaux, it is possible to compute the dimension of any irreducible representation ρ_λ.

6.4 CONSTRUCTING IRREDUCIBLE REPRESENTATION MATRICES

Using the Gel'fand-Tsetlin basis that we have defined, we now present algorithms for constructing irreducible representation matrices. The first part of this section describes how to construct the representation matrix ρ_λ with respect to the GZ basis when evaluated at an adjacent transposition of the form $\sigma = (i-1, i)$. In the second part of this section, we use the adjacent transpositions as a generating set (Definition 5) to construct representation matrices evaluated at arbitrary permutations. We present all results in this section without proofs, though they can be found in textbooks such as James and Kerber [65], Chen [18], and Sagan [114].

We will phrase the matrix representation construction algorithm in terms of the *axial distance* which we now define. Given a permutation $\sigma \in S_n$, one can always apply σ to a Young tableau t to get a new Young tableau, which we denote by $\sigma \circ t$, by permuting the labels within the tableau. For example,

$$(1,2) \circ \begin{array}{|c|c|c|}\hline \mathbf{1} & \mathbf{2} & 3 \\ \hline 4 & 5 \\ \cline{1-2}\end{array} = \begin{array}{|c|c|c|}\hline \mathbf{2} & \mathbf{1} & 3 \\ \hline 4 & 5 \\ \cline{1-2}\end{array}.$$

Note, however, that even if t is a standard tableau, $\sigma \circ t$ is not guaranteed to be standard.

Definition 40. The *axial distance*, $d_t(i,j)$, between entries i and j in tableau t, is defined to be:

$$d_t(i,j) \equiv (\mathrm{col}(t,j) - \mathrm{col}(t,i)) - (\mathrm{row}(t,j) - \mathrm{row}(t,i)),$$

where $\mathrm{row}(t,i)$ denotes the row of label i in tableau t, and $\mathrm{col}(t,i)$ denotes the column of label i in tableau t.

Intuitively, the axial distance between i and j in a standard tableau t is equal to the (signed) number of steps that are required to travel from i to j, if at each step, one is allowed to traverse a single box in the tableau in one of the four cardinal directions. For example, the axial distance from 3 to 4 with respect to tableau: $t = \begin{array}{|c|c|c|}\hline 1 & 2 & 3 \\ \hline 4 & 5 \\ \cline{1-2}\end{array}$ is:

$$\begin{aligned} d_t(3,4) &= \left(\mathrm{col}\left(\begin{array}{|c|c|c|}\hline 1 & 2 & 3 \\ \hline 4 & 5 \\ \cline{1-2}\end{array}, 4\right) - \mathrm{col}\left(\begin{array}{|c|c|c|}\hline 1 & 2 & 3 \\ \hline 4 & 5 \\ \cline{1-2}\end{array}, 3\right)\right) \\ &\quad - \left(\mathrm{row}\left(\begin{array}{|c|c|c|}\hline 1 & 2 & 3 \\ \hline 4 & 5 \\ \cline{1-2}\end{array}, 4\right) - \mathrm{row}\left(\begin{array}{|c|c|c|}\hline 1 & 2 & 3 \\ \hline 4 & 5 \\ \cline{1-2}\end{array}, 3\right)\right), \\ &= (1-3) - (2-1) = -3. \end{aligned}$$

6.4.1 *Constructing representation matrices for adjacent transpositions*

Given any partition λ of n, and any i such that $2 \leqslant i \leqslant n$, we now show how to construct the matrix $\rho_\lambda(i-1, i)$. In the following discussion, we will consider a fixed ordering, $t_1, \ldots, t_{d_\lambda}$, on the set of standard tableaux of shape λ and refer to both standard tableaux and columns of $\rho_\lambda(\sigma)$ interchangeably. Thus t_1 refers to first column, t_2 refers to the second

column and so on. And we will index elements in $\rho_\lambda(\sigma)$ using pairs of standard tableau, (t_j, t_k). The rule for constructing the matrix coefficient $[\rho_\lambda(i-1,i)]_{t_j,t_k}$ is as follows.

1. Define the (t_j, t_k) coefficient of $\rho_\lambda(i-1,i)$ to be zero if it is (1), off-diagonal ($j \neq k$) and (2), not of the form $(t_j, (i-1,i) \circ t_k)$.

2. If (t_j, t_k) is a diagonal element, (i.e., of the form (t_j, t_j)), define:

$$[\rho_\lambda(i-1,i)]_{t_j,t_j} = 1/d_{t_j}(i-1,i),$$

   where $d_{t_j}(i-1,i)$ is the axial distance which we defined earlier in the section.

3. If (t_j, t_k) can be written as $(t_j, (i-1,i) \circ t_j)$ define:

$$[\rho_\lambda(i-1,i)]_{t_j,\sigma\circ t_j} = \sqrt{1-1/d^2_{t_j}(i-1,i)}.$$

Note that the only time that off-diagonal elements can be nonzero under the above rules is when $(i-i,i) \circ t_j$ happens to also be a standard tableau. If we apply an adjacent transposition, $\sigma = (i-1,i)$ to a standard tableau t, then $\sigma \circ t$ is guaranteed to be standard if and only if $i-1$ and i were neither in the same row nor column of t. This can be seen by examining each case separately.

1. $i-1$ **and** i **are in the same row or same column of** t**.** If i and $i-1$ are in the same row of t, then $i-1$ lies to the left of i. Applying $\sigma \circ t$ swaps their positions so that i lies to the left of $i-1$, and so we see that $\sigma \circ t$ cannot be standard. For example,

$$(3,4) \circ \begin{array}{|c|c|c|}\hline 1 & 2 & 5 \\ \hline 3 & 4 \\ \cline{1-2}\end{array} = \begin{array}{|c|c|c|}\hline 1 & 2 & 5 \\ \hline 4 & 3 \\ \cline{1-2}\end{array}.$$

   Similarly, we see that if i and $i-1$ are in the same column of t, $\sigma \circ t$ cannot be standard. For example,

$$(3,4) \circ \begin{array}{|c|c|c|}\hline 1 & 3 & 5 \\ \hline 2 & 4 \\ \cline{1-2}\end{array} = \begin{array}{|c|c|c|}\hline 1 & 4 & 5 \\ \hline 2 & 3 \\ \cline{1-2}\end{array}.$$

2. $i-1$ **and** i **are neither in the same row nor column of** t**.** In the second case, $\sigma \circ t$ can be seen to be a standard tableau due to the fact that $i-1$ and i are adjacent indices. For example,

$$(3,4) \circ \begin{array}{|c|c|c|}\hline 1 & 2 & 3 \\ \hline 4 & 5 \\ \cline{1-2}\end{array} = \begin{array}{|c|c|c|}\hline 1 & 2 & 4 \\ \hline 3 & 5 \\ \cline{1-2}\end{array}.$$

Therefore, to see if $(i-1,i) \circ t$ is standard, we need only check to see that $i-1$ and i are in different rows and columns of the tableau t. The pseudocode for constructing the irreducible representation matrices for adjacent swaps is summarized in Algorithm 6.1. Note that the matrices constructed in the algorithm are sparse, with no more than two nonzero elements in any given column.

Algorithm 6.1: Pseudocode for computing irreducible representations matrices with respect to the Gel'fand-Tsetlin basis at adjacent transpositions. Input: $i \in \{2, \ldots, n\}$ and a partition λ. Output: $\rho_\lambda(i-1, i)$.

ADJACENTRHO(i, λ)

$\rho \leftarrow \mathbf{0}_{d_\lambda \times d_\lambda}$;
foreach *standard tableaux* t *of shape* λ **do**
  $d \leftarrow (\mathrm{col}(t, i) - \mathrm{col}(t, i-1)) - (\mathrm{row}(t, i) - \mathrm{row}(t, i-1))$;
  $\rho(t, t) \leftarrow 1/d$;
  if $i-1$ *and* i *are in different rows and columns of* t **then**
    $\rho((i-1, i) \circ (t), t) \leftarrow \sqrt{1 - 1/d^2}$;
  end
end
return (ρ);

Example 41. *We compute the representation matrix of* $\rho_{(3,2)}$ *evaluated at the adjacent transposition* $\sigma = (i-1, i) = (3, 4)$. *For this example, we will use the enumeration of the standard tableaux of shape* $(3, 2)$ *given in Equation 6.3.*

For each $(3, 2)$*-tableau* t_j, *we identify whether* $\sigma \circ t_j$ *is standard and compute the axial distance from* 3 *to* 4 *on the tableau* t_j.

j	*1*	*2*	*3*	*4*	*5*
t_j	$\begin{array}{\|c\|c\|c\|}\hline 1 & 3 & 5 \\ \hline 2 & 4 \\ \cline{1-2}\end{array}$	$\begin{array}{\|c\|c\|c\|}\hline 1 & 2 & 5 \\ \hline 3 & 4 \\ \cline{1-2}\end{array}$	$\begin{array}{\|c\|c\|c\|}\hline 1 & 3 & 4 \\ \hline 2 & 5 \\ \cline{1-2}\end{array}$	$\begin{array}{\|c\|c\|c\|}\hline 1 & 2 & 4 \\ \hline 3 & 5 \\ \cline{1-2}\end{array}$	$\begin{array}{\|c\|c\|c\|}\hline 1 & 2 & 3 \\ \hline 4 & 5 \\ \cline{1-2}\end{array}$
$(3,4) \circ t_j$	$\begin{array}{\|c\|c\|c\|}\hline 1 & 4 & 5 \\ \hline 2 & 3 \\ \cline{1-2}\end{array}$	$\begin{array}{\|c\|c\|c\|}\hline 1 & 2 & 5 \\ \hline 4 & 3 \\ \cline{1-2}\end{array}$	$\begin{array}{\|c\|c\|c\|}\hline 1 & 4 & 3 \\ \hline 2 & 5 \\ \cline{1-2}\end{array}$	$\begin{array}{\|c\|c\|c\|}\hline 1 & 2 & 3 \\ \hline 4 & 5 \\ \cline{1-2}\end{array}$	$\begin{array}{\|c\|c\|c\|}\hline 1 & 2 & 4 \\ \hline 3 & 5 \\ \cline{1-2}\end{array}$
$(3,4) \circ t_j$ *Standard?*	*No*	*No*	*No*	*Yes*	*Yes*
axial distance $(d_{t_j}(3,4))$	*-1*	*1*	*1*	*3*	*-3*

Putting the results together in a matrix yields:,

$$\rho_{(3,2)}(3,4) = \left[\begin{array}{c|ccccc} & t_1 & t_2 & t_3 & t_4 & t_5 \\ \hline t_1 & -1 & & & & \\ t_2 & & 1 & & & \\ t_3 & & & 1 & & \\ t_4 & & & & \frac{1}{3} & \sqrt{\frac{8}{9}} \\ t_5 & & & & \sqrt{\frac{8}{9}} & -\frac{1}{3} \end{array}\right],$$

where all of the empty entries are zero.

6.4.2 *Constructing representation matrices for general permutations*

To construct representation matrices for general permutations, it is enough to observe that all permutations can be factored into a sequence of adjacent swaps (Chapter 2). For example, since the permutation $(1, 2, 5)$ can be factored into:

$$(1, 2, 5) = (4, 5)(3, 4)(1, 2)(2, 3)(3, 4)(4, 5),$$

Algorithm 6.2: Pseudocode for computing irreducible representation matrices for arbitrary permutations. Input: any permutation $\sigma \in S_n$, and partition λ. Output: the representation matrix $\rho_\lambda(\sigma)$ (with respect to the Gel'fand-Tsetlin basis).

```
GETRHO(σ, λ):

    ρ_λ(σ) ← I_{d_λ × d_λ} ;
    factors ← PERMTOADJSWAPS(σ) ;
    m ← length(factors);
    for j = 1, ..., m do
        (i − 1, i) ← factors(j) ;
        ρ_λ(σ) ← GETADJACENTRHO (i, λ) · ρ_λ(σ) ;
    end
    return (ρ);
```

we have that for any partition λ,

$$\rho_\lambda(1,2,5) = \rho_\lambda(4,5) \cdot \rho_\lambda(3,4) \cdot \rho_\lambda(1,2) \cdot \rho_\lambda(2,3) \cdot \rho_\lambda(3,4) \cdot \rho_\lambda(4,5),$$

since ρ_λ is a group representation. The pseudocode in Algorithm 6.2 constructs representation matrices for general permutations assuming the existence of a subroutine for constructing representation matrices for adjacent permutations.

6.5 FOURIER TRANSFORMING THE INDICATOR FUNCTION OF $S_k \subset S_n$

In addition to facilitating the computation of irreducible representation matrices, the Gel'fand-Tsetlin basis allows for us to often identify structure in the Fourier transform in certain functions. In this section, we consider a particularly useful class of functions — the indicator function of the subgroup $S_k \subset S_n$, which we study further in Chapter 7 in the context of building probabilistic models.

We first show how the branching rule allows us to decompose the Fourier transform of a function that is supported on the subgroup $S_{n-1} \subset S_n$.

Corollary 42. *If $h : S_n \to \mathbb{R}$ is supported on the subgroup S_{n-1}, then for each partition λ, the Fourier transform of h (with respect to the Gel'fand-Tsetlin basis adapted $S_1 \subset S_2 \subset \cdots \subset S_n$) decomposes into a direct sum of Fourier transforms on S_{n-1}. Specifically, we have:*

$$\widehat{h}_{\rho_\lambda} = \bigoplus_{\lambda^-} \textsc{Restrict}[\widehat{h}_{\rho_{\lambda^-}}], \tag{6.4}$$

where $\textsc{Restrict}[\hat{h}]$ *is the Fourier transform of* $\textsc{Restrict}[h]$*, the restriction of h to S_{n-1} (see Equation 3.12).*

Proof. Equation 6.4 follows as a direct corollary of the Branching rule (Equation 6.1) and the Definition of the Fourier transform (Definition 24). □

Consider the Fourier transform of the indicator function of $S_k \subset S_n$:

$$\delta_{S_k}(\sigma) = \begin{cases} 1 & \text{if } \sigma(j) = j \text{ for } j \in \{k+1, \ldots, n\} \\ 0 & \text{otherwise} \end{cases}.$$

We now apply the branching rule $n-k$ times to the indicator function δ_{S_k}. Since δ_{S_k} is supported on $S_k \subset S_n$, the Fourier transform of δ_{S_k} at the irreducible ρ_λ can be written as a direct sum of Fourier coefficient matrices at the irreducibles which appear in the $n-k$th level of the branching tree corresponding to λ. We thus have:

$$\left[\hat{\delta}_{S_k}\right]_{\rho_\lambda} = \bigoplus_{\mu \in \mathcal{T}^\lambda_{n-k}} \textsc{Restrict}^{n-k}[[\widehat{\delta}_{S_k}]_{\rho_\mu}],$$

where $\textsc{Restrict}^{n-k}[\widehat{\delta}_{S_k}]$ refers to iterated restrictions to the Fourier transform of the indicator function. Furthermore, since the restriction of δ_{S_k} to the subgroup S_k is a constant function, we see that all of the nontrivial irreducible summands are zero (since the Fourier transform of a constant function is zero at all nontrivial terms) and that the trivial terms are exactly $k!$. Because the trivial representation is one-dimensional, only a subset of the diagonal elements of $\left[\hat{\delta}_{S_k}\right]_{\rho_\lambda}$ can be nonzero.

Algorithmically we can construct the Fourier transform of δ_{S_k} at λ by enumerating all of the branching sequences for λ and setting the (j,j) diagonal element of $\left[\hat{\delta}_{S_k}\right]_{\rho_\lambda}$ to be $k!$ if the corresponding jth branching sequence contains the partition (k). Alternatively, we can state the procedure in terms of standard tableaux. First, we define a restriction operation on a standard tableau t.

Definition 43. Given a standard tableau t with n boxes and a positive integer $k < n$, we define the *restriction* of t to S_k (denoted by $\textsc{Restrict}^{n-k}[t]$) to be the standard tableau t after removing boxes containing labels $k+1$, $\ldots$, n.

To construct the Fourier transform of δ_{S_k} at λ, we iterate through the standard tableaux of shape λ, and set the (j,j) diagonal element of $\left[\hat{\delta}_{S_k}\right]_{\rho_\lambda}$ to be $k!$ if the restriction of the jth tableau to S_k, $\textsc{Restrict}^{n-k}[t_j]$, takes the form $\begin{array}{|c|c|c|c|c|}\hline 1 & 2 & 3 & \cdots & k \\ \hline \end{array}$. See Algorithm 6.3.

Example 44. *We compute* $\left[\hat{\delta}_{S_2}\right]_{(3,2)}$ *as an example. The branching sequences for* $\lambda = (3,2)$ *are:*

$$\begin{array}{|c|c|c|}\hline 1 & 3 & 5 \\ \hline 2 & 4 \\ \cline{1-2}\end{array} \quad \longleftrightarrow \quad [(1) \to (1,1) \to (2,1) \to (2,2) \to (3,2)],$$

$$\begin{array}{|c|c|c|}\hline 1 & 2 & 5 \\ \hline 3 & 4 \\ \cline{1-2}\end{array} \quad \longleftrightarrow \quad [(1) \to (2) \to (2,1) \to (2,2) \to (3,2)],$$

$$\begin{array}{|c|c|c|}\hline 1 & 3 & 4 \\ \hline 2 & 5 \\ \cline{1-2}\end{array} \quad \longleftrightarrow \quad [(1) \to (1,1) \to (2,1) \to (3,1) \to (3,2)],$$

$$\begin{array}{|c|c|c|}\hline 1 & 2 & 4 \\ \hline 3 & 5 \\ \cline{1-2}\end{array} \quad \longleftrightarrow \quad [(1) \to (2) \to (2,1) \to (3,1) \to (3,2)],$$

$$\begin{array}{|c|c|c|}\hline 1 & 2 & 3 \\ \hline 4 & 5 \\ \cline{1-2}\end{array} \quad \longleftrightarrow \quad [(1) \to (2) \to (3) \to (3,1) \to (3,2)].$$

Algorithm 6.3: Pseudocode for computing the Fourier transform of the indicator function of $S_k \subset S_n$ at the partition λ. Input: A partition λ of n, and any $0 < k <= n$. Output: The Fourier transform of the indicator function of $S_k \subset S_n$ at the irreducible representation ρ_λ.

S_k-INDICATOR(k, λ):

$\left[\hat{\delta}_{S_k}\right]_{\rho_\lambda} \leftarrow \mathbf{0}_{d_\lambda \times d_\lambda}$;
foreach *standard tableaux* t *of shape* λ **do**
  if RESTRICT$^{n-k}[t] =$ $\boxed{1}\boxed{2}\boxed{3}\boxed{\cdots}\boxed{k}$ **then**
    $\left[\hat{\delta}_{S_k}\right]_{\rho_\lambda}(t,t) \leftarrow k!$;
  end
end
return $\left(\left[\hat{\delta}_{S_k}\right]_{\rho_\lambda}\right)$;

Since there are only three sequences which contain the partition (2)*, only those three basis elements have nonzero entries. And finally, noting that the appropriate normalization constant here is simply* $|S_2| = 2! = 2$*, we see that:*

$$\left[\hat{\delta}_{S_2}\right]_{(3,2)} = \left[\begin{array}{c|ccccc} & \begin{array}{|c|c|c|}\hline 1&3&5\\\hline 2&4\\\cline{1-2}\end{array} & \begin{array}{|c|c|c|}\hline \mathbf{1}&\mathbf{2}&5\\\hline 3&4\\\cline{1-2}\end{array} & \begin{array}{|c|c|c|}\hline 1&3&4\\\hline 2&5\\\cline{1-2}\end{array} & \begin{array}{|c|c|c|}\hline \mathbf{1}&\mathbf{2}&4\\\hline 3&5\\\cline{1-2}\end{array} & \begin{array}{|c|c|c|}\hline \mathbf{1}&\mathbf{2}&3\\\hline 4&5\\\cline{1-2}\end{array} \\ \hline \begin{array}{|c|c|c|}\hline 1&3&5\\\hline 2&4\\\cline{1-2}\end{array} & 0 & 0 & 0 & 0 & 0 \\ \begin{array}{|c|c|c|}\hline \mathbf{1}&\mathbf{2}&5\\\hline 3&4\\\cline{1-2}\end{array} & 0 & 2 & 0 & 0 & 0 \\ \begin{array}{|c|c|c|}\hline 1&3&4\\\hline 2&5\\\cline{1-2}\end{array} & 0 & 0 & 0 & 0 & 0 \\ \begin{array}{|c|c|c|}\hline \mathbf{1}&\mathbf{2}&4\\\hline 3&5\\\cline{1-2}\end{array} & 0 & 0 & 0 & 2 & 0 \\ \begin{array}{|c|c|c|}\hline \mathbf{1}&\mathbf{2}&3\\\hline 4&5\\\cline{1-2}\end{array} & 0 & 0 & 0 & 0 & 2 \end{array}\right].$$

We conclude with a comment on sparsity. It is clear from Algorithm 6.3 that the coefficient matrices of $\left[\hat{\delta}_{S_k}\right]_\lambda$ are all diagonal matrices with at most $O(d_\lambda)$ nonzero entries. In fact, we can sometimes show that a given model has $O(1)$ nonzero entries.

Consider, for example, the indicator function $\delta_{S_{n-1}}$, which is nonzero only at the first two partitions, (n), and $(n-1,1)$. The zeroth-order term is, $\left[\hat{\delta}_{S_{n-1}}\right]_{\rho_{(n)}} = (n-1)!$. The first-order Fourier coefficient matrix, $\left[\hat{\delta}_{S_{n-1}}\right]_{\rho_{(n-1,1)}}$, is a matrix of all zeroes except for a single element on the diagonal, $\left[\hat{\delta}_{S_{n-1}}\right]_{\rho_{(n-1,1)}}(t,t)$, where $t = \begin{array}{|c|c|c|c|}\hline 1&2&3&\cdots\\\hline n\\\cline{1-1}\end{array}$, which takes on the value $(n-1)!$.

6.6 CLAUSEN'S FAST FOURIER TRANSFORM (FFT) ALGORITHM

As a final application of the branching rule, we go beyond indicator functions of subgroups in this section and consider computing the Fourier transform of arbitrary functions on S_n by introducing *Clausen's Fast Fourier transform* algorithm. Given a function $h : S_n \to \mathbb{R}$, Clausen's FFT computes all Fourier coefficient matrices with respect to the Gel'fand-Tsetlin basis in $O(n^3 n!)$ time. While much faster than the naive $O((n!)^2)$ algorithm, Clausen's FFT is of course too slow to be of practical interest in most cases.

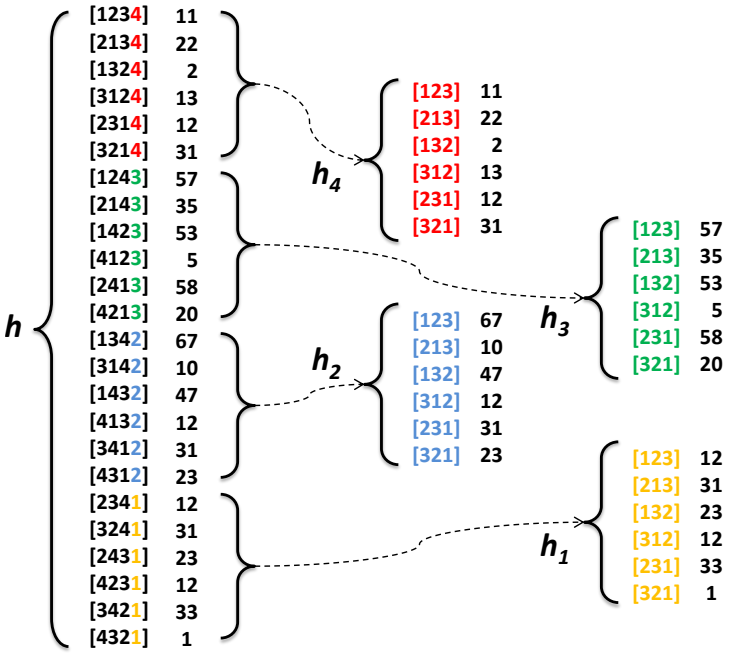


Figure 14: An illustration of the relationship between a function h defined on S_n and functions h_i defined over S_{n-1} (where $n = 4$ in this example, and numbers in the right columns denote function values). Compare this diagram to Figure 4 in Chapter 2 in which the same decomposition is used to recursively enumerate permutations.

However, the ideas that underlie the algorithm are identical to some of the ideas that we use in later chapters to form bandlimited probabilistic models in the Fourier domain and to factor distributions into independent distributions.

The FFT algorithm works by decomposing the Fourier transform problem for a function on S_n into n smaller Fourier transforms defined over S_{n-1}-cosets by using RESTRICT and EMBED, operations. In particular, we can write a function h as a sum of the following functions (indexed by $i = 1, \ldots, n$) which are defined over S_{n-1}.

$$h_i \equiv \text{RESTRICT}\left[\text{SHIFT}[h, \epsilon, \pi_i^{-1}]\right], \tag{6.5}$$

where $\pi_i = (i, i+1, \ldots, n)$. While Equation 6.5 may look complicated, each h_i simply corresponds to the collection of permutations for which item n is constrained to map to i, but shifted to the subgroup $S_{n-1} \subset S_n$. For intuition, see Figure 14, which illustrates that if h is represented as a single length $n!$ array of numbers, then the h_i correspond to contiguous segments of h with length $(n-1)!$. Note that this is the same decomposition that is used in Chapter 2 for enumerating permutations.

The function h can therefore be written as a sum of the h_i functions (after applying the inverse shift to each function and embedding it back into S_n):

$$h = \sum_{i=1}^{n} \textsc{Shift}\left[\textsc{Embed}[h_i], \epsilon, \pi_i\right]. \tag{6.6}$$

We show below that given the Fourier coefficients of the h_i, one can formulate a "Fourier domain" counterpart of Equation 6.6 to obtain the Fourier coefficients of h. The FFT then computes the Fourier transform of h recursively by first computing the Fourier transforms of the functions h_i on S_{n-1}, which are then decomposed into sums of functions on S_{n-2}, and so on. In the following, we discuss how to perform the shift, restrict and embed operations with respect to Fourier coefficient matrices.

6.6.1 *Embedding and Restriction operations in the Fourier domain*

The first two operations, Embed and Restrict, both rely on the branching rule. The pseudocode for implementing each of the below operations are given in Algorithms 6.4 and 6.5.

Theorem 45.

1. *Given the Fourier transform of a function $h : S_{n-1} \to \mathbb{R}$, the Fourier transform of the function $\textsc{Embed}[h] : S_n \to \mathbb{R}$ is given by:*

FourierEmbed:

$$\textsc{Embed}[\widehat{h_\lambda}](\sigma) = \bigoplus_{\lambda^-} \widehat{h}_{\rho_{\lambda^-}}, \tag{6.7}$$

where λ^- indexes over the partitions of $n-1$ which can be obtained from the Ferrers diagram of λ by removing a single box.

2. *Similarly, given the Fourier transform of a function $h : S_n \to \mathbb{R}$, the Fourier transform of the function $\textsc{Restrict}[h] : S_{n-1} \to \mathbb{R}$ is given by:*

FourierRestrict:

$$\textsc{Restrict}[\widehat{h}_{\rho_{\lambda^-}}] = \sum_{\lambda} \frac{d_\lambda}{n d_{\lambda^-}} \widehat{h}_{\rho_\lambda}[\lambda^-], \tag{6.8}$$

where λ indexes over the partitions of n which can be obtained from the Ferrers diagram of λ^- by adding a single box.

Proof. Equation 6.7 is simply a restatement of Corollary 42 in terms of the embedding operator (Equation 3.11).

We now establish Equation 6.8. Let h be a function on S_n. To derive the expression for the restriction of h, we will expand the function $\textsc{Restrict}[h]$ first using the inverse Fourier transform as defined for functions on S_n,

Algorithm 6.4: Algorithm for computing the embedding of a function. Input: $\{\widehat{h}_\rho\}_{\rho\in\Lambda_{n-1}}$, the Fourier coefficient matrices of a function $h : S_{n-1} \to \mathbb{R}$, Output: $\{\widehat{h'}_\rho\}_{\rho\in\Lambda_n}$, the Fourier coefficient matrices of $\textsc{Embed}[h] : S_n \to \mathbb{R}$.

```
FourierEmbed({ĥ_ρ}_{ρ∈Λ_{n-1}}):

    foreach partition λ of n do
        cnt ← 0 ;
        Initialize ĥ'_{ρ_λ} ← 0_{d_λ × d_λ} ;
        foreach partition λ⁻ obtained by removing a single box from λ do
            d ← numrows(ĥ_{ρ_λ}) ;
            ĥ'_{ρ_λ}(cnt : (cnt + d − 1), cnt : (cnt + d − 1)) ← ĥ_{ρ_{λ⁻}};
            cnt ← cnt + d;
        end
    end
    return ({ĥ'_ρ}_{ρ∈Λ_n});
```

then manipulate the expression to look as if it were the inverse Fourier transform for a function on S_{n-1}. Consider any $\sigma \in S_{n-1} \subset S_n$.

$$
\begin{aligned}
\textsc{Restrict}[h](\sigma) &= h([\sigma(1)\ \dots\ \sigma(n-1)\ n]), \\
&\qquad \text{(Equation 3.12)} \\
&= \frac{1}{n!}\sum_{\lambda} d_\lambda \mathrm{Tr}\left[\widehat{h}_{\rho_\lambda}^{T}\cdot\rho_\lambda([\sigma(1)\ \dots\ \sigma(n-1)\ n])\right], \\
&\qquad \text{(Apply inverse Fourier transform, Equation 5.6)} \\
&= \frac{1}{n!}\sum_{\lambda} d_\lambda \mathrm{Tr}\left[\widehat{h}_{\rho_\lambda}^{T}\cdot\left(\bigoplus_{\lambda^-}\rho_{\lambda^-}(\sigma)\right)\right], \\
&\qquad \text{(Branching rule, Equation 6.1)} \\
&= \frac{1}{n!}\sum_{\lambda} d_\lambda \sum_{\lambda^-} \mathrm{Tr}\left[\widehat{h}_{\rho_\lambda}[\lambda^-]^{T}\cdot\rho_{\lambda^-}(\sigma)\right], \\
&\qquad \text{(where } \widehat{h}_{\rho_\lambda}[\lambda^-] \text{ refers to the block of } \widehat{h}_{\rho_\lambda} \\
&\qquad \text{corresponding to the } \lambda^- \text{ term in the branching} \\
&\qquad \text{rule decomposition)} \\
&= \frac{1}{(n-1)!}\sum_{\lambda^-} \mathrm{Tr}\left[\left(\sum_{\lambda}\frac{d_\lambda}{n d_{\lambda^-}}\widehat{h}_{\rho_\lambda}[\lambda^-]\right)^{T}\cdot\rho_{\lambda^-}(\sigma)\right]. \\
&\qquad \text{(rearranging)}
\end{aligned}
$$

□

6.6.2 *Left cycling in the Fourier domain*

We now turn out attention to the problem of shifting a distribution by left-multiplying by a single contiguous cycle of the form $(i, i+1, \dots, n)$. Even though this problem falls as a special case of the shifting problem which we discuss more generally in Chapter 8, we shall refer to this contiguous cycle case as the CycleLeft operation as alternative (more

Algorithm 6.5: Algorithm for computing the restriction of a function. Input: $\{\widehat{h}_\rho\}_{\rho\in\Lambda_n}$, the Fourier coefficient matrices of a function $h : S_n \to \mathbb{R}$, Output: $\{\widehat{h'}_\rho\}_{\rho\in\Lambda_{n-1}}$, the Fourier coefficient matrices of $\textsc{Restrict}[h] : S_{n-1} \to \mathbb{R}$.

$\textsc{FourierRestrict}(\{\widehat{h}_\rho\}_{\rho\in\Lambda_n})$:

foreach *partition* λ^- *of* $n-1$ **do**
  Initialize $\widehat{h}'_{\rho_{\lambda^-}} \leftarrow \mathbf{0}_{d_{\lambda^-}\times d_{\lambda^-}}$;
end
foreach *partition* λ *of* n **do**
  $\mathsf{cnt} \leftarrow 0$;
  foreach *partition* λ^- *obtained by removing a single box from* λ **do**
    $\widehat{h}'_{\rho_{\lambda^-}} \leftarrow \widehat{h}'_{\rho_{\lambda^-}} + \frac{d_\lambda}{n d_{\lambda^-}} \cdot \widehat{h}_{\rho_\lambda}(\mathsf{cnt} : (\mathsf{cnt} + d_{\lambda^-} - 1), \mathsf{cnt} : (\mathsf{cnt} + d_{\lambda^-} + 1))$;
    $\mathsf{cnt} \leftarrow \mathsf{cnt} + d_{\lambda^-}$;
  end
end
return $(\{\widehat{h'}_\rho\}_{\rho\in\Lambda_{n-1}})$;

efficient) algorithms can be applied. Below, we argue that given the Fourier coefficient matrices of a distribution h, the Fourier coefficient matrices of $\textsc{CycleLeft}[h, (i, i+1, \ldots, n)]$ can be computed by left multiplying by at most $O(n)$ representation matrices evaluated at adjacent transpositions.

Proposition 46.

CycleLeft:

$$\textsc{CycleLeft}[\hat{h}_{\rho_\lambda}, (i, i+1, \ldots, n)] = \left(\prod_{j=i}^{n-1} \rho_\lambda((j, j+1))\right) \cdot \widehat{h}_{\rho_\lambda}. \quad (6.9)$$

Proof.

$$\begin{aligned}
\textsc{Shift}[\hat{h}_{\rho_\lambda}, \epsilon, (i, i+1, \ldots, n)] &= \sum_\sigma \textsc{Shift}[h, \epsilon, (i, i+1, \ldots, n)](\sigma)\rho_\lambda(\sigma), \\
&\quad \text{(Definition 24)} \\
&= \sum_{\sigma'} h((i, i+1, \ldots, n)^{-1}\sigma) \cdot \rho_\lambda(\sigma), \\
&\quad \text{(set } \sigma' = (i, i+1, \ldots, n)\sigma) \\
&= \rho_\lambda((i, i+1, \ldots, n)) \cdot \left(\sum_{\sigma'} h(\sigma')\rho_\lambda(\sigma')\right), \\
&\quad \text{(Definition 20)} \\
&= \left(\prod_{j=i}^{n-1} \rho_\lambda((j, j+1))\right) \cdot \widehat{h}_{\rho_\lambda},
\end{aligned}$$

where the last line follows by noticing that contiguous cycles factor as a product of at most $O(n)$ adjacent transpositions:

$$(i, i+1, \ldots, n) = (i, i+1)(i+1, i+2)\cdots(n-1, n).$$

□

Algorithm 6.6: Algorithm for Clausen's FFT (Fast Fourier transform). Input: A positive integer n and a function h defined on S_n. h is represented here as an $n!$ dimensional contiguous array and so the h_i correspond to contiguous $(n-1)!$ sized subsegments of h. Output: $\{\hat{h}_\rho\}_{\rho\in\Lambda_n}$, the Fourier coefficient matrices of h.

FFT(n, h):

if $n == 1$ **then**
  Initialize $\widehat{h}_{\rho_{(n)}} \leftarrow h([1])$;
  return $\left(\{\widehat{h}_\rho\}_{\rho\in\Lambda_1}\right)$;
end
foreach *partition* λ *of* n **do**
  Initialize $\widehat{h}_{\rho_\lambda} \leftarrow \mathbf{0}_{d_\lambda \times d_\lambda}$;
end
for $i = 1, \ldots, n$ **do**
  $h_i \leftarrow \textsc{Restrict}[\textsc{Shift}[h, \epsilon, \pi_i^{-1}]]$;
  $\{[\widehat{h_i}]_{\rho\in\Lambda_{n-1}}\} \leftarrow \mathrm{FFT}(n-1, h_i)$;
  $\{[\widehat{h'_i}]_{\rho\in\Lambda_n}\} \leftarrow \textsc{CycleLeft}(\textsc{Embed}(\{[\widehat{h_i}]_{\rho\in\Lambda_{n-1}}\}), (i, i+1, \ldots, n))$;
  foreach *partition* λ *of* n **do**
    $\widehat{h}_{\rho_\lambda} \leftarrow \widehat{h}_{\rho_\lambda} + [\widehat{h'_i}]_{\rho_\lambda}$;
  end
end
return $\left(\{\hat{h}_\rho\}_{\rho\in\Lambda_n}\right)$;

Algorithm 6.7: Algorithm for the inverse FFT. Input: $\{\hat{h}_\rho\}_{\rho\in\Lambda}$, the Fourier coefficient matrices of h. Output: A function h defined on S_n.

IFFT($n, \{\widehat{h}_\rho\}_{\rho\in\Lambda}$):

if $n == 1$ **then**
  return $\left(\widehat{h}_{\rho_{(n)}}\right)$;
end
Initialize $h \leftarrow \mathbf{0}_{n!\times 1}$;
for $i = n, n-1, \ldots, 1$ **do**
  $\{[\widehat{h_i}]_\rho\}_{\rho\in\Lambda_{n-1}} \leftarrow \textsc{Restrict}\left[\textsc{CycleLeft}[\{\hat{h}_\rho\}_{\rho\in\Lambda_n}, (n, n-1, \ldots, i)]\right]$;
  $h_i \leftarrow \mathrm{IFFT}(n-1, \{[\widehat{h_i}]_\rho\}_{\rho\in\Lambda_{n-1}})$;
  $h((n-i)(n-1)! : (n-i+1)(n-1)!) \leftarrow h_i$;
end
return (h);

Note that cycling operation of Equation 6.9 can be efficiently implemented by exploiting the sparsity of the representation matrices evaluated at adjacent transpositions. In particular, since each column has of $\rho_\lambda((j, j+1))$ has no more than two nonzero entries, multiplication by $\rho_\lambda((j, j+1))$ can be accomplished in $O(d_\lambda^2)$ operations instead of $O(d_\lambda^3)$ operations.

6.6.3 *The Fast Fourier Transform*

We finally state the recursive Clausen FFT in Algorithm 6.6, which first computes the Fourier transforms of each h_i, then pieces the Fourier transform of h together via Equation 6.6, which can be performed directly in the Fourier domain using Algorithm 6.4 and Equation 6.9. The running

time of Clausen's FFT is known to be $O(n^3 n!)$ [75]. We note that there are faster known variants of the FFT algorithm with running time $O(n^2 n!)$ [94] which fall outside the scope of this thesis. Algorithm 6.7 provides the corresponding Clausen iFFT (inverse Fast Fourier transform).

6.7 SUMMARY

In this chapter, we have presented a number of algorithms for (1), constructing explicit representation matrices, (2), constructing the Fourier coefficient matrices of S_k-indicator functions, and (3), computing Fourier and inverse Fast Fourier transforms of general functions defined over the symmetric group. In each case, the algorithms defined in this chapter are stated with respect to a combinatorial abstraction known as the standard Young tableaux which we motivated as arising through recursive applications of the branching rule. As we will see in the next chapter, directly computing the Fourier transform using the FFT is, in most cases, intractable and unnecessary. However, the same insights that form the basis for the Clausen FFT will allow us to efficiently construct low-order Fourier coefficient matrices of useful probabilistic models.

7

COMMON PROBABILISTIC MODELS AND THEIR FOURIER TRANSFORMS

IF the FFT has a running time complexity $O(n^3 n!)$, how can anyone hope to accomplish anything useful in a reasonable amount of time? In this chapter, we we discuss a collection of useful models for which we *can* efficiently compute low-order Fourier coefficients or even provide a closed-form expression. See Table 4 for a summary of the various models covered in this section.

We consider both *mixing* and *observation* models for various applications. In multiobject tracking, a mixing model might account for the fact that two tracks may have swapped identities with some probability. Or in card shuffling, a mixing model might reflect that a card has been inserted somewhere into the deck. In multiobject tracking, an observation model might tell us that Alice is at some track with probability one. Or it might reflect the fact that some subset of identities occupies some subset of tracks with no order information, as in the case of the *bluetooth model*. In ranking applications, an observation model might, for example, reflect that some object is ranked higher than, or preferred over some other object. Recall from Chapter 3 that one convolves a prior distribution by a mixing model and multiplies (using Bayes rule) a prior distribution by an observation model.

This chapter is divided into three parts, each describing a different approach to computing the Fourier coefficients of a model, with some being simpler or more efficient to implement in certain situations than others. In *direct constructions*, we naively apply the definition of the Fourier transform to obtain the Fourier coefficients of some model. In *marginal based constructions*, we first compute the low-order 'marginals' of some probabilistic model, then project the result onto the irreducible Fourier basis. Finally, in *coset-based constructions*, we introduce a family of 'atomic' indicator functions of subgroups of the form $S_k \subset S_n$ which are then combined using scale/shift/convolution operations to form more complex models. As we discuss in Chapter 17, there also remains the open possibility of learning models *directly* in the Fourier domain. For the sake of succinctness, many of the results in this chapter will be stated without proof.

7.1 DIRECT CONSTRUCTIONS IN THE FOURIER DOMAIN

In some applications we are fortunate enough to have a model that can be "directly" transformed efficiently using the definition of the Fourier transform (Definition 24). We provide two examples.

Mixing Model	Example Semantics	Relevant Subgroup
Pairwise	Identity confusion at tracks 1 and 2	S_2
k-subset	Identity confusion at tracks in $\{1,2,4,6\}$	S_k
Insertion	Insert top card somewhere in the deck	n/a

Observation Model	Example Semantics	Relevant Subgroup
Single track	Alice is at Track 1	S_{n-1}
Multitrack	Alice is at Track 1, Bob is at Track 2, etc.	S_{n-k}
Bluetooth	The girls occupy tracks $\{1,2,6,8\}$	$S_k \times S_{n-k}$
Pairwise ranking	Apples are better than oranges	S_{n-2}

Table 4: Several useful types of mixing and observation models are summarized in the above table. In many of these cases, computing the appropriate Fourier transform reduces to computing the Fourier transform of the indicator function of some related subgroup of S_n, and so we also mention the relevant subgroup in the second column. In the third column we provide an example illustrating the semantics of each model.

7.1.1 *Pairwise mixing models*

The simplest mixing model for identity management assumes that with probability p, nothing happens, and that with probability $(1-p)$, the identities for tracks i and j are swapped. The probability distribution for the *pairwise mixing model* is therefore:

$$q_{ij}(\pi) = \begin{cases} p & \text{if } \pi = \epsilon \\ 1-p & \text{if } \pi = (i,j) \\ 0 & \text{otherwise} \end{cases} . \tag{7.1}$$

Since q_{ij} is such a sparse distribution (in the sense that $q_{ij}(\pi) = 0$ for most π), it is possible to directly compute $\widehat{q}_{ij}$ using Definition 24:

$$\left[\widehat{q}_{ij}\right]_{\rho_\lambda} = pI + (1-p)\rho_\lambda((i,j)),$$

where I refers to the $d_\lambda \times d_\lambda$ identity matrix (since any representation must map the identity element ϵ to an identity matrix), and $\rho_\lambda((i,j))$ is the irreducible representation matrix ρ_λ evaluated at the transposition (i,j) (which can be computed using the algorithms from the previous chapter.

7.1.2 *Insertion mixing models*

As another example, we can consider the *insertion mixing model* (also called the *top-in shuffle* [29]) in which we take the top card in some deck of n cards, and with uniform probability, insert it *somewhere* in the deck, preserving all other original relative orderings. Insertions can be useful in ranking applications where we might wish to add a new item into consideration

without disturbing the marginal probabilities over relative rankings of existing items. The distribution for the insertion mixing model is given by:

$$q^{\text{insertion}}(\pi) = \begin{cases} \frac{1}{n} & \text{if } \pi \text{ is a cycle of the form } (j, j-1, \ldots, 1) \text{ for some } j \in \{1, \ldots, n\} \\ 0 & \text{otherwise} \end{cases}. \tag{7.2}$$

Since the insertion mixing model is supported on n permutations, it is again simple to directly construct the Fourier transform from the definition. We have:

$$\widehat{q}^{\text{insertion}}_{\rho_\lambda} = \frac{1}{n} \sum_{j=1}^{n} \rho_\lambda(j, j-1, \ldots, 1).$$

Going beyond the simple case of inserting a single card at a time, in Chapter 13, we introduce a generalization of the insertion mixing model, called the *biased riffle shuffle*, in which one inserts (interleaves) a collection of cards jointly into a larger deck.

7.2 MARGINAL BASED CONSTRUCTIONS

In *marginal based constructions*, we first compute the low-order 'marginals'[1] of some probabilistic model, then project the result onto the irreducible Fourier basis. Thus given a function $f : S_n \to \mathbb{R}$, we compute, for example, the first-order marginals $\hat{f}_{\tau_{(n-1,1)}}$, and conjugate by an intertwining operator (Equation 5.10) to obtain the Fourier coefficients at (n) and $(n-1, 1)$. Sometimes when the Fourier transform of f is provably non-zero *only* at low-order terms, a marginal based construction might be the easiest method to obtain Fourier coefficients.

7.2.1 *Color histogram observation models*

The simplest model assumes that we can get observations of the form: 'track ℓ is color k' (which is essentially the model considered by [80]). The probability of seeing color k at track ℓ given data association σ is

$$L(\sigma) = h(z_\ell = k|\sigma) = \alpha_{\sigma^{-1}(\ell),k}, \tag{7.3}$$

where $\sum_k \alpha_{\sigma^{-1}(\ell),k} = 1$. For each identity, the likelihood $L(\sigma) = h(z_\ell = k|\sigma)$ depends, for example, on a histogram over all possible colors. If the number of possible colors is K, then the likelihood model can be specified by an $n \times K$ matrix of probabilities. For example,

$$\alpha_{\sigma(\ell),k} = \left[\begin{array}{c|cccc} & k = \text{Red} & k = \text{Orange} & k = \text{Yellow} & k = \text{Green} \\ \hline \sigma(\text{Alice}) = \ell & 1/2 & 1/4 & 1/4 & 0 \\ \sigma(\text{Bob}) = \ell & 1/4 & 0 & 0 & 3/4 \\ \sigma(\text{Cathy}) = \ell & 0 & 1/2 & 1/2 & 0 \end{array}\right]. \tag{7.4}$$

1 The word 'marginals' is technically appropriate only when the function in question is a legal probability distribution (as opposed to likelihood functions, for example), however we use it to refer to similar summary statistics for general functions.

Since the observation model only depends on a single identity, the first-order terms of the Fourier transform suffice to describe the likelihood exactly. To compute the first-order Fourier coefficients at irreducibles, we proceed by computing the first-order Fourier coefficients at the first-order permutation representation (the first-order "marginals"), then transforming to irreducible coefficients. The Fourier transform of the likelihood at the first-order permutation representation is given by:

$$\begin{aligned}\left[\widehat{L}_{\tau_{(n-1,1)}}\right]_{ij} &= \sum_{\{\sigma:\sigma(j)=i\}} h\left(z_\ell = k|\sigma\right) \\ &= \sum_{\{\sigma:\sigma(j)=i\}} \alpha_{\sigma^{-1}(\ell),k}.\end{aligned}$$

To compute the ij-term, there are two cases to consider.

1. If $i = \ell$ (that is, if Track i is the same as the track that was observed), then the coefficient $\widehat{L}_{ij}$ is proportional to the probability that Identity j is color k.

$$\widehat{L}_{ij} = \sum_{\{\sigma:\sigma(j)=i\}} \alpha_{j,k} = (n-1)! \cdot \alpha_{j,k}. \tag{7.5}$$

2. If, on the other hand, $i \neq \ell$ (Track i is not the observed track)), then the coefficient $\hat{L}_{ij}$ is proportional to the sum over

$$\widehat{L}_{ij} = \sum_{\{\sigma:\sigma(j)=i\}} \alpha_{\sigma^{-1}(\ell),k}, \tag{7.6}$$

$$= \sum_{m\neq j} \sum_{\{\sigma:\sigma(j)=i \text{ and } \sigma(m)=\ell\}} \alpha_{\sigma^{-1}(\ell),k}, \tag{7.7}$$

$$= \sum_{m\neq j} (n-2)! \cdot \alpha_{m,k}. \tag{7.8}$$

Example 47. *We will compute the first-order marginals of the likelihood function on S_3 which arises from observing a "Red blob at Track 1". Plugging the values from the "Red" column of the α matrix (Equation 7.4) into Equation 7.5 and 7.8 yields the following matrix of first-order coefficients (at the $\tau_{(n-1,1)}$ permutation representation):*

$$\left[\hat{L}_{(n-1,1)}\right]_{ij} = \left[\begin{array}{c|ccc} & \textit{Track 1} & \textit{Track 2} & \textit{Track 3} \\ \hline \textit{Alice} & 1/4 & 1/2 & 3/4 \\ \textit{Bob} & 1/4 & 1/2 & 3/4 \\ \textit{Cathy} & 1 & 1/2 & 0 \end{array}\right].$$

The corresponding coefficients at the irreducible representations are:

$$\hat{L}_{(3)} = 1.5, \qquad \hat{L}_{(2,1)} = \begin{bmatrix} 0 & 0 \\ -\sqrt{3}/4 & -3/4 \end{bmatrix}, \qquad \hat{L}_{(1,1,1)} = 0.$$

7.2.2 *Unordered subset observation models (version 1)*

We sometimes receive measurements in the form of unordered lists. For example, the *bluetooth model* is the likelihood function that arises if tracks $\{1,\ldots,k\}$ are within range of a bluetooth detector and we receive a measurement that identities $\{1,\ldots,k\}$ are in range. In sports, we might observe that the first k tracks belong to the red team and that the last $n-k$ tracks belong to the blue team. And finally, in *approval voting*, one specifies a subset of approved candidates rather than, for example, picking a single favorite.

We consider two options for bluetooth-type situations. In the first option, we allow for some error-tolerance by setting the likelihood to be proportional to the number of tracks that are correctly returned in the measurement:

$$h^{\mathrm{bluetooth}}(z_{\{t_1,\ldots,t_k\}} = \{i_1,\ldots,i_k\}|\sigma) \propto |\{t_1,\ldots,t_k\} \cap \sigma(\{i_1,\ldots,i_k\})| + c, \tag{7.9}$$

where c is a constant term allowing for noisy observations. Our first bluetooth model can be expressed using only first order terms (intuitively because each track makes a linear contribution) and thus $\hat{h}^{\mathrm{bluetooth}}_{\rho_\lambda}$ is nonzero only at the first two partitions $\lambda = (n), (n-1,1)$. For simplicity, we consider the Fourier transform of the function: $h(\sigma) = |\sigma(\{1,\ldots,k\}) \cap \{1,\ldots,k\}|$. The first-order 'marginals' of h are covered in the following four cases:

- *($j \leqslant k$ and $i \leqslant k$)*:

$$L_{ij} = \sum_{\sigma:\sigma(j)=i} h(\sigma) = (k-1)^2(n-2)! + (n-1)!$$

- *($j \leqslant k$ and $i > k$)*:

$$L_{ij} = \sum_{\sigma:\sigma(j)=i} h(\sigma) = k(k-1)(n-2)!$$

- *($j > k$ and $i \leqslant k$)*:

$$L_{ij} = \sum_{\sigma:\sigma(j)=i} h(\sigma) = k(k-1)(n-2)!$$

- *($j > k$ and $i > k$)*:

$$L_{ij} = \sum_{\sigma:\sigma(j)=i} h(\sigma) = k^2(n-2)!$$

We discuss the second bluetooth-type model after discussing coset based constructions.

7.3 COSET BASED CONSTRUCTIONS

Most of the time, realistic models are not supported on only a handful of permutations. The approach we take now is to use a collection of 'primitive' functions to form more interesting models via scale/shift/convolution operations. In particular, we will make use of indicator functions of subsets of the form $S_{X,Y} \subset S_n$, where $X = (x_1, \ldots, x_k)$ and $Y = (y_1, \ldots, y_k)$ are ordered k-tuples with $\{x_1, \ldots, x_k\} \subset \{1, \ldots, n\}$, $\{y_1, \ldots, y_k\} \subset \{1, \ldots, n\}$ and no repetitions are allowed. $S_{X,Y}$ denotes the set of elements in S_n which are constrained to map each x_i to y_i:

$$S_{X,Y} \equiv \{\sigma \in S_n \ : \ \sigma(x_i) = y_i, \text{ for each i=1,...,k}\}. \tag{7.10}$$

The $S_{X,Y}$ can also be thought of as two-sided cosets associated with subgroups of the form $S_{n-k} \subset S_n$. For example, if $X = (1,2)$ and $Y = (3,4)$ with $n = 4$, then $S_{X,Y}$ is simply the set of all permutations that map $1 \mapsto 3$ and $2 \mapsto 4$. Thus, $S_{X,Y} = \{(1,3)(2,4), (1,3,2,4)\}$. Since $|X| = |Y| = k$, then $|S_{X,Y}| = (n-k)!$, and in the special case that $X = Y$, we have that $S_{X,Y}$ is in fact a subgroup isomorphic to S_{n-k}.

As we showed in Chapter 6, the Fourier transform of the indicator $\delta_{S_{X,Y}}$ takes a particularly simple (and low rank) form and can be efficiently computed. The method described in Chapter6 is based on the FFT which was described in the previous chapter and exploits the same structure of the symmetric group that is used by [80]. It is thus possible to understand why some observation models afford faster conditioning updates based on sparsity in Fourier domain.

The functions $\delta_{S_{X,Y}}$ can be viewed as a set of function primitives for constructing more complicated models via shift/scale/convolution operations in the Fourier domain. We now discuss the remaining models in Table 4 with the assumption that there exists some blackbox function which constructs the Fourier coefficients of the indicator function of (two-sided) cosets of the form $S_{X,Y} \subset S_n$ (see Algorithm 6.3 in Chap 6).

7.3.1 k-*subset mixing models*

In identity management, it is not always appropriate to mix only two people at once (as in Equation 7.1) and so we would like to formulate a mixing model which occurs over a subset of tracks, $X = \{t_1, \ldots, t_k\} \subset \{1, \ldots, n\}$. One way to 'mimic' the desired effect is to repeatedly draw pairs (i,j) from $\{t_1, \ldots, t_k\}$ and to convolve against the pairwise mixing models q_{ij}. A better alternative is to directly construct the Fourier coefficient matrices for the k-*subset mixing model*, in which we allow the tracks in X to be randomly permuted with uniform probability. In the following, $\bar{X}$ denotes some fixed ordering of the complement of X. For example, if $n = 5$, with $X = \{1,2,4\}$, then $\bar{X}$ is either $(3,5)$ or $(5,3)$. The k-subset mixing model is defined as:

$$q_X(\pi) = \begin{cases} \frac{1}{k!} & \text{if } \pi \in S_{\bar{X},\bar{X}} \subset S_n \\ 0 & \text{otherwise} \end{cases}. \tag{7.11}$$

Note that $S_{\bar{X},\bar{X}}$ is isomorphic to S_k and that the pairwise mixing model is the special case where $k = 2$. Intuitively, Equation 7.11 fixes all of the tracks outside of X and says that with uniform probability, the set of tracks in X experience some permutation of their respective identities. Equation 7.11 can also be written as $q_X(\pi) = \frac{1}{k!}\delta_{S_{\bar{X},\bar{X}}}(\pi)$, and thus the mixing model is simply a multiple of the indicator function of $S_{\bar{X},\bar{X}}$.

7.3.2 *Single/multitrack observation models*

In the *single track observation model* (used in [121, 115, 80], for example), we acquire an identity measurement z_j at track j. In the simplest version of the model, we write the likelihood function as:

$$h(z_i = j|\sigma) = \begin{cases} p & \text{if } \sigma(j) = i \\ \frac{1-p}{n-1} & \text{otherwise} \end{cases}, \tag{7.12}$$

where j ranges over all n possible identities. $h(z_i|\sigma)$ can also be written as a weighted sum of a uniform distribution U, and an indicator function:

$$h(z_i = j|\sigma) = \left(\frac{pn-1}{n-1}\right)\delta_{S_{j,i}}(\sigma) + \left(\frac{1-p}{n-1}\right)U(\sigma).$$

Equation 7.12 is useful when we receive measurements directly as single identities ("Alice is at Track 1 with such and such probability"). It is, however, far more common to receive lower level measurements that *depend* only upon a single identity, which we formalize with the following conditional independence assumption:

$$h(z_i|\sigma) = h(z_i|\sigma(j)). \tag{7.13}$$

For example, as in Equation 7.4, we might have a color histogram over each individual ("Alice loves to wear green") and observe a single color per timestep. Or we might acquire observations in the form of color histograms and choose to model a distribution over all possible color histograms. If for each identity j, $h(z_i|\sigma(j) = i) = \alpha_j$, then we can write the likelihood function as a weighted linear combination of n indicators,

$$L(\sigma) = h(z_i|\sigma) = \sum_j \alpha_j \delta_{S_{j,i}}(\sigma), \tag{7.14}$$

and by the linearity of the Fourier transform, we can obtain the Fourier coefficients of L:

$$\hat{L}_\lambda = \sum_j \alpha_j \left[\widehat{\delta_{S_{j,i}}}\right]_{\rho_\lambda}. \tag{7.15}$$

Finally, the single-track observations can be generalized to handle joint observations of multiple tracks at once with a higher-order model:

$$h(z_{(t_1,\ldots,t_k)} = (i_1,\ldots,i_k)|\sigma) = \begin{cases} p & \text{if } \sigma(i_\ell) = t_\ell \text{ for each } \ell \in \{1,\ldots,k\} \\ \frac{1-p}{\frac{n!}{(n-k)!}-1} & \text{otherwise} \end{cases}.$$

(7.16)

Unsurprisingly, while the Fourier coefficients of Equation 7.12 can be expressed exactly using first-order terms, the Fourier coefficients of the multi-track observation model, Equation 7.16, requires kth-order terms. It is important to note that joint multi-track observations are distinct from making k independent identity observations at the same timestep — we can handle the latter case by conditioning using a single-track observation model k times consecutively. Depending upon the specific sensor setup, one model may be more natural than the other.

7.3.3 *Unordered subset observation models (version 2)*

In contrast with the first bluetooth model (Equation 7.9), our second bluetooth-type model handles a higher-order form of measurement. Like the single/multi-track observation models, it says that with some probability we receive the correct unordered list, and with some probability, we receive some other list drawn uniformly at random:

$$h^{\text{bluetooth2}}(z_{\{t_1,\ldots,t_k\}} = \{i_1,\ldots,i_k\}|\sigma) = \begin{cases} p & \text{if } \sigma(\{i_1,\ldots,i_k\}) = \{t_1,\ldots,t_k\} \\ \frac{1-p}{\binom{n}{k}-1} & \text{otherwise} \end{cases}. \tag{7.17}$$

As with the single/multi-track observation models, the bluetooth model can be written as a weighted linear combination of a uniform distribution and the indicator function of an $S_k \times S_{n-k}$-coset (defined in Equation 12.1). To compute the Fourier transform of $h^{\text{bluetooth2}}$, it is enough to note that the indicator function of $S_k \times S_{n-k}$ can be thought of as a convolution of indicator functions of S_k and S_{n-k} in a certain sense. More precisely.

Proposition 48. *Let* $X = (1,\ldots,k)$ *and* $Y = (k+1,\ldots,n)$. *Then:* $\delta_{S_k \times S_{n-k}} = \delta_{S_{X,X}} * \delta_{S_{Y,Y}}$.

Invoking the convolution theorem (Proposition 51) shows that the Fourier coefficient matrices of $\delta_{S_k \times S_{n-k}}$ can be constructed by first computing the Fourier coefficients of $S_{X,X}$ and $S_{Y,Y}$, and pointwise multiplying corresponding coefficient matrices. We have:

$$\left[\widehat{\delta}_{S_k \times S_{n-k}}\right]_{\rho_\lambda} = \left[\widehat{\delta}_{S_{X,X}}\right]_{\rho_\lambda} \cdot \left[\widehat{\delta}_{S_{Y,Y}}\right]_{\rho_\lambda}, \text{ for all partitions } \lambda.$$

An interesting fact about the bluetooth model is that its Fourier terms are zero at all partitions with more than two rows.

Proposition 49. *Without loss of generality, assume that* $k \leqslant \frac{n}{2}$. *The Fourier transform of the bluetooth model,* $\hat{h}^{\text{bluetooth2}}_{\rho_\lambda}$ *is nonzero only at partitions of the form* $(n-s, s)$ *where* $s \leqslant k$.

7.3.4 *Pairwise ranking observation models*

Finally in the *pairwise ranking model*, we consider observations of the form "object j is ranked higher than object i" which can appear in various forms of voting and preference elicitation ("I like candidate x better than candidate y") or webpage/advertisement ranking. Here we think of σ as a mapping from objects to ranks. Our pairwise ranking model simply assigns higher probability to observations which agree with the ordering of i and j in σ.

$$h^{rank}(z_{k\ell}|\sigma) = \begin{cases} p & \text{if } \sigma(k) < \sigma(\ell) \\ 1-p & \text{otherwise} \end{cases} . \tag{7.18}$$

When $k = n-1$, $\ell = n$ and $p = 1$, we have:

$$\begin{aligned} h^{rank}(z_{k\ell}|\sigma) &= \begin{cases} 1 & \text{if } \sigma(n-1) < \sigma(n) \\ 0 & \text{otherwise} \end{cases} \\ &= \sum_{i<j} \delta_{S_{(n-1,n),(i,j)}}(\sigma). \end{aligned}$$

Perhaps unsurprisingly, pairwise ranking models can be sufficiently captured by first-order and second-order (ordered) Fourier coefficients [2].

Proposition 50. *The Fourier coefficients of the pairwise ranking model,* $\hat{h}^{rank}_{\rho_\lambda}$*, are nonzero only at three partitions:* $\lambda = (n)$, $(n-1,1)$, *and* $(n-2,1,1)$.

7.4 CONCLUSION

In this chapter, we have presented a number of useful probabilistic models that recur in a variety of applications, from identity management, to card shuffling, to ranking. In each of the models described in this chapter, we were able to provide methods for efficiently computing low-order terms of their respective Fourier transforms. In certain cases, we have even been able to establish that the Fourier transform of a model is zero beyond a certain frequency level.

We have found that the methods that we typically use to compute Fourier transforms generally can be categorized into three methodologies: the direct construction, the marginal based construction, and the coset based construction. Each of these methodologies is broadly applicable and we believe that they can be potentially useful for many more models which we have not considered in this thesis.

2 Additionally, $\hat{h}^{rank}_{\rho_{(n-1,1)}}$ and $\hat{h}^{rank}_{\rho_{(n-2,1,1)}}$ are known to be rank 1 matrices, a fact which can potentially be exploited for faster conditioning updates in practice.

8

PROBABILISTIC REASONING IN THE FOURIER DOMAIN

WHAT we have shown thus far in the thesis is that there is a principled method for compactly summarizing distributions over permutations based on the idea of bandlimiting — saving only the low-frequency terms of the Fourier transform of a function, which, as we have discussed in Chapter 5, is equivalent to maintaining a set of low-order marginal probabilities. We now turn to the problem of performing probabilistic inference using our compact summaries. One of the main advantages of viewing marginals as Fourier coefficients is that it provides a natural principle for formulating polynomial time approximate inference algorithms, which is to rewrite all inference related operations with respect to the Fourier domain, then to perform the Fourier domain operations ignoring high-order terms.

The idea of bandlimiting a distribution is ultimately moot, however, if it becomes necessary to transform back to the primal domain each time an inference operation is called. Naively, the Fourier Transform on S_n scales as $O((n!)^2)$, and even the fastest Fast Fourier Transforms for functions on S_n are no faster than $O(n^2 \cdot n!)$ (see [94] for example). To resolve this issue, we present a formulation of inference which operates solely in the Fourier domain, allowing us to avoid a costly transform. We begin by discussing exact inference in the Fourier domain, which is no more tractable than the original problem because there are $n!$ Fourier coefficients, but it will allow us to discuss the bandlimiting approximation in the next section.

In this chapter, we consider all of the probabilistic inference operations that were introduced in Chapter 3. We focus in particular on the hidden Markov model operations of prediction/rollup, and conditioning, which both have beautiful Fourier domain counterparts. While we have motivated both of these operations in the familiar context of hidden Markov models, they are fundamental and appear in many other settings. The assumption for the rest of this section is that the Fourier transforms of the transition and observation models are known. We have already discussed methods for obtaining these models in Chapter 7. The main results of this chapter (excluding the discussions about complexity) extend naturally to other finite groups besides S_n.

8.1 NORMALIZATION IN THE FOURIER DOMAIN

Perhaps the simplest probabilistic operation which can be implemented in the Fourier domain is normalization. While an exhaustive algorithm involves summing and elementwise multiplying by an $n!$-dimensional probability vector, one can accomplish the same operation with respect to Fourier coefficients by noticing that the normalization constant of a

Algorithm 8.1: Algorithm for normalizing a positive function in the Fourier domain. The input is $\{\hat{h}_\rho\}_{\rho\in\Lambda}$, the collection of irreducible Fourier coefficient matrices of h.

FOURIERNORMALIZE($\{\hat{h}_\rho\}_{\rho\in\Lambda}$):

$Z \leftarrow \widehat{h}_{(n)}$;
foreach $\rho \in \Lambda$ **do**
  $\widehat{h'}_\rho \leftarrow \frac{1}{Z}\widehat{h}_\rho$;
end
return $\{\widehat{h'}_\rho\}_{\rho\in\Lambda}$;

function $h : S_n \to \mathbb{R}$ is simply the 1×1 Fourier coefficient matrix of h at the trivial representation ($Z = \sum_\sigma h(\sigma) = \widehat{h}_{(n)}$). If $h' = h/Z$ is the normalized function, then we have, using linearity of the Fourier transform:

Fourier domain normalization operation:

$$\widehat{h'}_\rho \leftarrow \frac{1}{\widehat{h}_{(n)}}\widehat{h}_\rho, \text{ for any representation } \rho \text{ of } S_n. \tag{8.1}$$

Thus to normalize a function using Fourier coefficients, one simply divides each of the Fourier coefficient matrices of h by the trivial coefficient $\hat{h}_{(n)}$ (see the pseudocode for Fourier normalization in Algorithm 8.1).

8.2 PREDICTION/ROLLUP IN THE FOURIER DOMAIN

As discussed in Chapter 3, we focus our attention on random walk transition models (see Chapter 9 for a more detailed discussion about the limitations of the random walk assumption). We showed in particular (Equation 3.8) that, under the random walk assumption, the prediction/rollup operation can be written as a convolution of two functions on the symmetric group.

Just as with Fourier transforms on the real line, the Fourier coefficients of the convolution of two distributions f and g on groups can be obtained from the Fourier coefficients of f and g individually, using the *convolution theorem* (see also [29]), which guarantees that *the Fourier transform of a convolution is the pointwise product of Fourier transforms*:

Proposition 51 (Convolution Theorem)**.** *Let f and g be probability distributions on a group G. Then for any representation ρ,*

$$\left[\widehat{f * g}\right]_\rho = \widehat{f}_\rho \cdot \widehat{g}_\rho,$$

where the operation on the right side is matrix multiplication. Note that ρ is not necessarily assumed to be irreducible.

Proof.

$$\widehat{f}_\rho \widehat{g}_\rho = \left(\sum_\sigma f(\sigma)\rho(\sigma)\right)\left(\sum_\pi g(\pi)\rho(\pi)\right), \qquad \text{(Definition 24)}$$

Algorithm 8.2: Pseudocode for the Fourier Prediction/Rollup Algorithm. The input is $\left\{\hat{h}_\rho\right\}_{\rho\in\Lambda}$ and $\{\hat{q}_\rho\}_{\rho\in\Lambda}$, the Fourier coefficient matrices of h (the distribution at time t), and q, the mixing distribution, respectively.

FourierPredictionRollup$\left(\left\{\widehat{h}_\rho^{(t)}\right\}_{\rho\in\Lambda}, \left\{\widehat{q}_\rho^{(t)}\right\}_{\rho\in\Lambda}\right)$:

foreach $\rho \in \Lambda$ **do**
  $\hat{h}_\rho^{(t+1)} \leftarrow \hat{q}_\rho^{(t)} \cdot \hat{h}_\rho^{(t)}$;
end
return $\{\widehat{h}_\rho^{(t+1)}\}_{\rho\in\Lambda}$;

$$
\begin{aligned}
&= \sum_{\pi}\sum_{\sigma} f(\sigma\pi^{-1})\rho(\sigma\pi^{-1})g(\pi)\rho(\pi), \\
&= \sum_{\pi}\sum_{\sigma} f(\sigma\pi^{-1})g(\pi)\rho(\sigma\pi^{-1})\rho(\pi), \qquad \text{(rearranging)} \\
&= \sum_{\pi}\sum_{\sigma} f(\sigma\pi^{-1})g(\pi)\rho(\sigma), \qquad \text{(Definition 20)} \\
&= \sum_{\sigma} \left[f * g(\sigma)\right]\rho(\sigma), \qquad \text{(Definition 17)} \\
&= \left[\widehat{f * g}\right]_\rho, \qquad \text{(Definition 24).}
\end{aligned}
$$

□

Therefore, in the hidden Markov model setting, assuming that the Fourier transforms $\widehat{q}_\rho^{(t)}$ and $\widehat{h}_\rho^{(t)}$ are given, the prediction/rollup update rule (for random walk transition models) is simply:

Fourier domain convolution operation:

$$\widehat{h}_\rho^{(t+1)} \leftarrow \widehat{q}_\rho^{(t)} \cdot \widehat{h}_\rho^{(t)}. \tag{8.2}$$

See also the pseudocode in Algorithm 8.2. Note that the update only requires knowledge of $\hat{h}$ and does not require the distribution h itself. Furthermore, the update is *pointwise* in the Fourier domain in the sense that the coefficients at the representation ρ affect $\widehat{h}_\rho^{(t+1)}$ *only* at ρ. Consequently, prediction/rollup updates in the Fourier domain never increase the representational complexity. For example, if we maintain third-order marginals, then a single step of prediction/rollup called at time t returns the *exact* third-order marginals at time $t+1$, and nothing more.

Example 52. *We run the prediction/rollup routines on the first two time steps of the example in Figure 7, first in the primal domain, then in the Fourier domain. At each mixing event, two tracks, i and j, swap identities with some probability. Using a mixing model given by:*

$$q(\pi) = \begin{cases} 3/4 & \text{if } \pi = \epsilon \\ 1/4 & \text{if } \pi = (i,j) \\ 0 & \text{otherwise} \end{cases},$$

we obtain results shown in Tables 5 and 6.

σ	$h^{(0)}$	$q^{(1)}$	$h^{(1)}$	$q^{(2)}$	$h^{(2)}$
ϵ	1	3/4	3/4	3/4	9/16
$(1,2)$	0	1/4	1/4	0	3/16
$(2,3)$	0	0	0	0	0
$(1,3)$	0	0	0	1/4	3/16
$(1,2,3)$	0	0	0	0	1/16
$(1,3,2)$	0	0	0	0	0

Table 5: Primal domain prediction/rollup example.

	$\widehat{h}^{(0)}$	$\widehat{q}^{(1)}$	$\widehat{h}^{(1)}$
$\rho_{(3)}$	1	1	1
$\rho_{(2,1)}$	$\begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix}$	$\begin{bmatrix} 1/2 & 0 \\ 0 & 1 \end{bmatrix}$	$\begin{bmatrix} 1/2 & 0 \\ 0 & 1 \end{bmatrix}$
$\rho_{(1,1,1)}$	1	1/2	1/2

	$\widehat{q}^{(2)}$	$\widehat{h}^{(2)}$
$\rho_{(3)}$	1	1
$\rho_{(2,1)}$	$\begin{bmatrix} 7/8 & -\sqrt{3}/8 \\ -\sqrt{3}/8 & 5/8 \end{bmatrix}$	$\begin{bmatrix} 7/16 & -\sqrt{3}/8 \\ -\sqrt{3}/16 & 5/8 \end{bmatrix}$
$\rho_{(1,1,1)}$	1/2	1/4

Table 6: Fourier domain prediction/rollup example.

8.2.1 *Complexity of Prediction/Rollup*

We will discuss complexity in terms of the dimension of the largest maintained irreducible Fourier coefficient matrix, which we will denote by d_{max} (see Table 3 for irreducible dimensions). If we maintain 2^{nd} order marginals, for example, then $d_{max} = O(n^2)$, and if we maintain 3^{rd} order marginals, then $d_{max} = O(n^3)$.

Performing a single prediction/rollup step in the Fourier domain involves performing a single matrix multiplication for each irreducible and thus requires $O(d_{max}^3)$ time using the naive multiplication algorithm.

In certain situations, faster updates can be achieved. For example, in the pairwise mixing model of Example 52, the Fourier transform of q distribution takes the form: $\hat{q}_{\rho_\lambda} = \alpha I_{d_\lambda} + \beta \rho_\lambda(i,j)$, where I_{d_λ} is the $d_\lambda \times d_\lambda$ identity matrix (see also Chapter 7). As it turns out, the matrix $\rho_\lambda(i,j)$ can be factored into a product of $O(n)$ sparse matrices each with at most $O(d_\lambda)$

Algorithm 8.3: Algorithm for shifting function in the Fourier domain. The input is $\{\hat{h}_\rho\}_{\rho\in\Lambda}$, the collection of Fourier coefficient matrices of h, and the relabelings of the input and output spaces, given by π_{in} and π_{out}, respectively. See also Equation 3.10.

FOURIERSHIFT($\{\hat{h}_\rho\}_{\rho\in\Lambda}$):

foreach $\rho \in \Lambda$ **do**
  $\widehat{h'}_\rho \leftarrow \rho_\lambda(\pi_{out}) \cdot \hat{h}_\lambda \cdot \rho_\lambda^{-1}(\pi_{in})$;
end
return $\{\widehat{h'}_\rho\}_{\rho\in\Lambda}$;

nonzero entries. To see why, recall from out discussion in Chapter 2 that the transposition (i,j) factors into a sequence of $O(n)$ adjacent transpostions:

$$(i,j) = (i,i+1)(i+1,i+2)\cdots(j-1,j)(j-2,j-1)\cdot(i+1,i+2)(i,i+1).$$

If we use the Gel'fand-Tsetlin basis adapted to the subgroup chain $S_1 \subset \cdots \subset S_n$ (see Chapter 6), then we also know that the irreducible representation matrices evaluated at adjacent transpositions are sparse with no more than $O(d_{max}^2)$ nonzero entries. Thus by carefully exploiting sparsity during the prediction/rollup algorithm, one can achieve an $O(nd_{max}^2)$ update, which is faster than $O(d_{max}^3)$ as long as one uses more than first-order terms.

8.3 SHIFT OPERATIONS IN THE FOURIER DOMAIN

The shift operation (Equation 3.10) in the Fourier domain takes on a form similar to that of convolution (in fact, the operation of shifting can be viewed as convolution by a delta function). In this section, we present the Fourier domain version of the shifting operation. Let h be a distribution on S_n and let $h' = \text{SHIFT}[h, \pi_{in}, \pi_{out}]$. Given the Fourier transform of h, the following proposition answers how we can compute the Fourier transform of h':

Proposition 53. *Let λ be any partition of n. Then:*

Fourier domain shift operation:

$$\hat{h'}_\lambda = \text{SHIFT}[\hat{h}, \pi_{in}, \pi_{out}]_\lambda = \rho_\lambda(\pi_{out}) \cdot \hat{h}_\lambda \cdot \rho_\lambda^{-1}(\pi_{in}). \qquad (8.3)$$

(Notationally, we will write: $\widehat{h'} = \text{SHIFT}[\hat{h}, \pi_{in}, \pi_{out}]$).

Proof.

$$\begin{aligned}
\widehat{h'}_\lambda &= \sum_\sigma h'(\sigma)\rho_\lambda(\sigma), && \text{(Definition 24)} \\
&= \sum_\sigma h(\pi_{out}^{-1}\sigma\pi_{in})\rho_\lambda(\sigma), && \text{(Equation 3.10)} \\
&= \sum_{\sigma'} h(\sigma')\rho_\lambda(\pi_{out}\sigma'\pi_{in}^{-1}), && \text{(set } \sigma' = \pi_{out}^{-1}\sigma\pi_{in}\text{)} \\
&= \rho_\lambda(\pi_{out}) \left(\sum_\sigma h(\sigma)\rho_\lambda(\sigma) \right) \rho_\lambda^{-1}(\pi_{in}), \\
& && \text{(Definition 20, Linearity)} \\
&= \rho_\lambda(\pi_{out}) \cdot \widehat{h}_\lambda \cdot \rho_\lambda^{-1}(\pi_{in}), && \text{(Definition 24).}
\end{aligned}$$

□

See pseudocode for Fourier shifting in Algorithm 8.3. As with the Fourier domain convolution operation, the Fourier domain shift operation is *pointwise* in frequency level and therefore shifts never increase the underlying representational complexity. As with convolution, if we maintain, say, third-order marginals of h, then after shifting, we will still have exact third-order marginals of h'.

8.4 CONDITIONING IN THE FOURIER DOMAIN

In contrast with the prediction/rollup operation and shifting, conditioning *can* potentially increase the representational complexity. As an example, from the identity management setting, suppose that we know the following first-order marginal probabilities:

$$h(\text{Alice is at Track 1 or Track 2}) = .9, \text{ and}$$

$$h(\text{Bob is at Track 1 or Track 2}) = .9.$$

If we then make the following first-order observation:

$$h(\text{Cathy is at Track 1 or Track 2}) = 1,$$

then it can be inferred that Alice and Bob cannot *both* occupy Tracks 1 and 2 at the same time, i.e.,

$$h(\{\text{Alice,Bob}\} \text{ occupy Tracks } \{1,2\}) = 0,$$

demonstrating that after conditioning, we are left with knowledge of second-order (unordered) marginals despite the fact that the prior and likelihood functions were only known up to first-order. Intuitively, the example shows that conditioning "smears" information from low-order Fourier coefficients to high-order coefficients, and that one cannot hope for a pointwise operation as was afforded by prediction/rollup. We now show precisely how irreducibles of different complexities "interact" with each other in the Fourier domain during conditioning.

An application of Bayes rule (Equation 3.5) to find a posterior distribution $h(\sigma|z)$ after observing some evidence z requires two steps: a *pointwise product* of likelihood $h(z|\sigma)$ and prior $h(\sigma)$, followed by a normalization step:

$$h(\sigma|z) = \frac{1}{Z} \cdot h(z|\sigma) \cdot h(\sigma).$$

For notational convenience, we will refer to the likelihood function as $L(z|\sigma)$ henceforth. We showed earlier that the normalization constant $Z = \sum_\sigma L(z|\sigma) \cdot h(\sigma)$ is given by the Fourier transform of $\widehat{L^{(t)}h^{(t)}}$ at the trivial representation (Equation 8.1) — and therefore the normalization step of conditioning can be implemented by simply dividing each Fourier coefficient by the scalar $\left[\widehat{L^{(t)}h^{(t)}}\right]_{\rho_{(n)}}$.

The pointwise product of two functions f and g, however, is trickier to formulate in the Fourier domain. For functions on the real line, the pointwise product of functions can be implemented by convolving the Fourier coefficients of $\hat{f}$ and $\hat{g}$, and so a natural question is: can we apply a similar operation for functions over general groups? Our answer to this is that there is an analogous (but more complicated) notion of convolution in the Fourier domain of a general finite group. We present a convolution-based conditioning algorithm which we call *Kronecker Conditioning*, which, in contrast to the pointwise nature of the Fourier Domain prediction/rollup step, and much like convolution, smears the information at an irreducible ρ_ν to other irreducibles.

8.4.1 *Fourier transform of the pointwise product*

Our approach to computing the Fourier transform of the pointwise product in terms of $\hat{f}$ and $\hat{g}$ is to manipulate the function $f(\sigma)g(\sigma)$ so that it can be seen as the result of an inverse Fourier transform (Equation 5.6). Hence, the goal will be to find matrices R_ν (as a function of $\hat{f}, \hat{g}$) such that for any $\sigma \in G$,

$$f(\sigma) \cdot g(\sigma) = \frac{1}{|G|} \sum_\nu d_{\rho_\nu} \mathrm{Tr}\left(R_\nu^T \cdot \rho_\nu(\sigma)\right), \tag{8.4}$$

after which we will be able to read off the Fourier transform of the pointwise product as $\left[\widehat{fg}\right]_{\rho_\nu} = R_\nu$.

For any $\sigma \in G$, we can write the pointwise product in terms of $\hat{f}$ and $\hat{g}$ using the inverse Fourier transform:

$$\begin{aligned} f(\sigma) \cdot g(\sigma) &= \left[\frac{1}{|G|} \sum_\lambda d_{\rho_\lambda} \mathrm{Tr}\left(\hat{f}_{\rho_\lambda}^T \cdot \rho_\lambda(\sigma)\right)\right] \cdot \left[\frac{1}{|G|} \sum_\mu d_{\rho_\mu} \mathrm{Tr}\left(\hat{g}_{\rho_\mu}^T \cdot \rho_\mu(\sigma)\right)\right] \\ &= \left(\frac{1}{|G|}\right)^2 \sum_{\lambda,\mu} d_{\rho_\lambda} d_{\rho_\mu} \left[\mathrm{Tr}\left(\hat{f}_{\rho_\lambda}^T \cdot \rho_\lambda(\sigma)\right) \cdot \mathrm{Tr}\left(\hat{g}_{\rho_\mu}^T \cdot \rho_\mu(\sigma)\right)\right]. \end{aligned} \tag{8.5}$$

1. If A and B are square, $\mathrm{Tr}(A \otimes B) = (\mathrm{Tr}A) \cdot (\mathrm{Tr}B)$.
2. $(A \otimes B) \cdot (C \otimes D) = AC \otimes BD$.
3. Let A be an $n \times n$ matrix, and C an invertible $n \times n$ matrix. Then $\mathrm{Tr}A = \mathrm{Tr}\left(C^{-1}AC\right)$.
4. Let A be an $n \times n$ matrix and B_i be matrices of size $m_i \times m_i$ where $\sum_i m_i = n$. Then $\mathrm{Tr}\left(A \cdot \left(\bigoplus_i B_i\right)\right) = \sum_i \mathrm{Tr}\left(A_i \cdot B_i\right)$, where A_i is the block of A corresponding to block B_i in the matrix $\left(\bigoplus_i B_i\right)$.

Table 7: Matrix Identities used in Proposition 54.

Now we want to manipulate this product of traces in the last line to be just one trace (as in Equation 8.4), by appealing to some properties of the *Kronecker Product*. The Kronecker product of an $n \times n$ matrix $U = (u_{i,j})$ by an $m \times m$ matrix V, is defined to be the $nm \times nm$ matrix

$$U \otimes V = \begin{pmatrix} u_{1,1}V & u_{1,2}V & \dots & u_{1,n}V \\ u_{2,1}V & u_{2,2}V & \dots & u_{2,n}V \\ \vdots & \vdots & \ddots & \vdots \\ u_{n,1}V & u_{n,2}V & \dots & u_{n,n}V \end{pmatrix}.$$

We summarize some important matrix properties in Table 7. The connection to our problem is given by matrix property 1. Applying this to Equation 8.5, we have:

$$\begin{aligned} \mathrm{Tr}\left(\hat{f}^{T}_{\rho_\lambda} \cdot \rho_\lambda(\sigma)\right) \cdot \mathrm{Tr}\left(\hat{g}^{T}_{\rho_\mu} \cdot \rho_\mu(\sigma)\right) &= \mathrm{Tr}\left(\left(\hat{f}^{T}_{\rho_\lambda} \cdot \rho_\lambda(\sigma)\right) \otimes \left(\hat{g}^{T}_{\rho_\mu} \cdot \rho_\mu(\sigma)\right)\right) \\ &= \mathrm{Tr}\left(\left(\hat{f}_{\rho_\lambda} \otimes \hat{g}_{\rho_\mu}\right)^{T} \cdot \left(\rho_\lambda(\sigma) \otimes \rho_\mu(\sigma)\right)\right), \end{aligned}$$

where the last line follows by Property 2. The term on the left, $\hat{f}_{\rho_\lambda} \otimes \hat{g}_{\rho_\mu}$, is a matrix of coefficients. The term on the right, $\rho_\lambda(\sigma) \otimes \rho_\mu(\sigma)$, itself happens to be a representation, called the *Kronecker (or Tensor) Product Representation*. In general, the Kronecker product representation is reducible, and so it can be decomposed into a direct sum of irreducibles. In particular, if ρ_λ and ρ_μ are any two irreducibles of G, there exists a similarity transform $C_{\lambda\mu}$ such that, for any $\sigma \in G$,

$$C^{-1}_{\lambda\mu} \cdot [\rho_\lambda \otimes \rho_\mu](\sigma) \cdot C_{\lambda\mu} = \bigoplus_\nu \bigoplus_{\ell=1}^{z_{\lambda\mu\nu}} \rho_\nu(\sigma). \tag{8.6}$$

The $\oplus$ symbols here refer to a matrix direct sum as in Equation 5.3, ν indexes over all irreducible representations of S_n, while ℓ indexes over a number of *copies* of ρ_ν which appear in the decomposition. We index blocks on the right side of this equation by pairs of indices (ν, ℓ). The number of copies of each ρ_ν (for the tensor product pair $\rho_\lambda \otimes \rho_\mu$) is denoted by the integer $z_{\lambda\mu\nu}$, the collection of which, taken over all triples (λ, μ, ν), are commonly

referred to as the *Clebsch-Gordan series*. Note that we allow the $z_{\lambda\mu\nu}$ to be zero, in which case ρ_ν does not contribute to the direct sum. The matrices $C_{\lambda\mu}$ are known as the *Clebsch-Gordan coefficients*. The *Kronecker Product Decomposition* problem is that of finding the irreducible components of the Kronecker product representation, and thus to find the Clebsch-Gordan series/coefficients for each pair of irreducible representations (ρ_λ, ρ_μ).

Decomposing the Kronecker product inside Equation 8.6 using the Clebsch-Gordan series and coefficients yields the desired Fourier transform, which we summarize in the form of a proposition. In the case that f and g are defined over an Abelian group, we will show that the following formulas reduce to the familiar form of convolution.

Proposition 54. *Let $\hat{f}, \hat{g}$ be the Fourier transforms of functions f and g respectively, and for each ordered pair of irreducibles (ρ_λ, ρ_μ), define: $A_{\lambda\mu} \triangleq C_{\lambda\mu}^{-1} \cdot (\hat{f}_{\rho_\lambda} \otimes \hat{g}_{\rho_\mu}) \cdot C_{\lambda\mu}$. Then the Fourier tranform of the pointwise product fg is:*

Fourier domain conditioning operation:

$$\left[\widehat{fg}\right]_{\rho_\nu} = \frac{1}{d_{\rho_\nu}|G|} \sum_{\lambda\mu} d_{\rho_\lambda} d_{\rho_\mu} \sum_{\ell=1}^{z_{\lambda\mu\nu}} A_{\lambda\mu}^{(\nu,\ell)}, \tag{8.7}$$

where $A_{\lambda\mu}^{(\nu,\ell)}$ is the block of $A_{\lambda\mu}$ corresponding to the (ν,ℓ) block in $\bigoplus_\nu \bigoplus_{\ell=1}^{z_{\lambda\mu\nu}} \rho_\nu$ from Equation 8.6.

Proof. We use the fact that $C_{\lambda\mu}$ is an orthogonal matrix for all pairs (ρ_λ, ρ_μ), i.e., $C_{\lambda\mu}^T \cdot C_{\lambda\mu} = I$.

$$\begin{aligned}
f(\sigma)\cdot g(\sigma) &= \left[\frac{1}{|G|}\sum_\lambda d_{\rho_\lambda} \mathrm{Tr}\left(\hat{f}_{\rho_\lambda}^T \cdot \rho_\lambda(\sigma)\right)\right] \cdot \left[\frac{1}{|G|}\sum_\mu d_{\rho_\mu} \mathrm{Tr}\left(\hat{g}_{\rho_\mu}^T \cdot \rho_\mu(\sigma)\right)\right], \\
&= \left(\frac{1}{|G|}\right)^2 \sum_{\lambda,\mu} d_{\rho_\lambda} d_{\rho_\mu} \left[\mathrm{Tr}\left(\hat{f}_{\rho_\lambda}^T \cdot \rho_\mu(\sigma)\right) \cdot \mathrm{Tr}\left(\hat{g}_{\rho_\mu}^T \cdot \rho_\mu(\sigma)\right)\right], \\
&\qquad \text{(by Property 1)} \\
&= \left(\frac{1}{|G|}\right)^2 \sum_{\lambda,\mu} d_{\rho_\lambda} d_{\rho_\mu} \left[\mathrm{Tr}\left(\left(\hat{f}_{\rho_\lambda}^T \cdot \rho_\lambda(\sigma)\right) \otimes \left(\hat{g}_{\rho_\mu}^T \cdot \rho_\mu(\sigma)\right)\right)\right], \\
&\qquad \text{(by Property 2)} \\
&= \left(\frac{1}{|G|}\right)^2 \sum_{\lambda,\mu} d_{\rho_\lambda} d_{\rho_\mu} \mathrm{Tr}\left(\left(\hat{f}_{\rho_\lambda} \otimes \hat{g}_{\rho_\mu}\right)^T \cdot \left(\rho_\lambda(\sigma) \otimes \rho_\mu(\sigma)\right)\right), \\
&\qquad \text{(by Property 3)} \\
&= \left(\frac{1}{|G|}\right)^2 \sum_{\lambda,\mu} d_{\rho_\lambda} d_{\rho_\mu} \mathrm{Tr}\left(C_{\lambda\mu}^T \cdot \left(\hat{f}_{\rho_\lambda} \otimes \hat{g}_{\rho_\mu}\right)^T \cdot C_{\lambda\mu}\right. \\
&\qquad \left. \cdot C_{\lambda\mu}^T \cdot \left(\rho_\lambda(\sigma) \otimes \rho_\mu(\sigma)\right) \cdot C_{\lambda\mu}\right), \\
&\qquad \text{(by definition of } C_{\lambda\mu} \text{ and } A_{\lambda\mu})
\end{aligned}$$

$$= \left(\frac{1}{|G|}\right)^2 \sum_{\lambda,\mu} d_{\rho_\lambda} d_{\rho_\mu} \mathrm{Tr}\left(A_{\lambda\mu}^T \cdot \left(\bigoplus_{\nu} \bigoplus_{\ell=1}^{z_{\lambda\mu\nu}} \rho_\nu(\sigma)\right)\right),$$

(by Property 4)

$$= \frac{1}{|G|^2} \sum_{\lambda\mu} d_{\rho_\lambda} d_{\rho_\mu} \sum_{\nu} d_{\rho_\nu} \sum_{\ell=1}^{z_{\lambda\mu\nu}} \mathrm{Tr}\left(\left(d_{\rho_\nu}^{-1} A_{\lambda\mu}^{(\nu,\ell)}\right)^T \rho_\nu(\sigma)\right),$$

(rearranging terms)

$$= \frac{1}{|G|} \sum_{\nu} d_{\rho_\nu} \mathrm{Tr}\left[\left(\sum_{\lambda\mu} \sum_{\ell=1}^{z_{\lambda\mu\nu}} \frac{d_{\rho_\lambda} d_{\rho_\mu}}{d_{\rho_\nu}|G|} A_{\lambda\mu}^{(\nu,\ell)}\right)^T \rho_\nu(\sigma)\right].$$

Recognizing the last expression as an inverse Fourier transform completes the proof. □

The Clebsch-Gordan series, $z_{\lambda\mu\nu}$, plays an important role in Equation 8.7, which says that the (ρ_λ, ρ_μ) cross-term contributes to the pointwise product at ρ_ν *only* when $z_{\lambda\mu\nu} > 0$. In the simplest case, we have that

$$z_{(n),\mu,\nu} = \begin{cases} 1 & \text{if } \mu = \nu \\ 0 & \text{otherwise} \end{cases},$$

which is true since $\rho_{(n)}(\sigma) = 1$ for all $\sigma \in S_n$. As another example, it is known that:

$$\rho_{(n-1,1)} \otimes \rho_{(n-1,1)} \equiv \rho_{(n)} \oplus \rho_{(n-1,1)} \oplus \rho_{(n-2,2)} \oplus \rho_{(n-2,1,1)}, \tag{8.8}$$

or equivalently,

$$z_{(n-1,1),(n-1,1),\nu} = \begin{cases} 1 & \text{if } \nu \text{ is one of } (n),(n-1,1),(n-2,2), \text{ or } (n-2,1,1) \\ 0 & \text{otherwise} \end{cases}.$$

So if the Fourier transforms of the likelihood and prior are zero past the first two irreducibles ((n) and $(n-1,1)$), then a single conditioning step results in a Fourier transform which, in general, carries second-order information at $(n-2,2)$ and $(n-2,1,1)$, but is guaranteed to be zero past the first four irreducibles (n), $(n-1,1)$, $(n-2,2)$ and $(n-2,1,1)$.

As far as we know, there are no analytical formulas for finding the entire Clebsch-Gordan series or coefficients, and in practice, acquiring the coefficients requires considerable precomputation. We emphasize however, that as fundamental constants related to the irreducibles of the Symmetric group, they need only be computed *once and for all* (like the digits of π, for example) and can be stored in a table for all future reference. For a detailed discussion of techniques for computing the Clebsch-Gordan series/coefficients, see Appendices B and D. We have made a set of precomputed coefficients available on our lab website [1], but we will assume throughout the rest of the thesis that both the series and coefficients have been

1 See http://www.select.cs.cmu.edu/data/index.html

made available as a lookup table. Algorithm 8.4 provides pseudocode for Kronecker conditioning.

As a final remark, note that Proposition 54 can be rewritten somewhat more intuitively by absorbing the scalars and submatrices of the Clebsch-Gordan coefficients into projection matrices $P_{\lambda\mu}^{(\nu,\ell)}$.

Proposition 55. *Let $\hat{f}, \hat{g}$ be the Fourier transforms of functions f and g respectively. For each triple of partitions (λ, μ, ν) there exists a positive integer $z_{\lambda,\mu,\nu}$ and projection operators $P_{\lambda\mu}^{(\nu,\ell)}$ for each $\ell \in \{1, 2, \ldots, z_{\lambda\mu\nu}\}$ such that the Fourier tranform of the pointwise product fg is:*

$$\left[\widehat{fg}\right]_{\rho_\nu} = \sum_{\lambda,\mu} \sum_{\ell=1}^{z_{\lambda\mu\nu}} (P_{\lambda\mu}^{(\nu,\ell)})^T \cdot (\hat{f}_{\rho_\lambda} \otimes \hat{g}_{\rho_\mu}) \cdot P_{\lambda\mu}^{(\nu,\ell)}. \tag{8.9}$$

When f and g are functions on an Abelian group G, then it is a well known fact that all irreducible representations are one-dimensional, and so Equation 8.9 reduces to $\left[\widehat{fg}\right]_{\rho_\nu} = \sum_{\lambda,\mu} (\hat{f}_{\rho_\lambda} \cdot \hat{g}_{\rho_\mu})$, where all the tensor products have simply become scalar multiplications and the familiar definition of convolution is recovered.

8.4.2 *Complexity of conditioning*

The complexity of (bandlimited) conditioning (assuming precomputed Clebsch-Gordan series/coefficients) depends on the order of the coefficients maintained for both the prior and the observation model. However, it is difficult to state a general complexity bound for arbitrary finite groups due to our limited understanding of the Clebsch-Gordan series. Here we consider conditioning only on the symmetric group of order n with the assumption that the number of irreducibles maintained is very small (and in particular, not allowed to grow with respect to n). Our assumption is realistic in practice since for moderately large n, it is impractical to consider maintaining higher than, say, third-order terms. If we denote the dimension of the largest maintained irreducibles of the prior and likelihood by d_{max}^{prior} and d_{max}^{obs}, respectively, then the complexity of conditioning is dominated by the step that forms a matrix $C^T \cdot (A \otimes B) \cdot C$, where the matrices $A \otimes B$ and C are each $(d_{max}^{prior} \cdot d_{max}^{obs})$-dimensional. Note, however, that since we are only interested in certain blocks of $C^T \cdot (A \otimes B) \cdot C$, the full matrix need not be computed. In particular, the largest extracted block has size d_{max}^{prior}, and so the complexity of conditioning is $O\left((d_{max}^{obs})^2 (d_{max}^{prior})^3\right)$ using the naive matrix multiplication algorithm.

As we have discussed in Chapter 7, there exist situations in which the observation model is fully specified by first-order Fourier terms. In such cases, $d_{max}^{obs} = O(n)$ and we can perform conditioning in the Fourier domain in $O(n^2 \cdot (d_{max}^{prior})^3)$ time. If a model is fully specified by second-order terms, for example, then the update requires $O(n^4 \cdot (d_{max}^{prior})^3)$ time.

To speed up conditioning, one can often exploit matrix sparsity in two ways. First, we observe that the Clebsch-Gordan coefficient matrices are

often sparse (we cannot yet prove this, see Figure 22) and so we can save a conjectured factor of $(d_{max}^{prior} \cdot d_{max}^{obs})$ in practice. Secondly, for the coset-based observation models discussed in Chapter 7, we can show that (under an appropriate relabeling of identities and tracks), the Fourier coefficient matrices of the observation model are sparse (with $O(d_{max}^{obs})$ or sometimes even $O(1)$ nonzero entries for $\hat{L}_\lambda$). For the simplest observations which take the form ("Identity j is at track j"), for example, we can obtain $O((d_{max}^{prior})^3)$ running time (without accounting for the conjectured sparsity of the Clebsch-Gordan coefficients), which matches the time required for the prediction/rollup update.

We now conclude our section on inference with a fully worked example of Kronecker conditioning.

Example 56. *For this example, refer to Table 2 for the representations of S_3. Given functions $f, g : S_3 \to \mathbb{R}$, we will compute the Fourier transform of the pointwise product $f \cdot g$.*

Since there are three irreducibles, there are nine tensor products $\rho_\lambda \otimes \rho_\mu$ to decompose, six of which are trivial either because they are one-dimensional, or involve tensoring against the trivial representation. The nontrivial tensor products to consider are $\rho_{(2,1)} \otimes \rho_{(1,1,1)}$, $\rho_{(1,1,1)} \otimes \rho_{(2,1)}$ and $\rho_{(2,1)} \otimes \rho_{(2,1)}$. The Clebsch-Gordan series for the nontrivial tensor products are:

	$z_{(2,1),(1,1,1),\nu}$	$z_{(1,1,1),(2,1),\nu}$	$z_{(2,1),(2,1),\nu}$
$\nu = (3)$	0	0	1
$\nu = (2,1)$	1	1	1
$\nu = (1,1,1)$	0	0	1

The Clebsch-Gordan coefficients for the nontrivial tensor products are given by the following orthogonal matrices:

$$C_{(2,1)\otimes(1,1,1)} = \begin{bmatrix} 0 & 1 \\ -1 & 0 \end{bmatrix}, \qquad C_{(1,1,1)\otimes(2,1)} = \begin{bmatrix} 0 & -1 \\ 1 & 0 \end{bmatrix},$$

$$C_{(2,1)\otimes(2,1)} = \frac{\sqrt{2}}{2} \begin{bmatrix} 1 & 0 & -1 & 0 \\ 0 & -1 & 0 & 1 \\ 0 & -1 & 0 & -1 \\ 1 & 0 & 1 & 0 \end{bmatrix}.$$

As in Proposition 54, define:

$$A_{(2,1)\otimes(1,1,1)} = C_{(2,1)\otimes(1,1,1)}^T \left(\hat{f}_{(2,1)} \otimes \hat{g}_{(1,1,1)}\right) C_{(2,1)\otimes(1,1,1)}, \quad (8.10)$$

$$A_{(1,1,1)\otimes(2,1)} = C_{(1,1,1)\otimes(2,1)}^T \left(\hat{f}_{(1,1,1)} \otimes \hat{g}_{(2,1)}\right) C_{(1,1,1)\otimes(2,1)}, \quad (8.11)$$

$$A_{(2,1)\otimes(2,1)} = C_{(2,1)\otimes(2,1)}^T \left(\hat{f}_{(2,1)} \otimes \hat{g}_{(2,1)}\right) C_{(2,1)\otimes(2,1)}, \quad (8.12)$$

Then Proposition 54 gives the following formulas:

$$\widehat{f \cdot g}_{\rho_{(3)}} = \frac{1}{3!} \cdot \left[\hat{f}_{\rho_{(3)}} \cdot \hat{g}_{\rho_{(3)}} + \hat{f}_{\rho_{(1,1,1)}} \cdot \hat{g}_{\rho_{(1,1,1)}} + 4 \cdot \left[A_{(2,1)\otimes(2,1)}\right]_{1,1}\right], \quad (8.13)$$

$$\widehat{f \cdot g}_{\rho_{(2,1)}} = \frac{1}{3!} \cdot \left[\hat{f}_{\rho_{(2,1)}} \cdot \hat{g}_{\rho_{(3)}} + \hat{f}_{\rho_{(3)}} \cdot \hat{g}_{\rho_{(2,1)}} + A_{(1,1,1)\otimes(2,1)}\right.$$

$$+A_{(2,1)\otimes(1,1,1)} + 2 \cdot \left[A_{(2,1)\otimes(2,1)}\right]_{2:3,2:3}\Big], \tag{8.14}$$

$$\widehat{f \cdot g}_{\rho_{(1,1,1)}} = \frac{1}{3!} \cdot \left[\hat{f}_{\rho_{(3)}} \cdot \hat{g}_{\rho_{(1,1,1)}} + \hat{f}_{\rho_{(1,1,1)}} \cdot \hat{g}_{\rho_{(3)}} + 4 \cdot \left[A_{(2,1)\otimes(2,1)}\right]_{4,4}\right], \tag{8.15}$$

where the notation $[A]_{a:b,c:d}$ *denotes the block of entries in* A *between rows* a *and* b*, and between columns* c *and* d *(inclusive).*

Using the above formulas, we can continue on Example 52 and compute the last update step in our identity management problem (Figure 7). At the final time step, we observe that Bob is at track 1 with 100% certainty. Our likelihood function is therefore nonzero only for the permutations which map Bob (the second identity) to the first track:

$$L(\sigma) \propto \begin{cases} 1 & \text{if } \sigma = (1,2) \text{ or } (1,3,2) \\ 0 & \text{otherwise} \end{cases}.$$

The Fourier transform of the likelihood function is:

$$\widehat{L}_{\rho_{(3)}} = 2, \quad \widehat{L}_{\rho_{(2,1)}} = \begin{bmatrix} -3/2 & \sqrt{3}/2 \\ -\sqrt{3}/2 & 1/2 \end{bmatrix}, \quad \widehat{L}_{\rho_{(1,1,1)}} = 0. \tag{8.16}$$

Plugging the Fourier transforms of the prior distribution ($\widehat{h}^{(2)}$ *from Table 6) and likelihood (Equation 8.16) into Equations 8.10, 8.11, 8.12, we have:*

$$A_{(2,1)\otimes(1,1,1)} = \begin{bmatrix} 0 & 0 \\ 0 & 0 \end{bmatrix}, \qquad A_{(1,1,1)\otimes(2,1)} = \frac{1}{8}\begin{bmatrix} 1 & \sqrt{3} \\ -\sqrt{3} & -3 \end{bmatrix},$$

$$A_{(2,1)\otimes(2,1)} = \frac{1}{32}\begin{bmatrix} -7 & -\sqrt{3} & 11 & 5\sqrt{3} \\ -2\sqrt{3} & -10 & -6\sqrt{3} & -14 \\ 20 & 22\sqrt{3} & -4 & 4\sqrt{3} \\ -11\sqrt{3} & -23 & -\sqrt{3} & -13 \end{bmatrix}$$

To invoke Bayes rule in the Fourier domain, we perform a pointwise product using Equations 8.13, 8.14, 8.15, and normalize by dividing by the trivial coefficient, which yields the Fourier transform of the posterior distribution as:

$$\left[\widehat{h(\sigma|z)}\right]_{\rho_{(3)}} = 1, \quad \left[\widehat{h(\sigma|z)}\right]_{\rho_{(2,1)}} = \begin{bmatrix} -1 & 0 \\ 0 & 1 \end{bmatrix}, \quad \left[\widehat{h(\sigma|z)}\right]_{\rho_{(1,1,1)}} = -1. \tag{8.17}$$

Finally, we can see that the result is correct by recognizing that the Fourier transform of the posterior (Equation 8.17) corresponds exactly to the distribution which is 1 at $\sigma = (1,2)$ *and 0 everywhere else. Bob is therefore at Track 1, Alice at Track 2 and Cathy at Track 3.*

σ	ϵ	$(1,2)$	$(2,3)$	$(1,3)$	$(1,2,3)$	$(1,3,2)$
$h(\sigma)$	0	1	0	0	0	0

Algorithm 8.4: Pseudocode for the Kronecker Conditioning Algorithm. Inputs are $\{\widehat{L}_{\rho_\lambda}\}_{\rho_\lambda \in \Lambda}$ and $\{\widehat{h}_{\rho_\mu}\}_{\rho_\mu \in \Lambda}$, the Fourier coefficient matrices of the likelihood function L and prior distribution h.

FourierKroneckerCondition$\left(\{\widehat{L}_{\rho_\lambda}\}_{\rho_\lambda \in \Lambda}, \{\widehat{h}_{\rho_\mu}\}_{\rho_\mu \in \Lambda}\right)$:

foreach $\rho_\nu \in \Lambda_P$ **do** $\widehat{LP}_{\rho_\nu} \leftarrow \mathbf{0}$ *//Initialize Posterior*
//Pointwise Product
foreach $\rho_\lambda \in \Lambda_L$ **do**
  foreach $\rho_\mu \in \Lambda_P$ **do**
    $z \leftarrow \text{CGseries}(\rho_\lambda, \rho_\mu)$;
    $C_{\lambda\mu} \leftarrow \text{CGcoefficients}(\rho_\lambda, \rho_\mu)$; $A_{\lambda\mu} \leftarrow C_{\lambda\mu}^T \cdot (\hat{L}_{\rho_\lambda} \otimes \hat{P}_{\rho_\mu}) \cdot C_{\lambda\mu}$;
    for $\rho_\nu \in \Lambda_P$ *such that* $z_{\lambda\mu\nu} \neq 0$ **do**
      for $\ell = 1$ ***to*** $z_{\lambda\mu\nu}$ **do**
        $\left[\widehat{L^{(t)}h^{(t)}}\right]_{\rho_\nu} \leftarrow \left[\widehat{L^{(t)}h^{(t)}}\right]_{\rho_\nu} + \frac{d_{\rho_\lambda} d_{\rho_\mu}}{d_{\rho_\nu} n!} A_{\lambda\mu}^{(\nu,\ell)}$; *//$A_{\lambda\mu}^{(\nu,\ell)}$ is the (ν,ℓ) block of $A_{\lambda\mu}$*
      end
    end
  end
end
return FourierNormalize $\left(\left\{\widehat{L^{(t)}h^{(t)}}_{\rho_\nu}\right\}_{\rho_\nu \in \Lambda}\right)$;

8.5 MAXIMIZATION

In certain situations, one might be more interested in computing a maximum a posteriori permutation rather than probabilities. In this section, we briefly discuss the Fourier domain maximization problem, in which we compute $\arg\max_{\sigma \in S_n} h(\sigma)$ given the Fourier coefficients of h. As we have discussed in Chapter 3, it is intractable to optimize arbitrary functions; it is interesting however, to ask whether it is any easier to optimize low-order functions (functions whose Fourier coefficients are zero past k^{th}-order terms).

For first-order functions, polynomial time optimization *is* indeed possible. To see why, consider a first-order function h whose Fourier coefficient matrices are zero except at the terms $\lambda = (n)$ and $(n-1,1)$. Expanding h with respect to its Fourier coefficients allows for h to be written as the sum of a constant function and first-order term:

$$h(\sigma) = A + \text{Tr}(B^T \cdot \tau_{(n-1,1)}(\sigma)).$$

Therefore to optimize the function h, one must solve problems of the following form:

$$\arg\max_{\sigma \in S_n} \text{Tr}(B^T \cdot \tau_{(n-1,1)}(\sigma)) = \arg\max_{\sigma \in S_n} \sum_{i=1}^{n} B_{\sigma(i),i}. \tag{8.18}$$

Since Equation 8.18 is simply a linear assignment problem, we see that h can be optimized exactly in polynomial time using, for example, linear programming or the Hungarian algorithm [131].

HIGHER-ORDER MAXIMIZATION. It may then seem natural that if we could optimize first-order functions in polynomial time, then we would be

able to also efficiently optimize second-order functions in polynomial time, but as we show in the following, this is not true in general.

Below, we consider a second-order (unordered) function $f : S_n \to \mathbb{R}$ which can be written using only Fourier terms corresponding to the first three partitions, $\lambda = (n)$, $(n-1,1)$, and $(n-2,2)$. Thus, any second-order function f can be written as:

$$f(\sigma) = A + \mathrm{Tr}(B^T \cdot \rho_{(n-1,1)}(\sigma)) + \mathrm{Tr}(C^T \cdot \rho_{(n-2,2)}),$$

for some matrices A, B, C. We consider the second-order optimization problem (which we will refer to as *SOOP*), in which one must find the minimum of such a function f. In particular, we provide a simple polynomial-time reduction from the Travelling Salesman problem to SOOP, thus showing that optimizing second-order functions is NP-hard

Let $G = (V, E)$ be a complete (undirected) graph on n vertices with a cost function defined on edges $c : E \to \mathbb{R}^+$. The objective of the *Travelling Salesman Problem (TSP)* is to find a minimum cost roundtrip path (*tour*) that visits every vertex exactly once. We will denote the set of all tours by Θ. The TSP is known to be NP-hard and cannot even be deterministically approximated in polynomial time. We prove the following.

Theorem 57. *Any TSP instance can be reduced to an instance of SOOP in polynomial time.*

Proof. We define a map $\phi : S_n \to \Theta$ which associates each permutation $\sigma = [\sigma_1\ \sigma_2\ \ldots\ \sigma_n] \in S_n$ to a tour through G which visits vertex σ_i at step i:

$$\phi : \sigma \mapsto\ (\sigma_1 - \sigma_2 - \cdots - \sigma_n).$$

Note that ϕ is a many-to-one mapping and several permutations can be associated with the same tour since cyclic permutations of σ result in the same tour. However, it is important to note that ϕ is surjective on Θ.

The cost function for any TSP instance can therefore also be written as a function over permutations, $f : S_n \to \mathbb{R}^+$:

$$f(\sigma) = \sum_{i=1}^{n} f_i(\sigma),$$

where, for each i, $f_i : S_n \to \mathbb{R}$ is defined by $f_i(\sigma) = c(\sigma_i, \sigma_{i+1})$ (except when $i = n$, in which case $f_n(\sigma) = c(\sigma_n, \sigma_1)$). The solution to the TSP instance is given by: $\phi(\arg\min_\sigma f(\sigma))$.

To see that f is a second-order function and that its Fourier coefficients can be computed in polynomial time from the edge costs c, note that each f_i can be written as a sum of $O(n^2)$ indicator functions:

$$f_i(\sigma) = \sum_{(u,v)\in E} c(u,v)\delta_{\sigma(\{i,i+1\})=\{u,v\}}(\sigma).$$

We see that f can be written as a sum of $O(n^3)$ second-order indicators, and is therefore itself a second-order function with nonzero coefficients computable in polynomial time.

We have shown that we can solve a TSP instance by finding the Fourier coefficients of f in polynomial time, calling an oracle to obtain $\arg\min_\sigma f(\sigma)$, then mapping the solution to a tour via ϕ. □

We conclude by remarking that despite the hardness results that we have discussed in this section, there exist approaches that lack theoretical guarantees but have been shown to work in practical situations. For example, see [77, 78] for efficient branch-and-bound approaches to the optimization problem. In the experimental results that we present in the next chapter, we simply optimize the first-order terms of the maintained posterior distributions.

8.6 SUMMARY

Fourier theoretic summaries are attractive because they have tuneable approximation quality, and have intuitive interpretations in terms of low-order marginals. As we have shown in this chapter, they also allow us to leverage results and insights from noncommutative Fourier analysis to formulate inference algorithms.

The main contributions of this chapter include methods for performing general probabilistic inference operations completely in the Fourier domain. In particular, we have shown how general probabilistic manipulations such as prediction/rollup, conditioning, shifting, normalization and maximization can be recast as algorithms operating on Fourier coefficients. As the primary contribution of this chapter, we developed the Kronecker conditioning algorithm, which conditions a distribution on evidence using Bayes rule while using only Fourier coefficients. While prediction/rollup operations can be written as pointwise products in the Fourier domain, we showed that conditioning operations can be written, in dual fashion, as generalized convolutions in the Fourier domain. Our conditioning algorithm is general and handles any observation model which can be written in the Fourier domain. Due to this generality, we are able to efficiently compute the Fourier transforms of a wide variety of probabilistic models (see Chapter 7) which may potentially be useful in different applications.

We conclude by remarking that the mathematical framework developed in this chapter is quite general. In fact, the prediction/rollup, conditioning, normalization and shifting formulations *hold over any finite group*, providing a principled method for approximate inference for problems with underlying group structure.

9

APPROXIMATE BANDLIMITED INFERENCE

FOR each of the probabilistic inference algorithms that have been presented in the previous chapter, it is possible to run a bandlimited version of the same algorithm simply by ignoring high frequency terms. For example, one can maintain only up to kth order terms in a multi-target tracking problem, thus allowing for polynomial time inference. Ignoring terms comes at a price, however, and in this chapter, we explore the consequences of this bandlimiting approach on approximation accuracy.

We draw several important conclusions in this chapter:

- As we discuss, much like ordinary functions on the real line, smoother distributions are better approximated by low frequency Fourier basis functions, and thus Fourier theoretic approaches are well suited to problems with high uncertainty.
- We show that among the inference operations of convolution, normalization, shifting, and conditioning, conditioning is the only operation during which errors can be introduced in the bandlimiting approximation.
- We explore how errors during conditioning can propagate from high order to low order during inference and propose a method for dealing with this error, We show in particular that it is typically both faster and more accurate to condition upon low-order observations (i.e., observations which involve only several items or objects at a time).
- Finally, we demonstrate the effectiveness of our approximate inference approach on a real camera-based multi-people tracking setting.

9.1 ERROR FROM BANDLIMITING

We now consider the consequences of performing inference using the Fourier transform at a reduced set of coefficients. Important issues include understanding how error can be introduced into the system, and when our algorithms are expected to perform well as an approximation. Specifically, we fix a bandlimit λ^{MIN} and maintain the Fourier transform of h only at irreducibles which are at λ^{MIN} or above in the dominance ordering:

$$\Lambda = \{\rho_\lambda \ : \ \lambda \trianglerighteq \lambda^{\text{MIN}}\}.$$

For example, when $\lambda^{\text{MIN}} = (n-2,1,1)$, Λ is the set $\{\rho_{(n)}, \rho_{(n-1,1)}, \rho_{(n-2,2)}, \text{ and } \rho_{(n-2,1,1)}\}$, which corresponds to maintaining second-order (ordered) marginal probabilities of the form $h(\sigma((i,j)) = (k,\ell))$. During inference, we follow the procedure outlined in the previous section but discard the higher

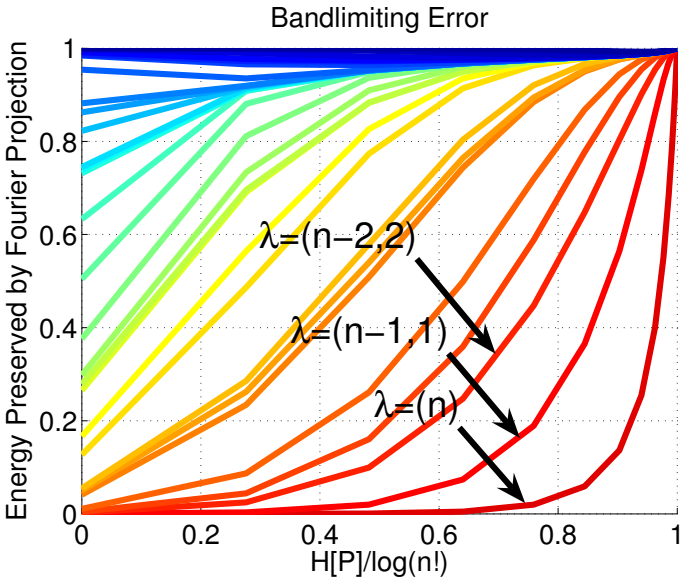


Figure 15: In general, smoother distributions are well approximated by low-order Fourier projections. In this graph, we show the approximation quality of the Fourier projections on distributions with different entropies, starting from sharply peaked delta distributions on the left side of the graph, which get iteratively smoothed until they becomes the maximum entropy uniform distribution on the right side. On the y-axis, we measure how much *energy* is preserved in the bandlimited approximation, which we define to be $\frac{|h'|^2}{|h|^2}$, where h' is the bandlimited approximation to h. Each line represents the approximation quality using a fixed number of Fourier coefficients. At one extreme, we achieve perfect signal reconstruction by using all Fourier coefficients, and at the other, we perform poorly on "spiky" distributions, but well on high-entropy distributions, by storing a single Fourier coefficient.

order terms which can be introduced during the conditioning step. We note that it is not necessary to maintain the same number of irreducibles for both prior and likelihood during the conditioning step.

The first question to ask is: when should one expect a bandlimited approximation to be close to $h(\sigma)$ as a function? Qualitatively, if a distribution is relatively smooth, then most of its energy is stored in the low-order Fourier coefficients. However, in a phenomenon quite reminiscent of the Heisenberg uncertainty principle from quantum mechanics, it is exactly when the distribution is sharply concentrated at a small subset of permutations, that the Fourier projection is unable to faithfully approximate the distribution. We illustrate this uncertainty effect in Figure 15 by plotting the accuracy of a bandlimited distribution against the entropy of a distribution.

Even though the bandlimited distribution is sometimes a poor approximation to the true distribution, the marginals maintained by our algorithm are often sufficiently accurate. And so instead of considering the approximation accuracy of the bandlimited Fourier transform to the true joint distribution, we consider the accuracy only at the marginals which are maintained by our method.

9.2 ERROR FROM INFERENCE

We now analyze the errors incurred during our inference procedures with respect to the accuracy at maintained marginals. It is immediate that the Fourier domain prediction/rollup operation is *exact* due to its pointwise nature in the Fourier domain. For example, if we have the second order marginals at time $t = 0$, then we can find the exact second order marginals at all $t > 0$ if we only perform prediction/rollup operations.

CONVOLUTIONS INCREASE UNCERTAINTY. In fact, the convolution updates for prediction/rollup typically *increase* the uncertainty of a distribution, often leading to more accurate approximations by a low order representation. Shin et al. [121] showed, for example, that the entropy must increase for a certain kind of random walk on S_n (where π could be either the identity or the transposition (i, j)), but in fact, we show now that the result is easily generalized for any random walk mixing model and for any finite group.

Proposition 58. *Let* G *be a finite group. Suppose* $h(\sigma^{(1)}, \ldots, \sigma^{(T)}, z^{(1)}, \ldots, z^{(T)})$ *factors according to a hidden Markov model, with* $\sigma^{(t)} \in G$ *for each* t*, and* $\sigma^{(t+1)} = \tau\sigma(t)$ *with* $\tau \sim q$ *for some mixing distribution* q*. Then:*

$$H\left[h^{(t+1)}(\sigma^{(t+1)})\right] \geqslant \max\left\{H\left[q^{(t)}(\tau^{(t)})\right], H\left[h^{(t)}(\sigma^{(t)})\right]\right\},$$

where $H[h(\sigma)]$ *denotes the statistical entropy functional,*

$$H[h(\sigma)] = -\sum_{\sigma \in G} h(\sigma) \log h(\sigma).$$

Proof. We have:

$$\begin{aligned} h^{(t+1)}(\sigma^{(t+1)}) &= \left[q^{(t)} * h^{(t)}\right](\sigma^{(t+1)}) \\ &= \sum_{\sigma^{(t)}} q(\sigma^{(t+1)} \cdot (\sigma^{(t)})^{-1}) h^{(t)}(\sigma^{(t)}) \end{aligned}$$

Applying the Jensen Inequality to the entropy function (which is concave) yields:

$$\begin{aligned} H\left[h^{(t+1)}(\sigma^{(t+1)})\right] &\geqslant \sum_{\sigma^{(t)}} h^{(t)}(\sigma^{(t)}) H\left[q^{(t)}(\sigma \cdot (\sigma^{(t)})^{-1})\right], \\ &\qquad \text{(Jensen's inequality)} \\ &= \sum_{\sigma^{(t)}} h^{(t)}(\sigma^{(t)}) H\left[q^{(t)}(\sigma)\right], \\ &\qquad \text{(translation invariance of entropy)} \\ &= H\left[q^{(t)}(\sigma)\right], \\ &\qquad \text{(since } \textstyle\sum_{\sigma^{(t)}} h^{(t)}(\sigma^{(t)}) = 1\text{).} \end{aligned}$$

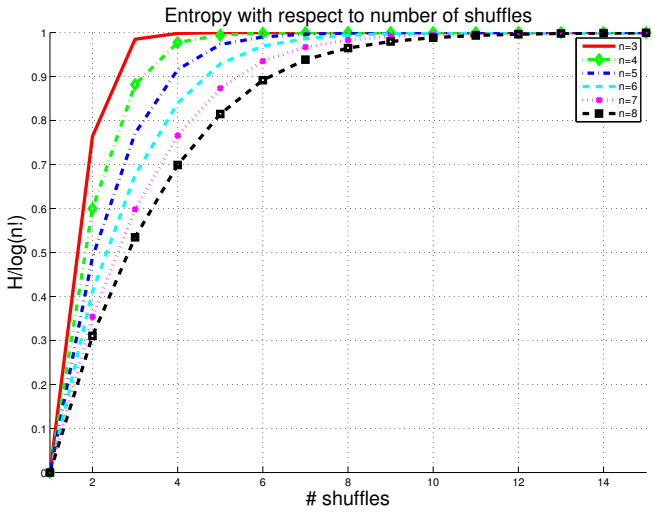


Figure 16: We start with a deck of cards in sorted order, and perform fifteen consecutive shuffles according to the rule given in Equation 9.1. The plot shows the entropy of the distribution over permutations with respect to the number of shuffles for $n = 3, 4, \ldots, 8$. When $H(q)/\log(n!) = 1$, the distribution has become uniform.

The proof that $H\left[h^{(t+1)}(\sigma^{(t+1)})\right] \geqslant H\left[h^{(t)}(\sigma^{(t)})\right]$ is similar with the exception that we must rewrite the convolution so that the sum ranges over $\tau^{(t)}$.

$$\begin{aligned} h^{(t+1)}(\sigma^{(t+1)}) &= \left[q^{(t)} * h^{(t)}\right](\sigma^{(t+1)}), \\ &= \sum_{\tau^{(t)}} q^{(t)}(\tau^{(t)}) h^{(t)}((\tau^{(t)})^{-1} \cdot \sigma^{(t+1)}). \end{aligned}$$

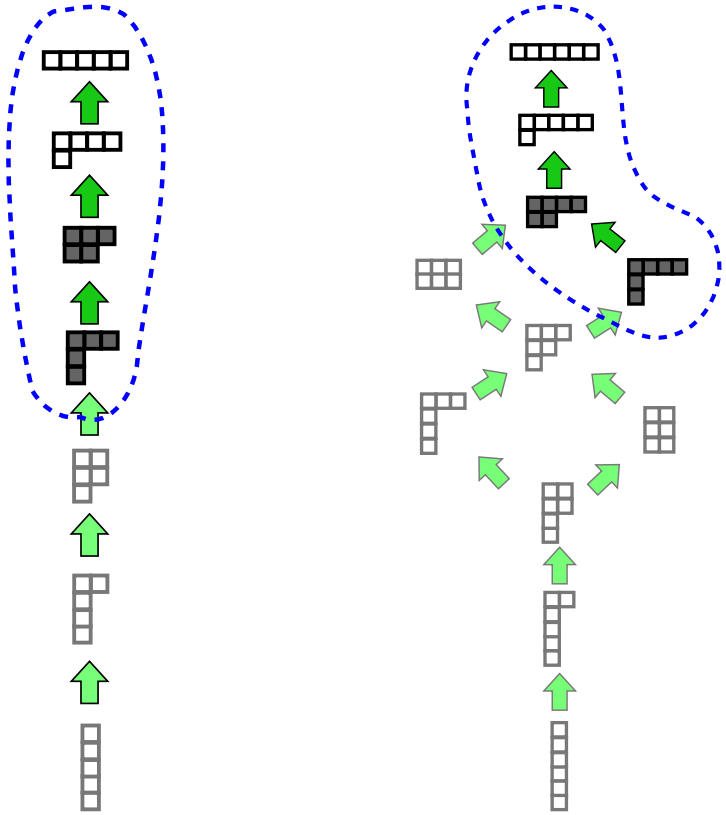
□

Example 59. *This example is based on one from [29]. Consider a deck of cards numbered $\{1, \ldots, n\}$. Choose a random permutation of cards by first picking two cards independently, and swapping (a card might be swapped with itself), yielding the following probability distribution over S_n:*

$$q(\pi) = \begin{cases} \frac{1}{n} & \text{if } \pi = \epsilon \\ \frac{2}{n^2} & \text{if } \pi \text{ is a transposition} \\ 0 & \text{otherwise} \end{cases}. \tag{9.1}$$

Repeating the above process for generating random permutations π gives a transition model for a hidden Markov model over the symmetric group. We can also see (Figure 16) that the entropy of the deck increases monotonically with each shuffle, and that repeated shuffles with $q(\pi)$ eventually bring the deck to the uniform distribution.

ERROR FROM CONDITIONING. Instead, the errors in inference are only committed by Kronecker conditioning, where they are implicitly introduced



(a) $n = 5$ (b) $n = 6$

Figure 17: We show the dominance ordering for partitions of $n = 5$ and $n = 6$ again. By setting $\lambda^{\text{MIN}} = (3,1,1)$ and $(4,1,1)$ respectively, we keep the irreducibles corresponding to the partitions in the dotted regions. If we call Kronecker Conditioning with a first-order observation model, then according to Theorem 60, we can expect to incur some error at the Fourier coefficients corresponding to $(3,1,1)$ and $(3,2)$ for $n = 5$, and $(4,1,1)$ and $(4,2)$ for $n = 6$ (shown as shaded tableaux), but to be exact at first-order coefficients.

at coefficients outside of Λ (by effectively setting the coefficients of the prior and likelihood at irreducibles outside of Λ to be zero), then propagated inside to the irreducibles of Λ.

In practice, we observe that the errors introduced at the low-order irreducibles during inference are small if the prior and likelihood are sufficiently diffuse, which makes sense since the high-frequency Fourier coefficients are small in such cases. We can sometimes show that the update is *exact* at low order irreducibles if we maintain *enough* coefficients.

Theorem 60. *If $\lambda^{\text{MIN}} = (n-p, \lambda_2, \dots)$, and the Kronecker conditioning algorithm is called with a likelihood function whose Fourier coefficients are nonzero only at ρ_μ when $\mu \trianglerighteq (n-q, \mu_2, \dots)$, then the approximate Fourier coefficients of the posterior distribution are exact at the set of irreducibles:*

$$\Lambda_{\text{EXACT}} = \{\rho_\lambda \ : \ \lambda \trianglerighteq (n - |p-q|, \dots)\}.$$

Proof. See Appendix B. □

For example, if we call Kronecker conditioning by passing in third-order terms of the prior and first-order terms of the likelihood, then all first and second-order (unordered and ordered) marginal probabilities of the posterior distribution can be reconstructed without error.

9.3 PROJECTION TO THE MARGINAL POLYTOPE

Despite the encouraging result of Theorem 60, the fact remains that consecutive conditioning steps can propagate errors to all levels of the bandlimited Fourier transform, and in many circumstances, result in a Fourier transform whose "marginal probabilities" correspond to no consistent joint distribution over permutations, and are sometimes negative. To combat this problem, we present a method for projecting to the space of coefficients corresponding to consistent joint distributions (which we will refer to as the *marginal polytope*) during inference.

We begin by discussing the first-order version of the marginal polytope projection problem. Given an $n \times n$ matrix, M, of real numbers, how can we decide whether there exists some probability distribution which has M as its matrix of first-order marginal probabilities? A necessary and sufficient condition, as it turns out, is for M to be *doubly stochastic*. That is, all entries of M must be nonnegative and all rows and columns of M must sum to one (the probability that Alice is at *some track* is 1, and the probability that *some identity* is at Track 3 is 1). The double stochasticity condition comes from the *Birkhoff-von Neumann* theorem [131] which states that a matrix is doubly stochastic *if and only if* it can be written as a convex combination of permutation matrices.

To "renormalize" first-order marginals to be doubly stochastic, some authors [120, 121, 8, 49, 110] have used the *Sinkhorn iteration*, which alternates between normalizing rows and columns independently until convergence is obtained. Convergence is guaranteed under mild conditions and it can be shown that the limit is a nonnegative doubly stochastic matrix which is closest to the original matrix in the sense that the Kullback-Leibler divergence is minimized [8].

There are several problems which cause the Sinkhorn iteration to be an unnatural solution in our setting. First, since the Sinkhorn iteration only works for nonnegative matrices, we would have to first cap entries to lie in the appropriate range, $[0, 1]$. More seriously, even though the Sinkhorn iteration would guarantee a doubly stochastic higher order matrix of marginals, there are several natural constraints which are violated when running the Sinkhorn iteration on higher-order marginals. For example, with second-order (ordered) marginals, it seems that we should at least enforce the following symmetry constraint:

$$h(\sigma : \sigma(k, \ell) = (i, j)) = h(\sigma : \sigma(\ell, k) = (j, i)),$$

which says, for example, that the marginal probability that Alice is in Track 1 and Bob is in Track 2 is the same as the marginal probability that Bob is in Track 2 and Alice is in Track 1. Another natural constraint that can be

broken is what we refer to as *lower-order marginal consistency*. For example, it should always be the case that:

$$h(j) = \sum_i h(i,j) = \sum_k h(j,k).$$

It should be noted that the doubly stochastic requirement is a special case of lower-order marginal consistency — we require that higher-order marginals be consistent on the 0^{th} order marginal.

While compactly describing the constraints of the marginal polytope exactly remains an open problem, we propose a method for projecting onto a *relaxed* form of the marginal polytope which addresses both symmetry and lower-order consistency problems by operating directly on irreducible Fourier coefficients instead of on the matrix of marginal probabilities. After each conditioning step, we apply a 'correction' to the approximate posterior $h^{(t)}$ by finding the bandlimited function in the relaxed marginal polytope which is closest to $h^{(t)}$ in an L_2 sense. To perform the projection, we employ the Plancherel Theorem [29] which relates the L_2 distance between functions on S_n to a distance metric in the Fourier domain.

Proposition 61 (Plancherel Theorem)**.** *Let* f *and* g *be real-valued functions on a finite group* G*. Then the* L_2 *distance between* f *and* g *is given by:*

$$\sum_\sigma (f(\sigma) - g(\sigma))^2 \quad = \quad \frac{1}{|G|} \sum_\nu d_{\rho_\nu} Tr\left(\left(\hat{f}_{\rho_\nu} - \hat{g}_{\rho_\nu}\right)^T \cdot \left(\hat{f}_{\rho_\nu} - \hat{g}_{\rho_\nu}\right)\right). \tag{9.2}$$

To find the closest bandlimited function in the relaxed marginal polytope, we formulate a quadratic program whose objective is to minimize the right side of Equation 9.2, and whose sum is taken only over the set of maintained irreducibles, Λ, subject to the set of constraints which require all marginal probabilities to be nonnegative. We thus refer to our correction step as *Plancherel Projection*. Our quadratic program can be written as:

$$\begin{aligned}
&\text{minimize}_{\hat{h}^{proj}} \quad \sum_{\lambda \in \Lambda} d_\lambda \mathrm{Tr}\left[\left(\hat{h} - \hat{h}^{proj}\right)^T_{\rho_\lambda} \left(\hat{h} - \hat{h}^{proj}\right)_{\rho_\lambda} \right] \\
&\text{subject to:} \qquad \left[\hat{h}^{proj}\right]_{(n)} = 1, \\
&\qquad\qquad \left[C_{\lambda^{MIN}} \cdot \left(\bigoplus_{\mu \trianglerighteq \lambda^{MIN}} \bigoplus_{\ell=1}^{K_{\lambda^{MIN},\mu}} \hat{h}^{proj}_{\rho_\mu} \right) \cdot C^T_{\lambda^{MIN}} \right]_{ij} \geqslant 0, \qquad \text{for all } (i,j),
\end{aligned}$$

where $K_{\lambda^{MIN}}$ and $C_{\lambda^{MIN}}$ are the precomputed constants from Equation 5.10. We remark that even though the projection will produce a Fourier transform corresponding to nonnegative marginals which are consistent with each other, there might not necessarily exist a joint probability distribution on S_n consistent with those marginals except in the special case of first-order marginals.

Example 62. *In Example 56, we ran the Kronecker conditioning algorithm using all of the Fourier coefficients. If only the first-order coefficients are available,*

however, then the expressions for zeroth and first order terms of the posterior (Equations 8.13,8.14) become:

$$\widehat{f \cdot g}_{\rho_{(3)}} = \frac{1}{3!} \cdot \left[\hat{f}_{\rho_{(3)}} \cdot \hat{g}_{\rho_{(3)}} + 4 \cdot \left[A_{(2,1)\otimes(2,1)} \right]_{1,1} \right], \tag{9.3}$$

$$\widehat{f \cdot g}_{\rho_{(2,1)}} = \frac{1}{3!} \cdot \left[\hat{f}_{\rho_{(2,1)}} \cdot \hat{g}_{\rho_{(3)}} + \hat{f}_{\rho_{(3)}} \cdot \hat{g}_{\rho_{(2,1)}} + 2 \cdot \left[A_{(2,1)\otimes(2,1)} \right]_{2:3,2:3} \right], \tag{9.4}$$

Plugging in the same numerical values from Example 56 and normalizing appropriately yields the approximate Fourier coefficients of the posterior:

$$\left[\widehat{h(\sigma|z)} \right]_{\rho_{(3)}} = 1 \qquad \left[\widehat{h(\sigma|z)} \right]_{\rho_{(2,1)}} = \begin{bmatrix} -10/9 & -77/400 \\ 77/400 & 4/3 \end{bmatrix},$$

which correspond to the following first-order marginal probabilities:

$$\hat{h}_{\tau_{(2,1)}} \left[\begin{array}{c|ccc} & A & B & C \\ \hline \textit{Track 1} & 0 & 11/9 & -2/9 \\ \textit{Track 2} & 1 & 0 & 0 \\ \textit{Track 3} & 0 & -2/9 & 11/9 \end{array}\right].$$

In particular, we see that the approximate matrix of 'marginals' contains negative numbers. Applying the Plancherel projection step, we obtain the following marginals:

$$\hat{h}_{\tau_{(2,1)}} \left[\begin{array}{c|ccc} & A & B & C \\ \hline \textit{Track 1} & 0 & 1 & 0 \\ \textit{Track 2} & 1 & 0 & 0 \\ \textit{Track 3} & 0 & 0 & 1 \end{array}\right],$$

which happen to be exactly the true posterior marginals. It should be noted however, that rounding the 'marginals' to be in the appropriate range would have worked in this particular example as well.

9.4 EXPERIMENTS

In this section we present the results of several experiments to validate our approximate inference approach for identity management. We evaluate performance first by measuring the quality of our approximation for problems where the true distribution is known. Instead of measuring a distance between the true distribution and the inverse Fourier transform of our approximation, it makes more sense in our setting to measure error only at the marginals which are maintained by our approximation. In the results reported below, we measure the L_1 error between the true matrix of marginals and the approximation. If nonnegative marginal probabilities are guaranteed, it also makes sense to measure KL-divergence.

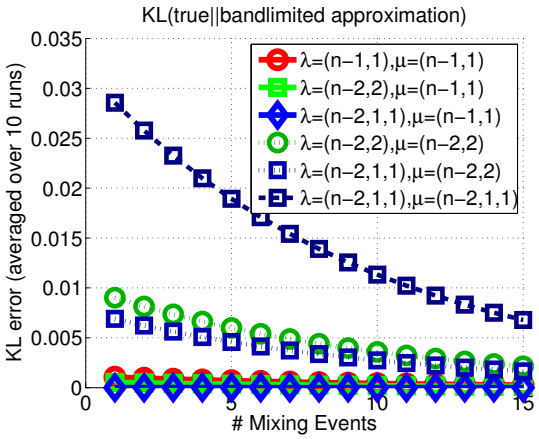


Figure 18: Kronecker Conditioning accuracy — we measure the accuracy of a single Kronecker conditioning operation after some number of mixing events.

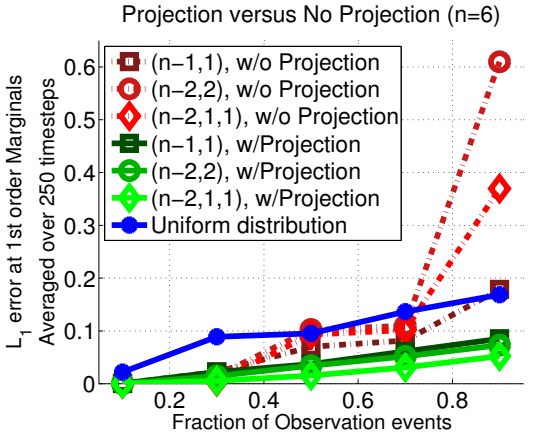
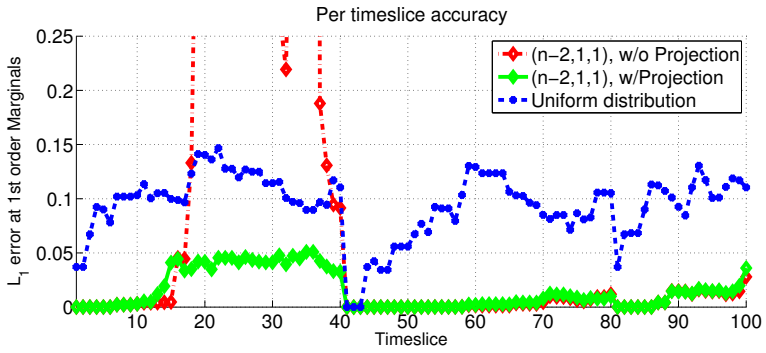


Figure 19: HMM accuracy — we measure the average accuracy of posterior marginals over 250 timesteps, varying the proportion of mixing and observation events.

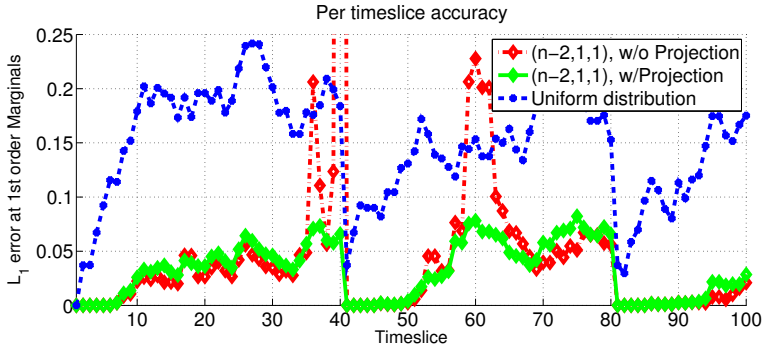
9.4.1 *Simulated data*

We first tested the accuracy of a single Kronecker conditioning step by calling some number of pairwise mixing events (which can be thought roughly as a measure of entropy), followed by a single first-order observation. In the y-axis of Figure 18, we plot the Kullback-Leibler divergence between the true first-order marginals and approximate first-order marginals returned by Kronecker conditioning. We compared the results of maintaining first-order, and second-order (unordered and ordered) marginals. As shown in Figure 18, Kronecker conditioning is more accurate when the prior is smooth and unsurprisingly, when we allow for higher order Fourier terms. As guaranteed by Theorem 60, we also see that the first-order terms of the posterior are exact when we maintain second-order (ordered) marginals.

To understand how our algorithms perform over many timesteps (where errors can propagate to all Fourier terms), we compared to exact inference



(a) $n = 6$ with 50% mixing events and 50% observations



(b) $n = 6$ with 30% mixing events and 70% observations

Figure 20: Accuracy as a function of time on two typical runs.

on synthetic datasets in which tracks are drawn at random to be observed or swapped. As a baseline, we show the accuracy of a uniform distribution. We observe that the Fourier approximation is better when there are either more mixing events (the fraction of conditioning events is smaller), or when more Fourier coefficients are maintained, as shown in Figure 19. We also see that the Plancherel Projection step is fundamental, especially when mixing events are rare.

Figures 20a and 20b show the per-timeslice accuracy of two typical runs of the algorithm. The fraction of conditioning events is 50% in Figure 20a, and 70% in Figure 20b. What we typically observe is that while the projected and nonprojected accuracies are often quite similar, the nonprojected marginals can perform significantly worse during certain segments.

Finally, we compared running times against an exact inference algorithm which performs prediction/rollup in the Fourier domain and conditioning in the primal domain. While the prediction/rollup step for pairwise mixing models can be implemented in $O(n!)$ time (linear in the size of the symmetric group), we show running times for the more general mixing models. Instead of the naive $O((n!)^2)$ complexity, its running time is a more efficient $O(n^3 n!)$ due to the Fast Fourier Transform [20]. It is clear that our algorithm scales gracefully compared to the exact solution (Figure 21), and in fact, we could not run exact inference for $n > 8$ due to memory constraints. In Figure 22, we show empirically that the Clebsch-Gordan coefficients are indeed sparse, supporting a faster conjectured runtime.

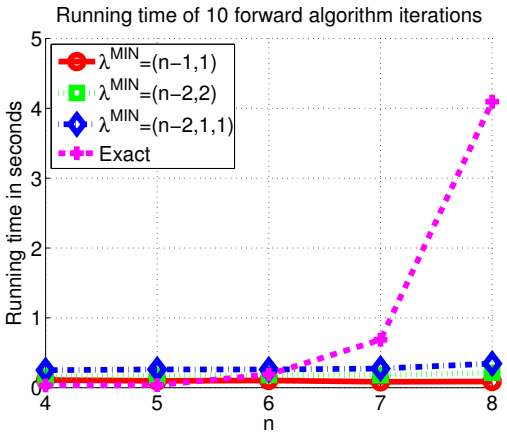


Figure 21: Running times: We compared running times of our polynomial time bandlimited inference algorithms against an exact algorithm with $O(n^3 n!)$ time complexity

9.4.2 *Identity management experiments*

We also evaluated our algorithm on data taken from a real network of eight cameras (Fig. 23a). In the data, there are $n = 11$ people walking around a room in fairly close proximity. To handle the fact that people can freely leave and enter the room, we maintain a list of the tracks which are external to the room. Each time a new track leaves the room, it is added to the list and a mixing event is called to allow for m^2 pairwise swaps amongst the m external tracks.

The number of mixing events is approximately the same as the number of observations. For each observation, the network returns a color histogram of the blob associated with one track. The task after conditioning on each observation is to predict identities for all tracks which are inside the room, and the evaluation metric is the fraction of accurate predictions. We compared against a baseline approach of predicting the identity of a track based on the most recently observed histogram at that track. This approach is expected to be accurate when there are many observations and discriminative appearance models, neither of which our problem afforded. As Figure 23b shows, both the baseline and first order model(without projection) fared poorly, while the projection step dramatically boosted the prediction accuracy for this problem. To illustrate the difficulty of predicting based on appearance alone, the rightmost bar reflects the performance of an *omniscient* tracker who knows the result of each mixing event and is therefore left only with the task of distinguishing between appearances. We conjecture that the performance of our algorithm (with projection) is near optimal.

9.5 CONCLUSION

In this chapter, we have presented an analysis of the errors which can accumulate in bandlimited inference operations and argued that Fourier

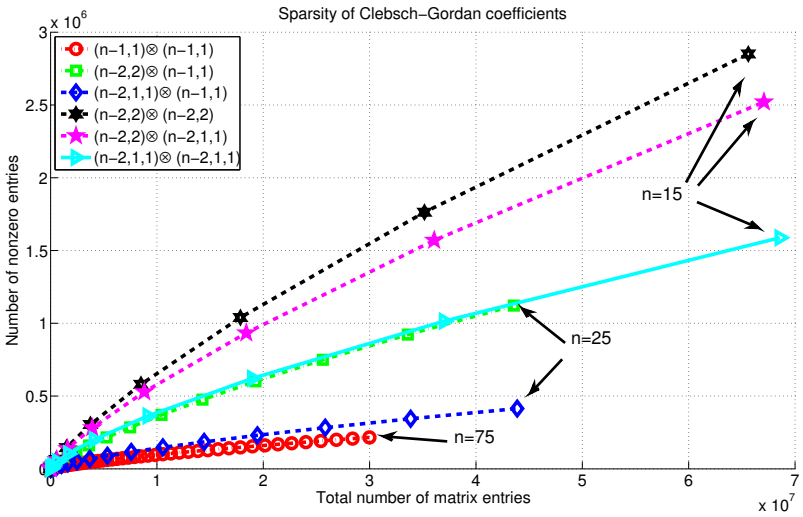


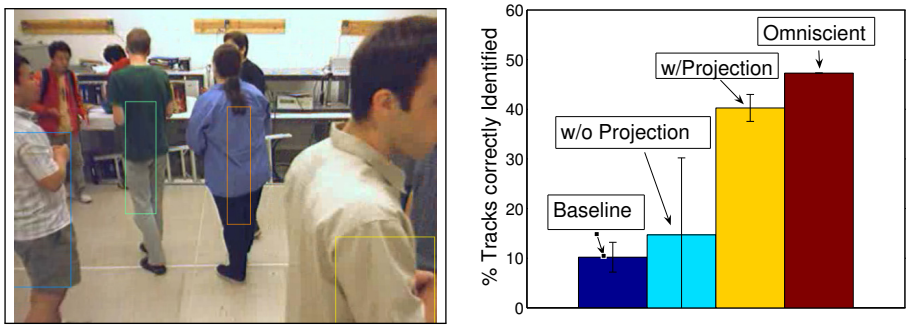
Figure 22: Clebsch-Gordan Sparsity: We measured the sparsity of the Clebsch-Gordan coefficients matrices by plotting the number of nonzero coefficients in a Clebsch-Gordan coefficient matrix against the number of total entries in the matrix for various n and pairs of irreducibles. For each fixed tensor product pair, we see that the number of nonzero entries scales sublinearly with respect to the total number of matrix elements.

based approaches work well when the underlying distributions are diffuse and are thus well approximated by low-frequency basis functions. While inference operations such as convolution, normalization and shifting are exact, problems can occur during conditioning in which errors in high-order terms due to bandlimiting can be propagated to lower-order terms.

We showed that it is typically faster and more accurate to condition on low-order observations which only involve a few items since their likelihood functions are sparser in the Fourier domain, often preventing error from propagating to the lower frequency levels (Theorem 60) Bandlimited conditioning can, on occasion, result in Fourier coefficients which correspond to no valid distribution. We showed, however, that the problem can be remedied by projecting to a relaxation of the marginal polytope.

Finally, our evaluation on data from a camera network shows that our methods perform well when compared to the optimal solution in small problems, or to an omniscient tracker in larger problems. Furthermore, we demonstrated that our projection step is fundamental in obtaining these high-quality results.

Algebraic methods have recently enjoyed a surge of interest in the machine learning community. We believe that our unified approach for performing probabilistic inference over permutations by means of representing distributions as additive combinations of Fourier basis functions, as well as our gentle exposition of group representation theory and noncommutative Fourier analysis will significantly lower the barrier of entry for machine learning researchers who are interested in using or further developing algebraically inspired algorithms which are useful for real-world problems.



(a) Sample Image (b) Accuracy for Camera Data

Figure 23: Evaluation on dataset from a real camera network.

10

RELATED WORK

RANKINGS and permutations have recently become an active area of research in machine learning due to their importance in information retrieval and preference elicitation. Rather than considering full distributions over permutations, many approaches, like RankSVM (Joachims [68]) and RankBoost (Freund et al. [38]), have instead focused on learning a single 'optimal' ranking with respect to some objective function.

There are also several authors (from both the statistics and machine learning communities) who have studied distributions over permutations/rankings (Mallows [91], Critchlow [26], Fligner and Verducci [36], Meila et al. [97], Taylor et al. [126], Lebanon and Mao [86]). Taylor et al. [126] consider distributions over S_n which are induced by the rankings of n independent draws from n individually centered Gaussian distributions with equal variance. They compactly summarize their distributions using an $O(n^2)$ matrix which is conceptually similar to our first-order summaries and apply their techniques to ranking web documents. Most other previous approaches at directly modeling distributions on S_n, however, have relied on distance based exponential family models. For example, the Mallows model [91] defines a Gaussian-like distribution over permutations as:

$$h(\sigma; c, \sigma_0) \propto \exp\left(-c\, d(\sigma, \sigma_0)\right), \tag{10.1}$$

where the function $d(\sigma, \sigma_0)$ is the *Kendall's tau distance* which counts the number of adjacent swaps that are required to bring σ^{-1} to σ_0^{-1}. We will discuss Mallows models in greater detail in Part III.

As we have shown in Part II, Fourier based methods (Diaconis [29], Kondor et al. [80], Huang et al. [57]) offer a principled alternative method for compactly representing distributions over permutations and performing efficient probabilistic inference operations. Our work draws from two strands of research — one from the data association/identity management literature, and one from a more theoretical area on Fourier analysis in statistics. In the following, we review several of the works which have led up to our current Fourier based approach.

10.0.1 *Previous work in identity management*

The identity management problem has been addressed in a number of previous works, and is closely related to, but not identical with, the classical data association problem of maintaining correspondences between tracks and observations. Both problems need to address the fundamental combinatorial challenge that there is a factorial or exponential number of associations to maintain between tracks and identities, or between tracks and observations respectively. A vast literature already exists on the the

data association problem, beginning with the *multiple hypothesis testing* approach (MHT) of Reid [112]. The MHT is a 'deferred logic' method in which past observations are exploited in forming new hypotheses when a new set of observations arises. Since the number of hypotheses can grow exponentially over time, various heuristics have been proposed to help cope with the complexity blowup. For example, one can choose to maintain only the k *best* hypotheses for some parameter k (Cox and Hingorani [25]), using Murty's algorithm [103]. But for such an approximation to be effective, k may still need to scale exponentially in the number of objects. A slightly more recent filtering approach is the *joint probabilistic data association filter* (JPDA) [9], which is a suboptimal single-stage approximation of the optimal Bayesian filter. JPDA makes associations sequentially and is unable to correct erroneous associations made in the past [109]. Even though the JPDA is more efficient than the MHT, the calculation of the JPDA association probabilities is still a #P-complete problem [21], since it effectively must compute matrix permanents. Polynomial approximation algorithms to the JPDA association probabilities have recently been studied using Markov chain Monte Carlo (MCMC) methods [106, 105].

The identity management problem was first explicitly introduced in Shin et al. [120]. Identity management differs from the classical data association problem in that its observation model is not concerned with the low-level tracking details but instead with high level information about object identities. Shin et al. [120] introduced the notion of the *belief matrix* approximation of the association probabilities, which collapses a distribution over all possible associations to just its first-order marginals. In the case of n tracks and n identities, the belief matrix B is an $n \times n$ doubly-stochastic matrix of non-negative entries b_{ij}, where b_{ij} is the probability that identity i is associated with track j. As we already saw in Chapter 5, the belief matrix approximation is equivalent to maintaining the zeroth- and first-order Fourier coefficients. Thus our current work is a strict generalization and extension of those previous results.

An alternative representation that has also been considered is an information theoretic approach (Shin et al. [121], Schumitsch et al. [115, 116]) in which the density is parameterized as:

$$h(\sigma; \Omega) \propto \exp \operatorname{Tr}\left(\Omega^T \cdot \tau_{(n-1,1)}(\sigma)\right).$$

In our framework, the information form approach can be viewed as a method for maintaining the Fourier transform of the *log* probability distribution at only the first two irreducibles. The information matrix approach is especially attractive in a distributed sensor network setting, since, if the columns of the information matrix are distributed to leader nodes tracking the respective targets, then the observation events become entirely local operations, avoiding the more expensive Kronecker conditioning algorithm in our setting. On the other hand, the information matrix coefficients do not have the same intuitive marginals interpretation afforded in our setting, and moreover, prediction/rollup steps cannot be performed analytically in the information matrix form. Normalization is also difficult in the in-

formation domain as it involves computing the matrix permanent of the matrix $\exp(\Omega)$, where the exponentiation is pointwise, and is consequently a #P-hard problem.

As in many classical data structures problems there are representational trade-off issues: some operations are less expensive in one representation and some operations in the the other. The best choice in any particular scenario will depend on the ratio between observation and mixing events. Recently, in Jiang et al. [67], we propose a maximum entropy based method for converting between the Fourier-based representation and the information representation. Taking a hybrid approach of both representations, we switch between the Fourier and information domains during inference depending on the ratio of observation to mixing events, balancing between algorithmic efficiency and approximation quality.

10.0.2 *Previous work on Fourier-based approximations*

The concept of using Fourier transforms to study probability distributions on groups is not new, with the earliest papers in this area having been published in the 1960s by Grenander [45]. Willsky [136] was the first to formulate the exact filtering problem in the Fourier domain for finite and locally compact Lie groups and contributed the first noncommutative Fast Fourier Transform algorithm (for metacyclic groups). However, he does not address approximate inference, suggesting instead to always transform to the appropriate domain for which either the prediction/rollup or conditioning operations can be accomplished using a pointwise product. While providing significant improvements in complexity for smaller groups, his approach is still infeasible for our problem given the factorial order of the symmetric group.

Diaconis [29] utilized the Fourier transform to analyze probability distributions on the symmetric group in order to study card shuffling and ranking problems. His work laid the ground for much of the progress made over the last two decades on probabilistic group theory and noncommutative FFT algorithms (Clausen and Baum [20], Rockmore [113]).

Kondor et al. [80] was the first to show that the data association problem could be efficiently approximated using FFT factorizations. In contrast to our framework where every model is assumed to be have been specified in the Fourier domain, they work with an observation model which can be written as the indicator function of cosets of subgroups of the form $S_k \subset S_n$.

Conceptually, one might imagine formulating a conditioning algorithm which applies the Inverse Fast Fourier Transform (IFFT) to the prior distribution, conditions in the primal domain using pointwise multiplication, then transforms back up to the Fourier domain using the FFT to obtain posterior Fourier coefficients. While such a procedure would ordinarily be intractable because of the factorial number of permutations, [80] elegantly shows that for certain coset-based observation models, it is not necessary to perform the full FFT recursion to do a pointwise product. They exploit

this observation to formulate an efficient conditioning algorithm whose running time depends on the complexity of the observation model (which can roughly be measured by the number of irreducibles required to fully specify it).

Our work generalizes the conditioning formulation from [80] in the sense that it can work for *any* observation model and extends easily to similar filtering problems over any finite group. In the case that the observation model is specified at sufficiently many irreducibles, our conditioning algorithm (prior to the projection step) returns the same approximate probabilities as the FFT-based algorithm. For example, we can show that the observation model given in Equation 7.12 is fully specified by two Fourier components, and that both algorithms have identical output. Additionally, [80] do not address the issue of projecting onto legal distributions, which, as we show in our experimental results is fundamental in practice.

Finally, over the last four years since both our paper (Huang et al. [57]) and Kondor et al. [80] were first published in the machine learning community, there have been a number of related papers on Fourier analysis for permutations or similar ideas, including Kondor and Borgwardt [79], Jagabathula and Shah [64], Guibas [47], Huang et al. [60], Huang and Guestrin [54], Kondor [78], Kondor and Barbosa [81], Kakarala [69], Jiang et al. [67], Huang and Guestrin [56]. Notably, Risi Kondor has published a number of works on the application of Clausen's FFT based methods for graph theoretic problems, such as computing compact graph descriptors ([79]) and efficient optimization for quadratic assignment problems ([78]). Most recently, Wimmer [137] has applied Fourier analytic techniques in a learning theoretic setting.

Part III

MULTIPLICATIVE DECOMPOSITIONS: PROBABILISTIC INDEPENDENCE BASED REPRESENTATIONS AND INFERENCE

11

OVERVIEW OF PART III: MULTIPLICATIVE DECOMPOSITIONS

In part of II of this thesis, we have explored additive Fourier-based decompositions for addressing the probabilistic representation and inference problem for permutations. Inspired by the popular graphical models literature in machine learning, we consider in part III, *multiplicative decompositions* of distributions over permutations.

Many probabilistic computations become greatly simplified when dealing with distributions which factor into a product of noninteracting distributions. For example, the popular naive Bayes assumption for classification, which assumes that the features in a problem are fully independent conditioned on a class label, allow one to develop fast classification algorithms and to learn parameters with low sample complexity. Generalizations of the naive Bayes model, such as Bayesian networks and Markov random fields allow one to capture the similar computational benefits with increased model expressiveness.

Taking advantage of probablistic independence with permutation data, however, is not as simple as it is with other data types, due to the mutual exclusivity assumptions for permutations which disallow items from mapping to the same target (thus inducing an explicit dependence between every pair of items). In part III we discuss such issues which are specific to the symmetric group in detail, and in particular, part of our focus in Part III lies in understanding the interaction between additive and multiplicative decompositions. The following is an outline of the main contributions as wel as a roadmap of the chapters ahead in Part III.

- In Chapter 12, we examine the simplest case of a multiplicative decomposition, in which subsets of items are fully independent of each other, in the probabilistic sense, and characterize the constraints on the Fourier coefficients of a distribution in which there exists independence structure.

  We develop algorithms for detecting independence using Fourier coefficients, allowing one to exploit both additive and multiplicative decomposition for the purposes of scalable representation and inference. We apply our algorithms to track large numbers of objects by adaptively finding independent subgroups of objects and factoring the distribution appropriately.

- Unfortunately, while full independence can sometimes be a reasonable assumption for tracking problems, it is rarely so for ranking. In Chapter 13 we introduce an intuitive generalization of the notion of probabilistic independence for permutations, *riffled independence*, based on interleaving independent rankings of subsets of items, as

if one were shuffling two piles of cards together. We show our generalized notion of riffled independence to be a more appropriate version of independence for ranked data and exhibit evidence that riffled independence relations can approximately hold in real ranked datasets.

We also discuss the problem of estimating parameters of a riffle independent model from ranking data. To perform such computations in a scalable way, we develop algorithms that can be used in the Fourier-theoretic framework of Part II for joining riffle independent factors (*RiffleJoin*), and for teasing apart the riffle independent factors from a joint (*RiffleSplit*), and provide theoretical and empirical evidence that our algorithms perform well.

- In Chapter 14, we provide an efficient, automated method for discovering sets of items which are riffle independent from a training set of rankings. We show that our clustering-like algorithms can be used to discover meaningful latent coalitions from real preference ranking datasets and to learn the structure of hierarchically decomposable models based on riffled independence.

- Where Chapters 13 and 14 are primarily concerned with *representation*, Chapter 15 addresses the problem of *inference*, and in particular, it discusses the question of conditioning a riffle independent prior distribution.

  In Part II, we showed that it is more efficient and more accurate to condition on low-order observations in the Fourier domain. The same insights do not apply for riffle independent representations. We show instead that one can exploit riffle independent structure of a prior distribution to efficiently condition on observations which take the form of *partial rankings*.

  Using these ideas for efficient inference, we propose an algorithm that is capable of efficiently estimating the structure and parameters of riffle independent models from heterogeneous collections of partially ranked data. We apply our methods to real voting and preference data evidencing the effectiveness of our methods.

The contributions of Part III have also appeared in publication in the following articles:

[1] Jonathan Huang, Carlos Guestrin, Xiaoye Jiang, and Leonidas J. Guibas. Exploiting probabilistic independence for permutations. *Journal of Machine Learning Research - Proceedings Track*, 5:248–255, 2009.

[2] Jonathan Huang and Carlos Guestrin. Riffled independence for ranked data. In Yoshua Bengio, Dale Schuurmans, John Lafferty, Chris K. I. Williams, and Aron Culotta, editors, *Advances in Neural Information Processing Systems 22*, NIPS '08, pages 799–807. 2009.

[3] Jonathan Huang and Carlos Guestrin. Learning hierarchical riffle independent groupings from rankings. In *Proceedings of the 27th Annual International Conference on Machine Learning*, ICML '10, pages 455–462, Haifa, Israel, 2010.

[4] Jonathan Huang and Carlos Guestrin. Uncovering the riffled independence structure of ranked data. *http://arxiv.org/abs/1006.1328*, 2011.

[5] Jonathan Huang, Ashish Kapoor, and Carlos Guestrin. Efficient probabilistic inference with partial ranking queries. In *The 27th Conference on Uncertainty in Artificial Intelligence*, UAI '11, Barcelona, Spain, July 2011.

12

FULLY INDEPENDENT DECOMPOSITIONS

In Part II, we showed how low frequency Fourier coefficients capture intuitively interpretable marginals, and presented general algorithms for efficient approximate inference operations, such as marginalization and conditioning, which can be performed completely in the Fourier domain. Unfortunately, as we have discussed in previous chapters, the Fourier based approach still suffers from two shortcomings:

- While low frequency Fourier coefficients provide a principled approximation to the underlying distribution and only require storing polynomially many numbers, the polynomials can grow quite fast for practical applications.
- In a phenomenon reminiscent of the Heisenberg uncertainty principle, bandlimited approximations which discard high frequencies are most effective with diffuse distributions since smooth functions tend to be well approximated by linear combinations of low frequency basis functions, but are less effective at approximating highly peaked distributions.

In a sense, these two shortcomings are at odds with each other since we can always achieve better approximations to sharp functions by maintaining higher frequency Fourier coefficients. However, an interesting observation is that when the distribution is sharp, it would make more sense to break up the problem into smaller parts and to reason about disjoint subsets of objects independently of each other.

Consider again the *identity management* problem. If we are completely uncertain about the assignment of people to tracks, and have a uniform distribution over permutations, this smooth distribution can be represented with only one nonzero parameter in the Fourier domain. At the limit when we know the location of every identity, our distribution becomes very peaked, and we may need to maintain $n!$ nonzero Fourier coefficients. In this peaked setting, however, there is no reason to track all n identities jointly, and we can break up the problem into n subproblems. In this chapter, we propose a principled method based on exploiting probabilistic independence which tackles both issues by trying to "get the best of both worlds".

The main contributions of this chapter are:

- We characterize the constraints on the Fourier coefficients of a distribution over permutations implied by probabilistic independence.
- We present a number of simple algorithms which operate entirely in the Fourier domain for combining factors to form a joint distribution and factoring a distribution, respectively. Our algorithms are

fully general in the sense that they work for any distribution over permutations.

- We prove theoretical results showing how many Fourier terms are required in our JOIN /SPLIT algorithms to achieve a desired number of Fourier terms in the result, and analyze the behaviour of our algorithms in near-independent situations in which a distribution does not fully factor.
- We discuss a method for detecting probabilistic independence using the Fourier coefficients of a distribution.
- We use our algorithms to adaptively decompose large identity management problems into much smaller ones, improving previous methods both in scalability and approximation quality.

12.1 INDEPENDENCE ON THE SYMMETRIC GROUP AND FIRST-ORDER CONDITIONS

While band-limiting our representation can decrease the storage cost from $O(n!)$ to some polynomial in n, maintaining the s^{th}-order marginals requires, in the worst-case, $O(n^{2s})$ space. Thus, for small n we can maintain higher order coefficients (larger s), but this representation quickly becomes intractable as n becomes large. Over the next sections, we will show how probabilistic independence is manifested in the Fourier coefficients of a distribution, and how, by exploiting this independence, we can break our distribution into smaller subgroups, which allow us to maintain higher order coefficients.

In this section, we begin by defining independence and discussing a simple condition on the matrix of first-order marginal probabilities implied by independence.

Definition 63. Consider the set $A_{in} = \{1, \ldots, p\} \subset \{1, \ldots, n\}$ and its complement $B_{in} = \{p+1, \ldots, n\}$. $\sigma(A_{in}) = (\sigma(1), \ldots, \sigma(p))$ and $\sigma(B_{in}) = (\sigma(p+1), \ldots, \sigma(n))$ are said to be *independent* under a distribution h over permutations of $\{1, \ldots, n\}$ if h factors as

$$h(\sigma) = f(\sigma(A_{in})) \cdot g(\sigma(B_{in})).$$

The function f, for example, is a marginal distribution for mappings of $\{1, \ldots, p\}$ into the set $\{1, \ldots, n\}$. As we shall see in the next section, however, we will in fact be able work with f as if it were a function on S_p due to mutual exclusivity constraints (and similarly, we will be able to work with g as if it were a function on S_{n-p}).

As an example, if $\sigma = [3, 4, 2, 1] \in S_4$ and $A_{in} = \{1, 2\}$ and $B_{in} = \{3, 4\}$ are independent, then $h(\sigma) = h(\sigma([1, 2]) = [3, 4]) \cdot h(\sigma([3, 4]) = [2, 1])$. We will refer to A_{in} and B_{in} as *cliques* since the variables of A_{in} and B_{in} form disjoint cliques in the graphical model representation of the above independence relation. Though in general we will want to consider

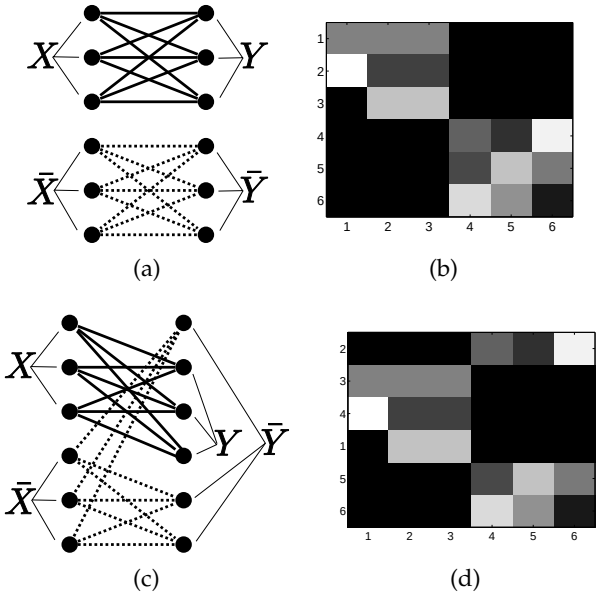


Figure 24: Examples of first-order independence: In (a) and (c), we show how the identities and tracks can be partitioned into disjoint subsets, A_{in}, B_{in}, A_{out}, and B_{out}. (b) and (d), show an example of what the corresponding first-order marginals would look like.

problems in which the subset A_{in} is arbitrary, we focus on $A_{in} = \{1, \ldots, p\}$ for now and defer the discussion of arbitrary splits until Section 12.4. In this section, we discuss a simple first-order criterion for independence on the symmetric group and show that it naturally leads us to study functions over product groups of the form $S_p \times S_q$, where $p + q = n$.

12.1.1 *First-order conditions*

Due to the mutual exclusivity constraints associated with permutations, a necessary (but not sufficient) condition for a distribution h on permutations to factor into a product of factors over A_{in} and B_{in} is that there must exist a subset $A_{out} \subset \{1, \ldots, n\}$ of the same size as A_{in} such that, with probability 1, elements of A_{in} map to A_{out} and elements of B_{in} map to $B_{out} = \{1, \ldots, n\} \setminus Y$. We will refer to the above condition as the *first-order independence criterion*. Intuitively, a distribution can only factor into independent parts if the set $\{1, \ldots, n\}$ can be partitioned into disjoint subsets of objects which do not interact with one another. See Figures 24a and 24c for some examples.

Lemma 64 (First-order independence condition)**.** *If* $\sigma(A_{in})$ *and* $\sigma(B_{in})$ *are independent under the distribution* h*, then there exists a subset* $A_{out} \subset \{1, \ldots, n\}$ *with* $|A_{out}| = |A_{in}|$ *such that* $h(\sigma) = 0$ *unless* $\sigma(A_{in}) \subset A_{out}$.[1]

1 With some abuse of notation, we use $\sigma(A_{in}) \subset A_{out}$ to denote the fact that the permutation σ maps elements A_{in} (people) to elements of A_{out} (tracks).

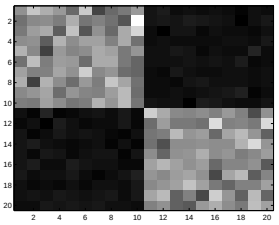


Figure 25: Approximate first-order independence: In practice, we expect first-order independence to only hold approximately.

Proof. Define the sets: $A_{out} = \{k : h(\sigma(i) = k) > 0 \text{ for some } i \in A_{in}\}$, and $Z = \{k : h(\sigma(j) = k) > 0 \text{ for some } j \in B_{in}\}$. By construction, $h(\sigma) = 0$ unless $\sigma(A_{in})$ is a permutation of elements in Y (and $\sigma(B_{in})$ is some permutation of elements in Z), so we need only show that $|A_{out}| = |A_{in}|$. By mutual exclusivity, $|A_{in}| \leqslant |A_{out}|$ and $|B_{in}| \leqslant |Z|$. We now show that $A_{out} \cap Z = \emptyset$, which will imply that $|A_{out}| = |A_{in}|$. Suppose that there exists some $k \in A_{out} \cap Z$. Then by the definitions of A_{out} and Z, there exists $i \in A_{in}$ and $j \in B_{in}$ such that both $h(\sigma(i) = k) > 0$ and $h(\sigma(j) = k) > 0$. However, by mutual exclusivity, $h(\sigma(i) = k \text{ and } \sigma(j) = k) = 0$, and by independence, we see that $h(\sigma(i) = k)h(\sigma(j) = k) = 0$, thus arriving at a contradiction since we assumed that neither $h(\sigma(i) = k)$ nor $h(\sigma(j) = k)$ is equal to zero. □

To see why the first-order independence condition is an insufficient indicator of independence, consider the simple example of a distribution on S_4 which always maps the set $A_{in} = \{1, 2\}$ to $A_{out} = \{1, 2\}$ and the set $B_{in} = \{3, 4\}$ to $B_{out} = \{3, 4\}$, but is constrained to map 1 to 1 whenever 3 maps to 3. In this case, the first-order marginals exhibit independence, but the distribution is not independent when we examine the higher order components.

Despite its insufficiency, the first-order independence condition plays a crucial role for us in several ways. As we discuss later, it can serve as a first pass at detecting independence as it reduces the detection problem into a clustering-like problem. On a somewhat more theoretical level, it also suggests that we should be thinking about groups of the form $S_p \times S_q \subset S_n$, where $|A_{in}| = |A_{out}| = p$ and $|B_{in}| = |B_{out}| = n - p = q$.

In particular, restated using group theoretic terminology, the first-order condition tells us that if a distribution h factors independently over A_{in} and B_{in}, then it must be supported on a left $S_p \times S_q$–coset — we will discuss the implications of this observations in Section 12.3, in which we derive Join and Split algorithms.

12.2 FOURIER THEORETIC CHARACTERIZATIONS OF PROBABILISTIC INDEPENDENCE

In this section, we return to probabilistic independence and generalize the results of Section 12.1.1 to hold for higher-order Fourier terms. Since we maintain the Fourier coefficients of the distribution instead of the actual distribution, there are several technical challenges associated with (1) splitting a distribution into independent factors, (2) joining independent factors to form a joint distribution, and (3) detecting independent subsets in the Fourier domain. To solve these problems, we will relate the Fourier coefficients of the joint distribution h with the Fourier coefficients of the factors f and g.

In Part III, we will use σ_p and σ_q to refer to permutations in S_p and S_q respectively. However, the notation $[\sigma_p, \sigma_q]$ will (abusively) denote the following permutation in S_n:

$$[\sigma_p, \sigma_q] = [\sigma_p(1), \ldots, \sigma_p(p), \sigma_q(1) + p, \ldots, \sigma_q(q) + p]. \tag{12.1}$$

Thus for $\sigma_p = [1,2,3]$ and $\sigma_q = [3,2,1]$, the $[\sigma_p, \sigma_q]$ is the permutation $[1,2,3,6,5,4]$ (instead of $[1,2,3,3,2,1]$, which is not a legitimate permutation). The subset of permutations in S_n which can be written as $\sigma = [\sigma_p, \sigma_q]$ is $S_p \times S_q$ (see Equation 12.1). Elements in $S_p \times S_q \subset S_n$ permute the indices within each subset, $\{1, \ldots, p\}$ and $\{p+1, \ldots, n\}$, amongst themselves but never across subsets.

In this section we focus on the special case when $A_{in} = A_{out} = \{1, \ldots, p\}$ (we show how to deal with general A_{in} and A_{out} when we discuss the detection step). We therefore assume that:

$$h(\sigma) = \begin{cases} f(\sigma_p)\, g(\sigma_q) & \text{if } \sigma = [\sigma_p, \sigma_q] \in S_n \\ 0 & \text{otherwise} \end{cases}, \tag{12.2}$$

for distributions $f : S_p \to \mathbb{R}$ and $g : S_q \to \mathbb{R}$.

To formulate our algorithms, we will relate the Fourier coefficients of the joint distribution h to the Fourier coefficients of the individual factors f and g. The main contribution of this section is a structural result showing that the Fourier coefficient matrices of a distribution which factors independently decomposes in a particular way.

Before beginning, we remind the reader of how the first-order matrix of marginals for the joint distribution h must decompose. Under the assumption that h factors independently with $A_{in} = A_{out} = \{1, \ldots, p\}$ (and $B_{in} = B_{out} = \{p+1, \ldots, n\}$), the first-order condition tells us that $h(\sigma(i) = j) = 0$ whenever $i > p$ and $j \leqslant p$, and vice versa. We can therefore conclude that the first-order matrix decomposes as a *block-diagonal* matrix with two blocks of sizes p and q respectively (see Figure 24b). With higher-order Fourier coefficient matrices, the decomposition becomes more complicated. As we see in Section 12.2.1, in addition to a block-diagonal decomposition with respect to a certain basis, there is also Kronecker product structure within each block.

12.2.1 *Higher-order characterizations*

To characterize how higher-order Fourier terms must decompose, our story begins with a brief excursion into the representation theory of groups of the form $S_p \times S_q$. While the permutation $\sigma = [\sigma_p, \sigma_q]$ can be seen as an element of S_n, it can additionally be viewed as an element of the subgroup $S_p \times S_q$. Thus, evaluated at an element $\sigma \in S_p \times S_q$, any irreducible, ρ_λ of S_n, can also be viewed as a representation of the group $S_p \times S_q$. As a representation of $S_p \times S_q$, however, ρ_λ is not necessarily irreducible, but it can be related to the irreducibles of $S_p \times S_q$ using Equation 5.4, as we show in the following.

12.2.2 *Littlewood Richardson decomposition*

But what are the irreducibles of $S_p \times S_q$? We use a standard representation theoretic result that the set of irreducibles of a direct product of two groups $H \times K$ is exactly the set of all pairwise *tensor (or Kronecker) products* of irreducible representations of H and K. To be precise, we will provide a formal definition of Kronecker product representations.

Definition 65. Suppose ρ_λ is a degree ℓ representation of S_p and ρ_μ a degree m representation of S_q. The *Kronecker product representation* is the function $\rho_\lambda \otimes \rho_\mu$ mapping elements in $S_p \times S_q$ to $\ell m \times \ell m$ matrices defined by:

$$\rho_\lambda \otimes \rho_\mu([\sigma_p, \sigma_q]) \equiv \rho_\lambda(\sigma_p) \otimes \rho_\mu(\sigma_q),$$

where $\sigma_p \in S_p$ and $\sigma_q \in S_q$.

Note that while this definition of the Kronecker product representation uses the same Kronecker product as that of Equation 8.6, the representations are distinct since in the first case (from Chapter 8), one obtains a new representation of S_n whereas here, we obtain a representation of $S_p \times S_q$. It is a standard result that the Kronecker product representation is *indeed* a representation of the group $S_p \times S_q$, and moreover, the complete set of irreducible representations of $S_p \times S_q$ is exactly the set: $\{\rho_\mu \otimes \rho_\nu\}$, where μ and ν range over partitions of p and q, respectively.

Example 66. *As an example, there are 9 irreducible representations of the product group $S_3 \times S_3$ since there are three irreducibles of S_3 (each corresponding to one of the three partitions, (3), $(2,1)$ and $(1,1,1)$) and thus the set of irreducible representations, and their dimensions (denoted by d) are:*

$$
\begin{aligned}
\rho_{(3)} \otimes \rho_{(3)} \quad & (d = 1 \cdot 1 = 1) \\
\rho_{(3)} \otimes \rho_{(2,1)} \quad & (d = 1 \cdot 2 = 2) \\
\rho_{(3)} \otimes \rho_{(1,1,1)} \quad & (d = 1 \cdot 1 = 1) \\
\rho_{(2,1)} \otimes \rho_{(3)} \quad & (d = 2 \cdot 1 = 2) \\
\rho_{(2,1)} \otimes \rho_{(2,1)} \quad & (d = 2 \cdot 2 = 4) \\
\rho_{(2,1)} \otimes \rho_{(1,1,1)} \quad & (d = 2 \cdot 1 = 2) \\
\rho_{(1,1,1)} \otimes \rho_{(3)} \quad & (d = 1 \cdot 1 = 1) \\
\rho_{(1,1,1)} \otimes \rho_{(2,1)} \quad & (d = 1 \cdot 2 = 2) \\
\rho_{(1,1,1)} \otimes \rho_{(1,1,1)} \quad & (d = 1 \cdot 1 = 1)
\end{aligned}
$$

representation ρ_λ, of S_n, when evaluated at a permutation $\sigma = [\sigma_p, \sigma_q]$ which lies in the subgroup $S_p \times S_q$, therefore has the following decomposition by Equation 5.4:

$$
L^{\lambda}_{\mu\nu} \cdot \rho_\lambda(\sigma) \cdot {L^{\lambda}_{\mu\nu}}^{T} = \bigoplus_{\mu,\nu} \bigoplus_{\ell=1}^{c^{\lambda}_{\mu,\nu}} \rho_\mu(\sigma_p) \otimes \rho_\nu(\sigma_q). \tag{12.3}
$$

The coupling matrix $L^{\lambda}_{\mu\nu}$, along with the multiplicities $c^{\lambda}_{\mu,\nu}$ are assumed to be precomputed (see Appendices C and D). The following Proposition gives the desired relation between the Fourier coefficients of the joint and the Fourier coefficients of the factors.

Proposition 67. *Given the Fourier coefficients of two independent factors* f *and* g*, the Fourier coefficient matrices of the joint distribution* h*, are:*

$$
\widehat{h}_{\rho_\lambda} = {L^{\lambda}_{\mu\nu}}^{T} \cdot \bigoplus_{\mu,\nu} \bigoplus_{\ell=1}^{c^{\lambda}_{\mu,\nu}} \left(\widehat{f}_{\rho_\mu} \otimes \widehat{g}_{\rho_\nu} \right) \cdot L^{\lambda}_{\mu\nu}. \tag{12.4}
$$

Proposition 67 is significant because it completely characterizes the form of the Fourier matrices of the joint distribution at all frequencies. Recalling that (Lemma 64) the first-order marginals are constrained to be block diagonal, we see (ignoring the change of basis, which does not depend on f or g) that Equation 12.4 in fact imposes block diagonal structure on the Fourier matrices at all orders. Additionally, we see that each nonzero block has Kronecker structure at higher orders and that the coefficients of the joint are redundant in the sense that information at lower frequencies of the factors f and g are duplicated to multiple higher frequencies of h. The proof of the proposition is as follows.

Proof of Proposition 67.

$$
L^{\lambda}_{\mu\nu} \cdot \left[\widehat{h}\right]_{\rho_\lambda} \cdot {L^{\lambda}_{\mu\nu}}^{T} = L^{\lambda}_{\mu\nu} \cdot \left(\sum_{\sigma \in S_n} h(\sigma) \rho_\lambda(\sigma) \right) \cdot {L^{\lambda}_{\mu\nu}}^{T},
$$

(Definition of Fourier transform)

$$= \sum_{\sigma \in S_p \times S_q} h([\sigma_p, \sigma_q]) \left(L^{\lambda}_{\mu\nu} \cdot \rho_\lambda(\sigma) \cdot {L^{\lambda}_{\mu\nu}}^{T} \right), \quad (12.5)$$

(h is supported on $S_p \times S_q$, linearity)

$$= \sum_{\sigma \in S_p \times S_q} h([\sigma_p, \sigma_q]) \left(\bigoplus_{\mu,\nu} \bigoplus_{\ell=1}^{c^{\lambda}_{\mu,\nu}} \rho_\mu(\sigma_p) \otimes \rho_\nu(\sigma_q) \right),$$

(Equation 5.2) (12.6)

$$= \bigoplus_{\mu,\nu} \bigoplus_{\ell=1}^{c^{\lambda}_{\mu,\nu}} \left(\sum_{\sigma \in S_p} f(\sigma_p) \rho_\mu(\sigma_p) \right) \otimes \left(\sum_{\sigma \in S_q} g(\sigma_q) \rho_\nu(\sigma_q) \right), \quad (12.7)$$

($h = f \cdot g$, Bilinearity of $\otimes$)

$$= \bigoplus_{\mu,\nu} \bigoplus_{\ell=1}^{c^{\lambda}_{\mu,\nu}} (\widehat{f}_{\rho_\mu} \otimes \widehat{g}_{\rho_\nu}), \quad (12.8)$$

(Definition of Fourier transform).

We remark that the independence assumption in Proposition 67 is *necessary* and Equation 12.4 does *not* hold for arbitrary functions even if they are zero outside of $S_p \times S_q$. ■

As it turns out, the multiplicities, $c^{\lambda}_{\mu,\nu}$, are equivalent to what mathematicians have studied in different contexts as *Littlewood–Richardson (LR)* coefficients [114]. The LR coefficients tell us which cross-terms contribute to the joint. For example, it can be shown that that first order terms corresponding to the partition $(n-1,1)$ can be reconstructed using only three terms, $(p) \otimes (q)$, $(p-1,1) \otimes (q)$, and $(p) \otimes (q-1,1)$. In the following formulas, we use the notation $\rho_\lambda \downarrow_{S_p \times S_q}$ to denote decompositions which are only guaranteed to hold when ρ_λ is evaluated at permutations which are members of the subgroup $S_p \times S_q$. The first order decomposition is written as:

$$\rho_{(n-1,1)} \downarrow_{S_p \times S_q} \equiv (\rho_{(p)} \otimes \rho_{(q)}) \oplus (\rho_{(p-1,1)} \otimes \rho_{(q)}) \oplus (\rho_{(p)} \otimes \rho_{(q-1,1)}),$$

where we have suppressed the notation for the basis transforms, $L^{\lambda}_{\mu\nu}$, for simplicity. The following formulas (discussed in Appendix C) are general for decomposing second-order terms:

$$\begin{aligned} \rho_{(n-2,2)} \downarrow_{S_p \times S_q} \equiv & (\rho_{(p)} \otimes \rho_{(q)}) \oplus (\rho_{(p)} \otimes \rho_{(q-1,1)}) \\ & \oplus (\rho_{(p)} \otimes \rho_{(q-2,2)}) \oplus (\rho_{(p-1,1)} \otimes \rho_{(q)}) \\ & \oplus (\rho_{(p-1,1)} \otimes \rho_{(q-1,1)}) \oplus (\rho_{(p-2,2)} \otimes \rho_{(q)}), \end{aligned}$$

$$\begin{aligned} \rho_{(n-2,1,1)} \downarrow_{S_p \times S_q} \equiv & (\rho_{(p)} \otimes \rho_{(q-1,1)}) \oplus (\rho_{(p)} \otimes \rho_{(q-2,1,1)}) \\ & \oplus (\rho_{(p-1,1)} \otimes \rho_{(q)}) \oplus (\rho_{(p-1,1)} \otimes \rho_{(q-1,1)}) \\ & \oplus (\rho_{(p-2,1,1)} \otimes \rho_{(q)}). \end{aligned}$$

Computing the LR coefficients has been shown, in general, to be a #P-complete problem [104]. For low-order Fourier terms (corresponding to partitions with only a few rows), however, the *Littlewood–Richardson rule* [114] computes the LR coefficients in reasonable time (see Appendix C). We plan to make tables and code available online. We refer interested readers to [114] for a more detailed discussion of the Littlewood–Richardson rule. We note that while the LR coefficients have been studied in various mathematical contexts [73, 114, 130, 43, 104], our work provides, to the best of our knowledge, the first connection to probabilistic independence.

As for the coupling matrices, $L^{\lambda}_{\mu\nu}$, there are no known analytical formulas, and in practice, acquiring these matrices requires considerable precomputation. As fundamental constants related to the irreducibles of the symmetric group, however, they need only be computed once and for all and can be stored in a table for all future reference. See Appendix D for a detailed discussion of techniques for computing the coupling coefficient matrices $L^{\lambda}_{\mu\nu}$.

In this section, we have characterized probabilistic independence in the Fourier domain in a structural sense. We showed that Fourier terms of the joint (written with respect to the appropriate basis) decompose not only into block-diagonal components, like the first-order marginals, but that each block decomposes as Kronecker products. More intriguing is the possibility of defining relaxations of independence based on Equation 12.4. A first-order independent distribution, as we have discussed, is one that cannot be distinguished from a distribution which fully factors independently by examining only the first-order marginals. Likewise, one can imagine defining higher-order notions. A second-order independent distribution, for example, is one that cannot be distinguished from one that fully factors by examining only the first and second order terms. While such higher-order 'relaxations' of independence are not exactly full independence, they can be thought of as approximations, and may turn out to be sufficient in certain cases for applications of independence, just as the pairwise independence assumption is often sufficient in certain randomized algorithm theorems [100].

12.3 ALGORITHMS

We now discuss two sets of algorithms for merging independent factors to form a joint distribution (JOIN), and for extracting independent factors from a joint (SPLIT). We first present (in Section 12.3.1) algorithms that are directly based on the Littlewood–Richardson factorization (Proposition 67) from the previous section. In Section 12.3.2, we present an alternative algorithm inspired by ideas from the noncommutative Fast Fourier transform [19, 20, 75] (Chapter 6) which does not require Littlewood–Richardson coefficients/matrices to be precomputed.

12.3.1 *Littlewood-Richardson based method*

12.3.1.1 JOIN.

The simpler operation of the two is the JOIN algorithm. Given the Fourier transforms of independent distributions, $\widehat{f}$ and $\widehat{g}$, the Fourier transform of the joint, $\widehat{h}$, can be constructed by simply forming the direct sum of appropriate tensor product terms $\widehat{f}_\mu \otimes \widehat{g}_\nu$, and conjugating by the precomputed coupling matrix $L^\lambda_{\mu\nu}$ (Equation 12.4). See Algorithm 12.1 for pseudocode.

One might worry that we would require maintaining high frequency terms of the independent factors in order to construct low frequency terms of the joint. We show, using the Littlewood–Richardson rule, that this is not the case when we maintain s^{th}-order marginals.

Theorem 68. *For any integer s such that $0 \leqslant s < n$, define the following partitions:*

$$\lambda^{\text{MIN}} = (n-s, \underbrace{1, \ldots, 1}_{s \text{ times}}),$$

$$\mu^{\text{MIN}} = (p-k, \underbrace{1, \ldots, 1}_{k \text{ times}}), \qquad \nu^{\text{MIN}} = (q-\ell, \underbrace{1, \ldots, 1}_{\ell \text{ times}}), \tag{12.9}$$

where $k = \min(s, p-1)$ and $\ell = \min(s, q-1)$. Given marginals of type μ^{MIN} for f and of type ν^{MIN} for g, JOIN *returns Fourier coefficients which can reconstruct marginals of type λ^{MIN} exactly.*

Theorem 68 formalizes the intuitive idea that it is possible, using the JOIN algorithm, to exactly construct s^{th}-order marginals of the joint distribution using only the s^{th}-order marginals of each independent factor. The proof of Theorem 68 is given in the Appendix. A more general principle holds for other partitions which do not take the form $\lambda^{\text{MIN}} = (n-s, 1, \ldots, 1)$, but we will focus on the simpler and more intuitive case of s^{th}-order marginals.

12.3.1.2 SPLIT.

Given the Fourier transform of the joint, $\widehat{h}$, we wish to formulate an algorithm which computes the Fourier coefficients of the factors $\widehat{f}$ and $\widehat{g}$, assuming that the sets $A_{in} = \{1, \ldots, p\}$ and $B_{in} = \{p+q, \ldots, n\}$ are independent under $h(\sigma)$. Conceptually, we would like to simply "invert" the Join algorithm by reading off the $\widehat{f}_\mu$ and $\widehat{g}_\nu$ matrices from $\bigoplus_{\mu,\nu} \bigoplus_{\ell=1}^{c^\lambda_{\mu\nu}} \widehat{f}_\mu \otimes \widehat{g}_\nu = L^\lambda_{\mu\nu} \cdot \widehat{h}_\lambda \cdot {L^\lambda_{\mu\nu}}^T$. The difficulty is that any specific Kronecker product $\widehat{f}_\mu \otimes \widehat{g}_\nu$, only determines $\widehat{f}_\mu$ and $\widehat{g}_\nu$ up to a scaling factor, and furthermore, in the approximate case when A_{in} and B_{in} are only "nearly" independent, the appropriate blocks of the matrix $L^\lambda_{\mu\nu} \cdot \widehat{h}_\lambda \cdot {L^\lambda_{\mu\nu}}^T$ do not take the form $M_1 \otimes M_2$.

As it turns out, however, we are in fact always able to determine the matrix coefficients in $\widehat{f}$ and $\widehat{g}$ using *only* blocks of the form $\widehat{f}_\mu \otimes 1$, or $1 \otimes \widehat{g}_\nu$, allowing us to literally read off the matrices for $\widehat{f}_\mu$ and $\widehat{g}_\nu$.

Theorem 69. *Define* λ^{MIN}, μ^{MIN}, *and* ν^{MIN} *as in Theorem 68. For any* $\mu \trianglerighteq \mu^{\text{MIN}}$, *there exists a block of* $L^{\lambda}_{\mu\nu} \cdot \widehat{h}_{\lambda} \cdot {L^{\lambda}_{\mu\nu}}^{T}$ *for some* $\lambda \trianglerighteq \lambda^{\text{MIN}}$ *which is identically equal to* $\widehat{f}_{\mu}$.

Likewise, for any $\nu \trianglerighteq \nu^{\text{MIN}}$, there exists a block of $L^{\lambda}_{\mu\nu} \cdot \widehat{h}_{\lambda} \cdot {L^{\lambda}_{\mu\nu}}^{T}$ for some $\lambda \trianglerighteq \lambda^{\text{MIN}}$ which is identically equal to $\widehat{g}_{\nu}$. See Algorithm 12.2 for pseudocode for the SPLIT algorithm and the appendix for a constructive procedure that finds the appropriate partitions λ in Theorem 69. As a corollary, we obtain a converse to Theorem 68 which says that given the s^{th} order marginals of the joint, we will be able to recover the s^{th} order marginals of the factors.

Corollary 70. *Given marginals of type* λ^{MIN}, SPLIT *returns Fourier coefficients of the factors* f *and* g *which can be used to exactly reconstruct marginals of type* μ^{MIN} *and* ν^{MIN}, *respectively.*

A similar (refined) guarantee holds with respect to the lexicographical ordering for partitions.

Theorem 71. *Let* k *be a positive integer. The first* k *terms of* $\widehat{f}$ *and* $\widehat{g}$ *with respect to the lexicographical ordering are sufficient for recovering the first* k *terms of* $\widehat{h}$ *exactly. Conversely, the first* k *terms of* $\widehat{h}$ *are sufficient for recovering the first* k *terms of* $\widehat{f}$ *and* $\widehat{g}$.

Finally, we discuss what happens when SPLIT is called on a distribution h which *does not* factor into independent distributions f and g. We show (proof given in Appendix C) that when the first order independence condition is satisfied (that is, when $h(\sigma) = 0$ for all $\sigma \notin S_p \times S_q$, but does not necessarily factor into two independent terms), the SPLIT algorithm still returns Fourier coefficients of the relevant marginal distributions.

Theorem 72. *Given the Fourier coefficients,* $\widehat{h}$ *of a distribution* h *satisfying the first order independence condition (but which does not necessarily factor as* $f \cdot g$*), the Split algorithm returns the (exact) Fourier coefficients of the marginal distributions over* $\{1, \ldots, p\}$ *and* $\{p+1, \ldots, n\}$.

When the first order condition does *not* apply, the result of SPLIT does not correspond to a normalized distribution, and so it is necessary in practice to normalize the resulting factors to sum to one, which can be performed in the Fourier domain by dividing every Fourier matrix by $\hat{f}_{(n)}$ (respectively, $\hat{g}_{(n)}$) (Equation 8.1). Unlike the conditioning operations presented in Chapter 8, the normalized outputs of the Split algorithm are guaranteed to be a legal distribution in the sense that $f(\sigma_p) \geqslant 0$ and $f(\sigma_q) \geqslant 0$ for all σ_p and σ_q provided that h is a legal distribution. Hence, there is no need to project the resulting Fourier coefficient matrices into the marginal polytope, as has been considered in Chapter 9.

12.3.2 *FFT based method*

In this section, we present an alternative set of JOIN/SPLIT algorithms based on the fast Fourier transform techniques that we presented in Chapter 6.

Algorithm 12.1: Pseudocode for the *Join* algorithm. Input: Fourier coefficient matrices of factors, $\{\widehat{f}_\rho\}_{\rho\in\Lambda_p}$, and $\{\widehat{g}_\rho\}_{\rho\in\Lambda_q}$. Output: Fourier coefficient matrices of joint distribution $\{\widehat{h}_\rho\}_{\rho\in\Lambda_n}$.

JOIN($\{\widehat{f}_\rho\}_{\rho\in\Lambda_p}, \{\widehat{g}_\rho\}_{\rho\in\Lambda_q}$):

foreach λ *such that* $\lambda \trianglerighteq \lambda^{\mathrm{MIN}}$ **do**
  $\widehat{h}_{\rho_\lambda} \leftarrow {L^\lambda_{\mu\nu}}^{\mathsf{T}} \cdot \left(\bigoplus_{\mu,\nu} \bigoplus_{\ell=1}^{c^\lambda_{\mu\nu}} \widehat{f}_{\rho_\mu} \otimes \widehat{g}_{\rho_\nu} \right) \cdot L^\lambda_{\mu\nu}$;
end
return $\left(\{\widehat{h}_\rho\}_{\rho\in\Lambda_n} \right)$;

Algorithm 12.2: Pseudocode for the *Split* algorithm. Input: Fourier coefficient matrices of joint distribution $\{\widehat{h}_\rho\}_{\rho\in\Lambda_n}$. Output: Fourier coefficient matrices of factors, $\{\widehat{f}_\rho\}_{\rho\in\Lambda_p}$, and $\{\widehat{g}_\rho\}_{\rho\in\Lambda_q}$.

SPLIT:($\{\widehat{h}_\rho\}_{\rho\in\Lambda_n}$)

foreach *partition* μ *of* p *such that* $\mu \trianglerighteq \mu^{\mathrm{MIN}}$ **do**
  Find $\lambda \trianglerighteq \lambda^{\mathrm{MIN}}$ such that $c^\lambda_{\mu,(q)} > 0$;
  $\widehat{f}_{\rho_\mu} \leftarrow (\mu,(q))$-block of the matrix $L^\lambda_{\mu\nu} \cdot \widehat{h}_{\rho_\lambda} \cdot {L^\lambda_{\mu\nu}}^{\mathsf{T}}$;
end
foreach *partition* ν *of* q *such that* $\nu \trianglerighteq \nu^{\mathrm{MIN}}$ **do**
  Find $\lambda \trianglerighteq \lambda^{\mathrm{MIN}}$ such that $c^\lambda_{(p),\nu} > 0$;
  $\widehat{g}_{\rho_\nu} \leftarrow ((p),\nu)$-block of the matrix $L^\lambda_{\mu\nu} \cdot \widehat{h}_{\rho_\lambda} \cdot {L^\lambda_{\mu\nu}}^{\mathsf{T}}$;
end
return $\left(\{\widehat{f}_\rho\}_{\rho\in\Lambda_p}, \{\widehat{g}_\rho\}_{\rho\in\Lambda_q} \right)$;

Recall that the insight behind the Clausen FFT is that one can decompose a function on S_n into a sum of functions on S_{n-1}. The FFT then recursively computes the Fourier transform of these smaller functions, and pieces them together afterwards to form the Fourier transform of the original function.

In the following, we use iterated EMBED/RESTRICT operations to work with a function over S_p as if it were a function over S_n, and vice versa. Both the RESTRICT and EMBED operations can be performed in the Fourier domain as we have discussed in Chapter 6.

Just as we have defined an EMBED operation for functions over permutations, we can similarly define an EMBED operation for permutations (we will not need a RESTRICT operation). Given a permutation $\sigma_p \in S_p$, EMBED$[\sigma_p]$ returns the following permutation in S_n:

$$\textsc{Embed}[\sigma_p](i) = \begin{cases} \sigma_p(i) & \text{if } i \leqslant p \\ i & \text{if } i > p \end{cases} .$$

Note that if f is the distribution of the random variable σ_p, then EMBED$[f]$ is the distribution of EMBED$[\sigma_p]$.

12.3.2.1 *FFTJoin*

The intuition behind our second JOIN algorithm, FFTJOIN, comes from the following interpretation of convolution. If $\sigma_1, \sigma_2 \in S_n$ are random permutations drawn *independently* from distributions f and g on S_n, then

the distribution of the composition $\sigma_1\sigma_2$ is given by the convolution of the two distributions, $P_1 * P_2$.

To make use of convolution in the JOIN setting, we will write the permutation $[\sigma_p\ \sigma_q]$ as the following composition of permutations:

$$[\sigma_p\ \sigma_q] = (\textsc{Embed}^q[\sigma_p])\,(\textsc{Shift}[\textsc{Embed}^p[\sigma_q], \omega_{p,q}, \omega_{p,q}])\,,$$

where $\omega_{p,q} = [p+1, p+2, \ldots, n, 1, 2, \ldots, p]$ (written using one-line notation).

Example 73. *Let* $p = q = 3$ *(*$n = 6$*), and suppose that* $\sigma_p = [1,2,3]$ *and* $\sigma_q = [3,2,1]$*. Then* $\omega_{3,3} = \omega_{3,3}^{-1} = [4,5,6,1,2,3]$*,* $\textsc{Embed}^3[\sigma_p]^3 = [1,2,3,4,5,6]$ *and* $\textsc{Embed}^3[\sigma_q] = [3,2,1,4,5,6]$*. It can be checked that:*

$$\begin{aligned}
[\sigma_p, \sigma_q] &= \left(\textsc{Embed}^3[\sigma_p]\right)\left(\textsc{Shift}[\textsc{Embed}^3[\sigma_q], \omega_{3,3}, \omega_{3,3}]\right), \\
&= \left(\textsc{Embed}^3[\sigma_p]\right)\left(\omega_{3,3}\,\textsc{Embed}^3[\sigma_q]\,\omega_{3,3}^{-1}\right), \\
&= [1,2,3,4,5,6]\,([4,5,6,1,2,3][3,2,1,4,5,6][4,5,6,1,2,3])\,, \\
&= [1,2,3,4,5,6][1,2,3,6,5,4], \\
&= [1,2,3,6,5,4].
\end{aligned}$$

Since σ_p and σ_q are independent, the permutations are $\textsc{Embed}[\sigma_p]$ and $\textsc{Shift}[\textsc{Embed}[\sigma_q], \omega_{p,q}, \omega_{p,q}]$ are independent and we can convolve their respective distributions to obtain the joint distribution. To achieve the same result in the Fourier domain, notice that both the Shift and Embed operations have Fourier domain counterparts that allow us to compute the distributions of $\textsc{Embed}[\sigma_p]$ and $\textsc{Shift}[\textsc{Embed}[\sigma_q], \omega_{p,q}]$ from the distributions of σ_p and σ_q (see Chapter 6). Finally, using the convolution theorem (Proposition 51), we can pointwise multiply the Fourier coefficient matrices for each distribution to find the Fourier coefficients of the joint distribution. See Algorithm 12.3 for pseudocode.

12.3.2.2 *FFTSplit*

Our FFT based algorithm for SPLIT is based on a somewhat different interpretation of convolution. We will view convolution here as an averaging (or marginalization) operator. If $h(\sigma) = f(\sigma(1), \ldots, \sigma(p)) \cdot g(\sigma(p+1), \ldots, \sigma(n))$, then f is the marginal distribution for $(\sigma(1), \ldots, \sigma(p))$ and g is the marginal distribution for $(\sigma(p+1), \ldots, \sigma(n))$. Loosely speaking, to find the marginal distribution for $(\sigma(p+1), \ldots, \sigma(n))$, for example, we will first sum over all settings of $(\sigma(1), \ldots, \sigma(p)\})$ by convolving the joint distribution by an indicator function for the subgroup S_p, then iteratively use the RESTRICT operator to obtain a function on S_q.

For simplicity, we will only discuss computing the marginal distribution of $(\sigma(p+1), \ldots, \sigma(n))$. To adapt the following algorithm for computing marginals of $(\sigma(1), \ldots, \sigma(p))$, one can use the Shift theorem (Equation 8.3). In the following, we discuss each step in detail. See Algorithm 12.4 for pseudocode.

Algorithm 12.3: Pseudocode for the FFT-based *Join* algorithm. Input: Fourier coefficient matrices of the factors, $\{\widehat{f}_\rho\}_{\rho\in\Lambda_p}$ and $\{\widehat{g}_\rho\}_{\rho\in\Lambda_q}$. Output: Fourier coefficient matrices of the joint distribution, $\{\widehat{h}_\rho\}_{\rho\in\Lambda_n}$.

FFTJOIN($\{\widehat{f}_\rho\}_{\rho\in\Lambda_p}, \{\widehat{g}_\rho\}_{\rho\in\Lambda_q}$):

$\widehat{f}' \leftarrow$ EMBED$^q[\hat{f}]$;
$\widehat{g}' \leftarrow$ SHIFT[EMBED$^p[\hat{g}], \omega_{p,q}, \omega_{p,q}]$;
foreach λ *such that* $\lambda \trianglerighteq \lambda^{MIN}$ **do**
  $\hat{h}_{\rho_\lambda} \leftarrow \widehat{f}'_{\rho_\lambda} \cdot \widehat{g}'_{\rho_\lambda}$;
end
return $\left(\{\widehat{h}_\rho\}_{\rho\in\Lambda_n}\right)$;

1. *(Integrating over S_p)* For the SPLIT operation, we identify the group S_p as a subgroup of elements in S_n which permute the first p elements but fix the remaining q elements. The indicator function of S_p, $\delta_{S_p} : S_n \to \mathbb{R}$, is therefore defined as:

$$\delta_{S_p}(\sigma) = \begin{cases} 1 & \text{if } \sigma(i) = i, \text{ for all } p+1 \leqslant i \leqslant n \\ 0 & \text{otherwise} \end{cases}. \tag{12.10}$$

   Convolving the joint distribution h against δ_{S_p} effectively integrates over the variables $\{1, \ldots, p\}$. Thus,

$$h'(\sigma) = [h * \delta_{S_p}](\sigma) = \sum_{\tau\in S_n} h(\sigma\tau)\delta_{S_p}(\tau^{-1}) = \sum_{\tau\in S_p} h(\sigma\tau).$$

   Using the convolution theorem again (Proposition 51), we compute the Fourier coefficients of h' via a matrix multiplication at each frequency level: $\widehat{h'}_\lambda = \widehat{h}_\lambda \cdot \widehat{[\delta_{S_p}]}_\lambda$. The Fourier transform of indicator functions of the form δ_{S_p} can be computed efficiently (Chapter 6) and $\widehat{[\delta_{S_p}]}_{\rho_\lambda}$ is known to be a diagonal matrix for each partition λ.

2. *(Restricting to S_q)* We have shown that the marginal probability of σ_q is given by $h'([\sigma_p, \sigma_q])$ and that the Fourier transform of h' can be computed. The function h', however, is still defined over S_n and we would like to obtain a function of the form $g : S_q \to \mathbb{R}$. To do this, we use the RESTRICT operator. Since RESTRICT drops arguments from the right, we use the Shift operation to first swap the arguments of h': $h^{flip}([\sigma_q, \sigma_p]) =$ SHIFT$[[, h]', \omega_{p,q}, \omega_{p,q}]$, then iteratively call the RESTRICT operation on h^{flip} until we obtain a function on S_q.

12.4 SCALABLE IDENTITY MANAGEMENT BY EXPLOITING INDEPENDENCE

As an application, we use our Fourier theoretic algorithms in the identity management setting. In previous chapters, we have reasoned *jointly* over assignments of all n tracks to all n identities. In realistic settings however, we believe that it is often sufficient to only reason over small cliques of tracks at a time. Thus, instead of maintaining Fourier coefficients over

Algorithm 12.4: Pseudocode for the FFT-based *Split* algorithm. Input: Fourier coefficient matrices of the joint distribution, $\{\widehat{h}_\rho\}_{\rho \in \Lambda_n}$. Output: Fourier coefficient matrices of the factors, $\{\widehat{f}_\rho\}_{\rho \in \Lambda_p}$ and $\{\widehat{g}_\rho\}_{\rho \in \Lambda_q}$.

FFTSPLIT($\{\widehat{h}_\rho\}_{\rho \in \Lambda_n}$):

```
//Compute the marginal over (p+1, ..., n)
foreach λ such that λ ⊵ λ^MIN do
    ĥ'_{ρ_λ} ← ĥ_{ρ_λ} · [δ̂_{S_p}]_{ρ_λ} ;
end
ĝ ← RESTRICT^p[SHIFT[ĥ', ω_{p,q}, ω_{p,q}]] ;
//Compute the marginal over (1, ..., p)
ĥ* ← SHIFT[ĥ, ω_{p,q}^{-1}, ω_{p,q}^{-1}] ;
foreach λ such that λ ⊵ λ^MIN do
    ĥ'_{ρ_λ} ← [ĥ*]_{ρ_λ} · [δ̂_{S_q}]_{ρ_λ} ;
end
f̂ ← RESTRICT^q[SHIFT[ĥ', ω_{p,q}, ω_{p,q}]] ;
return ({f̂_ρ}_{ρ∈Λ_p}, {ĝ_ρ}_{ρ∈Λ_q}) ;
```

Algorithm 12.5: Simplified pseudocode for adaptive identity management.

ADAPTIVEIDM:

```
//Initialize each track to be in its own clique
foreach clique c ∈ {1, ..., n} do
    clique(c).ids ← c ;
    clique(c).tracks ← c ;
end
//Initialize Fourier coefficients for each clique to the uniform distribution
foreach clique c ∈ {1, ..., n} do
    f̂^c_(1) ← 1 ;
end
foreach timeslice t do
    if tracks i and j mix then
        if tracks i and j fall in different cliques then
            c_i ← clique containing track i;
            c_j ← clique containing track j;
            Join cliques c_i and c_j and corresponding Fourier transforms f̂^{c_i},
            f̂^{c_j} ;
        end
        c ← clique containing tracks i and j;
        Apply Fourier domain prediction/rollup algorithm to clique c ;
    end
    if observe identity k at track i then
        c ← clique containing track i;
        Apply Fourier domain conditioning algorithm to clique c ;
    end
    foreach clique c do
        Run balanced biclustering algorithm on clique c ;
        if splitting criterion is satisfied then
            Split clique c into smaller cliques ;
        end
    end
end
```

all of S_n, we search for independent cliques and *adaptively* split the distribution into factors over smaller cliques whenever possible. In order to

find independent subsets of tracked identities, we first present a simple clustering-based method for independence detection.

12.4.1 *Detecting independent subsets of objects*

In this section, we discard the assumption that $A_{in} = A_{out} = \{1, \ldots, p\}$ and deal with the problem of explicitly finding sets A_{in} and A_{out} such that $h(\sigma(A_{in}) \subset A_{out}) = 1$ and $h(\sigma(B_{in}) \subset B_{out}) = 1$ as defined by the first-order condition. We begin with the simple observation that if we *knew* the sets A_{in} and A_{out}, then the first-order matrix of marginals would be rendered block diagonal under an appropriate reordering of the rows and columns (Figure 24b). Since A_{in} and A_{out} are unknown, our task is to find permutations of the rows and columns of the first-order matrix of marginals (Figure 24d) to obtain a block diagonal matrix. Viewing the matrix of first-order marginals as a set of edge weights on a bipartite graph between tracks and identities, we approach the detection step as a *biclustering* problem (in which one simultaneously clusters the tracks and identities) with an extra *balance* constraint forcing $|A_{in}| = |A_{out}|$. In our experiments, we use the SVD-based technique presented in [138], which finds bipartite graph partitions optimizing the normalized cut measure. We have modified their algorithm slightly so that the output satisfies the balance constraint.

Assuming now that we have obtained the sets A_{in} and A_{out} via the above clustering step, we can call the SPLIT algorithm by first renaming the tracks and identities so that $A_{in} = A_{out} = \{1, \ldots, p\}$. Suppose that, to achieve this reordering, we must permute the A_{in} (people) using a permutation π_1 and the A_{out} (tracks) using π_2. The Shift Theorem (Equation 8.3) can be applied to reorder the Fourier coefficients according to these new labels, and we can then apply Algorithm 12.2 unchanged.

We have focused on detecting independence in the first-order sense. As discussed in Section 12.1.1 however, this first-order notion is a necessary, but insufficient condition for higher order independence. As an example of when higher order terms are useful, consider the problem of tracking players in a football game from visual footage, where players on opposing teams are easily distinguishable from each other. In such a scenario, one would not accidentally confuse players on one team for players on the other team, and thus the first-order condition is satisfied. However, it might not be appropriate to track two teams independently of one another. Knowing, for example, that player A guards player B on the opposing team is valuable information that would be discarded if the two teams were to be tracked separately.

To deal with the fact that a candidate split might not be independent, note first that by Theorem 72, if first-order independence *does* indeed hold, then the Split algorithm returns the appropriate Fourier coefficients for each marginal distribution. Additionally, if one is concerned with independence at higher-order terms, note that our splitting algorithm is able to factor the distribution at every order. Once this factoring is performed, we can measure its effect on higher orders, e.g., using Plancherel's Theorem

(Proposition 61) to measure the distance between the original coefficients and the factored result, and decide whether or not to retain the partition.

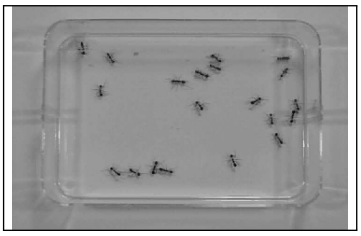
12.4.2 *Adaptive identity management*

In our adaptive approach (see Algorithm 12.5), we maintain a collection of disjoint cliques over the tracks and identities. After conditioning on any observation, we attempt to split. We also force splits whenever cliques grow to be too large to handle. Upon splitting, we allow the representational size to grow to higher orders — thus for very large n, we might only maintain first-order coefficients, but for smaller sized cliques, we might choose to represent higher-order coefficients. Finally, sometimes two tracks can become so close together that it is hard to distinguish which individual is on which track, in a mixing event. When these mixing events occur between tracks belonging to distinct cliques, it is necessary to jointly reason over objects from both cliques at once. We therefore merge the cliques using our JOIN algorithm and perform a mixing on the newly formed joint distribution as in Chapter 8.

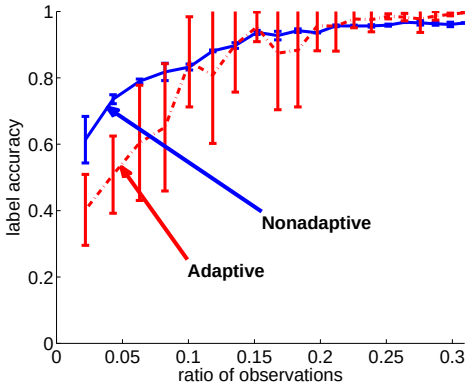
12.5 EXPERIMENTS

We evaluted our adaptive identity management algorithm on a biotracking dataset from [72]. In their data, there are 20 ants (Fig. 26a) moving in an enclosed area. The data is interesting for our purposes since n is relatively large compared to many other multi-object tracking datasets and there are interesting movement patterns with plenty of mixing events (which we log whenever ants walk within some distance of each other). At each timestep, we allow each ant to 'reveal' its identity with some probability (in our experiments, ranging from $p_{obs} = .005$ to $p_{obs} = .05$ per timeframe), and our task is to jointly label all tracks with identities for all timeframes. We measure accuracy using the fraction of correctly labeled tracks over the entire sequence (note that the accuracy of random guessing is $1/n = 5\%$ in expectation). As a splitting criterion, we decide to split if, after clustering, the sum over all off-block elements falls below a certain threshold ϵ (in all experiments, we fixed $\epsilon = 1/(2n)$).

In Figures 26b and 27a, we compare the performance of an adaptive approach against the nonadaptive algorithm from [57] as we vary the ratio of observations to mixing events. Figure 26b shows that the two algorithms perform similarly in accuracy, with the nonadaptive approach faring slightly better with fewer observations (due to more diffuse distributions) and slightly worse with more observations (due to the fact that the adaptive approach can represent higher-order Fourier terms). The real advantage of our adaptive approach is shown in Figure 27a, which plots a running time comparison. Since the conditioning step is the complexity bottleneck of performing inference in the Fourier domain, the running time scales according to the proportion of observations. However, since the adaptive algorithm typically conditions smaller cliques on average (especially with



(a) Small image from biotracking data — there are twenty ants moving in a tray in this dataset



(b) Accuracy comparison — we compare our adaptive algorithm against the performance of the nonadaptive algorithm from [59].

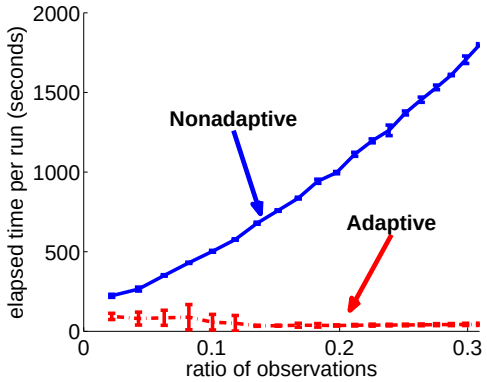
Figure 26: Experimental results on biotracking data

more observations), we see that it is far more scalable. In Figure 27b, we plot the average number of cliques and sizes of cliques which were formed in the same experiment. As expected, we see that the cliques get smaller and more numerous as the number of observations grows.

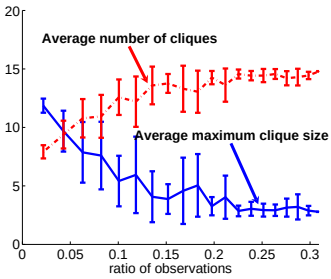
Finally, we simulated larger tracking problems by taking m different segments of the ant data and tracking $m \cdot n$ ants at the same time allowing for ants to 'teleport' to other segments with some probability. Figure 27c shows a comparison of average running time for these larger problems. Note that at such sizes, we can no longer feasibly run the original nonadaptive algorithms from [57, 80].

12.6 CONCLUSION

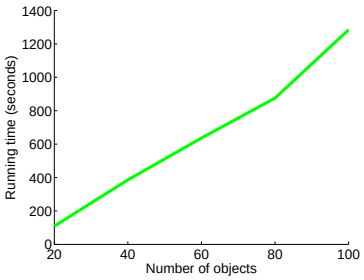
A pervasive technique in machine learning for making large problems tractable is to exploit probabilistic independence structures for decomposing large problems into much smaller ones. It is the structure of (conditional) independence, for example, which has made Bayes net and Markov random field representations so powerful and thus popular. In this chapter, we have presented the first study of independence for distributions over permutations from a *Fourier-theoretic* perspective. Our theoretical results show that



(a) Running times comparison between the adaptive algorithm and the nonadaptive algorithm. When there are many observations, the distribution factorizes readily, allowing the adaptive approach to achieve significant speedups.



(b) Clique Sizes — here we visualize the average number of existing cliques and average clique size per timestep as we increase the number of observations.



(c) Running times vs. n — note that we are able to scale the adaptive algorithm to handle up to a hundred objects while typical nonadaptive Fourier approaches do not scale very well past twenty.

Figure 27: Experimental results on biotracking data

distributions which factor independently must satisfy a certain structural decomposition in the Fourier domain. From these structural insights, we have formulated algorithms for joining and splitting distributions using Fourier coefficients. Our algorithms can be integrated seamlessly into the existing Fourier theoretic inference framework of Chapter 8 and are useful additions to the growing 'toolbox' of Fourier theoretic operations that can be performed on distributions over permutations. Combined with the bandlimited approximate inference algorithms from these earlier papers, we believe that our algorithms will contribute to making these Fourier based methods highly scalable and practical.

Finally, we view our contributions as a first step towards understanding and exploiting more intermediate notions of probabilistic independence, which lie somewhere between full independence and fully connected models, such as conditional or context-specific independence. In the next chapters, we explore a novel notion of independence known as *riffled*

independence which can be seen as a generalization of our work, which is particularly relevant for ranked data. Conditional and context-specific independence assumptions have proven themselves to be indispensible in the fields of machine learning and AI, and we believe that generalizations of independence (such as the concept of riffled independence) will be indispensible for performing learning and inference with permutation data.

13

BEYOND FULL INDEPENDENCE: RIFFLED INDEPENDENCE FOR RANKINGS

IF full independence is often an unrealistic assumption for typical machine learning problems, it is even more egregious an assumption for ranking problems due to the first-order condition discussed in the previous chapter. In this chapter, we propose a generalization of probabilistic independence called *riffled independence*, which we show to be a much more realistic assumption for ranked data.

Our generalization is similar in spirit to graphical models which use another generalization (conditional independence) to capture a richer class of distributions than that captured by simple naive Bayes models while retaining many of the same computational advantages.

The contributions of this chapter can be summarized as follows:

- We introduce an intuitive generalization of independence on permutations, which we call *riffled independence*, and show it to be a more appropriate notion of independence for ranked data.
- We introduce a novel family of distributions, called *biased riffle shuffles*, that are useful for riffled independence and propose an algorithm for computing its Fourier transform.
- We provide an efficient recursive procedure for computing the Fourier transform of the riffle shuffle distribution. Using this result, we devise a method for teasing apart the riffle-independent components of a distribution in the Fourier domain.
- We provide algorithms that can be used in the Fourier-theoretic framework of [80, 60, 59] for joining riffle independent factors (RiffleJoin), and for teasing apart the riffle independent factors from a joint (RiffleSplit), and provide theoretical and empirical evidence that our algorithms perform well.
- Finally, we show evidence that riffle independent relationships can approximately hold in real datasets, justifying the use of our methods.

13.1 SHORTCOMINGS OF FULL INDEPENDENCE

To scale to large problems, we have demonstrated in the previous chapter that, by exploiting *probabilistic independence*, one can dramatically improve the scalability of Fourier-based methods, e.g., for tracking problems, since confusion in data association only occurs over small independent subgroups of objects in many problems. Despite its utility for many tracking problems, however, we argue that the first-order condition implied by independence

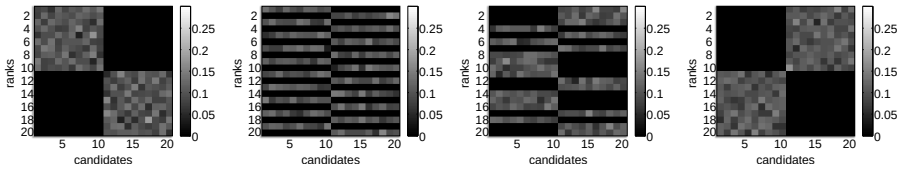


Figure 28: Example first-order matrices with $A = \{1,2,3\}$, $B = \{4,5,6\}$ fully independent, where black means $h(\sigma : \sigma(j) = i) = 0$. In each case, there is some 3-subset A' which A is constrained to map to with probability one. With respect to some rearranging of the rows, independence imposes a block-diagonal structure on first-order matrices.

imposes a rather harsh constraint on distributions, rendering independence highly unrealistic in ranking applications. Recall that if $\sigma(A)$ and $\sigma(B)$ are independent, then A and B are not allowed to map to the same ranks. That is, for some fixed p-subset $A' \subset \{1,\ldots,n\}$, $\sigma(A)$ is a permutation of elements in A' and $\sigma(B)$ is a permutation of its complement, B', with probability 1. See Figure 28.

Example 74. *Continuing with our vegetable/fruit example with $n = 6$, if the vegetable and fruit rankings,*

$$\sigma_A = [\sigma(Corn), \sigma(Peas)], \text{ and}$$

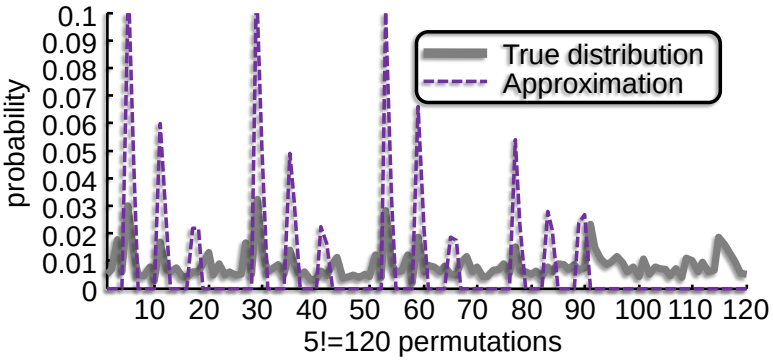
$$\sigma_B = [\sigma(Lemons), \sigma(Oranges), \sigma(Figs), \sigma(Grapes)],$$

are known to be independent. Then for $A' = \{1,2\}$, the vegetables occupy the first and second ranks with probability one, and the fruits occupy ranks $B' = \{3,4,5,6\}$ with probability one, reflecting that vegetables are always preferred over fruits according to this distribution.

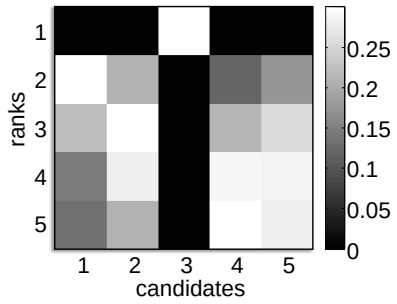
In sports tracking, permutations represent the mapping between the identities of players with positions on the field, and in such settings, the first-order condition might say, quite reasonably, that there is potential identity confusion within tracks for the red team and within tracks for the blue team but no confusion between the two teams. In our ranking example however, the first-order condition forces the probability of any vegetable being in third place to be zero, even though both vegetables will, in general, have nonzero marginal probability of being in second place, which seems quite unrealistic.

Example 75 (APA election data (continued))**.** *Consider approximating the APA vote distribution by a factorized distribution (as in Definition 63). In Figure 29, we plot (in solid purple) the factored distribution which is closest to the true distribution with respect to total variation distance. In our approximation, candidate 3 is constrained to be independent of the remaining four candidates and maps to rank 1 with probability 1.*

While capturing the fact that the "winner" of the election should be candidate 3, the fully factored distribution can be seen to be a poor approximation, assigning



(a)



(b)

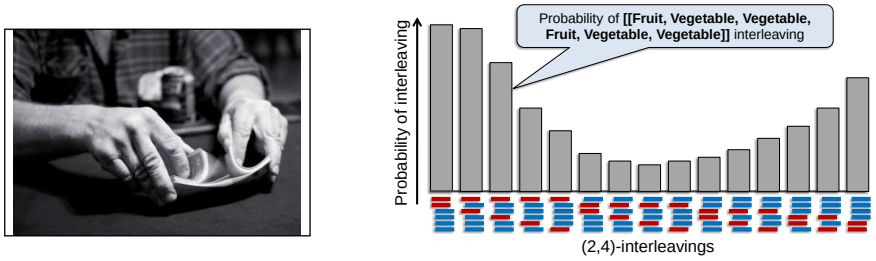
Figure 29: Approximating the APA vote distribution by a factored distribution in which candidate 3 is independent of candidates $\{1,2,4,5\}$. (a) in thick gray, the true distribution, in dotted purple, the approximate distribution. Notice that the factored distribution assigns zero probability to most permutations. (b) matrix of first order marginals of the approximating distribution.

zero probability to most permutations even if all permutations received a positive number of votes. Since the support of the true distribution is not contained within the support of the approximation, the KL divergence, $D_{KL}(h_{true}; h_{approx})$ *is infinite.*

In the next section, we overcome the restrictive first-order condition with the more flexible notion of *riffled independence*.

13.2 RIFFLED INDEPENDENCE: DEFINITIONS AND EXAMPLES

The *riffle (or dovetail) shuffle* [11] is perhaps the most commonly used method of card shuffling, in which one cuts a deck of n cards into two piles, $A = \{1,\ldots,p\}$ and $B = \{p+1,\ldots,n\}$, with size p and $q = n-p$, respectively, and successively drops the cards, one by one, so that the two piles become interleaved (see Figure 30a) into a single deck again. Inspired by the riffle shuffle, we present a novel relaxation of the full independence assumption, which we call *riffled independence*. Rankings that are riffle independent are formed by independently selecting rankings for two disjoint subsets of objects, then interleaving the two rankings using a riffle shuffle



(a) (b)

Figure 30: (a) Photograph of the riffle shuffle executed on a standard deck of cards; (b) Pictorial example of a $(2,4)$-interleaving distribution, with red cards (offset to the left) denoting Vegetables, and blue cards (offset to the right) denoting Fruits.

to form a final ranking over all objects. Intuitively, riffled independence models complex relationships within each set A and B while allowing correlations between the sets to be modeled only through a constrained form of shuffling.

Example 76. *Consider generating a ranking of vegetables and fruits. We might first 'cut the deck' into two piles, a pile of vegetables (*A*) and a pile of fruits (*B*), and in a first stage, independently decide how to rank each pile. For example, within vegetables, we might decide that Peas are preferred to Corn:* $[\![\mathbf{P}, \mathbf{C}]\!] = [\![\mathtt{Peas}, \mathtt{Corn}]\!]$. *Similarly, within fruits, we might decide on the following ranking:* $[\![\mathbf{L}, \mathbf{F}, \mathbf{G}, \mathbf{O}]\!] = [\![\mathtt{Lemons}, \mathtt{Figs}, \mathtt{Grapes}, \mathtt{Oranges}]\!]$ *(Lemons preferred over Figs, Figs preferred over Grapes, Grapes preferred over Oranges).*

In the second stage of our model, the fruit and vegetable rankings are interleaved to form a full preference ranking over all six items. For example, if the interleaving is given by: $[\![\mathtt{Veg}, \mathtt{Fruit}, \mathtt{Fruit}, \mathtt{Fruit}, \mathtt{Veg}, \mathtt{Fruit}]\!]$, *then the resulting full ranking is:*

$$\sigma = [\![\mathtt{Peas}, \mathtt{Lemons}, \mathtt{Figs}, \mathtt{Grapes}, \mathtt{Corn}, \mathtt{Oranges}]\!].$$

13.2.1 *Convolution based definition of riffled independence*

There are two ways to define riffled independence, and, we will first provide a definition using convolutions, a view inspired by our card shuffling intuitions. Mathematically, shuffles are modeled as random walks on the symmetric group. The ranking σ' *after* a shuffle is generated from the ranking *prior* to that shuffle, σ, by drawing a permutation, τ from an *interleaving distribution* $m(\tau)$, and setting $\sigma' = \tau\sigma$ (the composition of the mapping τ with σ). Given the distribution h' over σ, we can find the distribution $h(\sigma')$ *after* the shuffle via the convolution formula (Equation 3.8):

$$h(\sigma') = \sum_{\sigma,\tau\,:\,\sigma'=\tau\sigma} m(\tau)h'(\sigma).$$

Besides the riffle shuffle, there are a number of different shuffling strategies — the pairwise shuffle, for example, simply selects two cards at random and swaps them. The question then, is *what are interleaving shuffling distributions* m *that correspond to riffle shuffles?* To answer this question, we use the distinguishing property of the riffle shuffle, that, after cutting the deck into two piles of size p and $q = n - p$, it must preserve the relative ranking relations within each pile. Thus, if the i^{th} card appears above the j^{th} card in one of the piles, then after shuffling, the i^{th} card *remains* above the j^{th} card. In our example, relative rank preservation says that if Peas is preferred over Corn prior to shuffling, they continue to be preferred over Corn after shuffling. Any allowable riffle shuffling distribution must therefore assign zero probability to permutations which do not preserve relative ranking relations. As it turns out, the set of permutations which *do* preserve these relations have a simple description.

Definition 77 (Interleaving distributions)**.** The (p, q)*-interleavings* are defined as the following set:

$$\Xi_{A,B} \equiv \{\tau \in S_n : \tau(1) < \tau(2) < \cdots < \tau(p), \text{ and } \tau(p+1) < \tau(p+2) < \cdots < \tau(n)\}.$$

A distribution $m_{A,B}$ on S_n is called an *interleaving distribution* if it assigns nonzero probability *only* to elements in $\Xi_{A,B}$.

The (p, q)-interleavings can be shown to preserve relative ranking relations within each of the subsets $A = \{1, \ldots, p\}$ and $B = \{p+1, \ldots, n\}$ upon multiplication:

Lemma 78. *Let* $i, j \in A = \{1, \ldots, p\}$ *(or* $i, j \in B = \{p+1, \ldots, n\}$*) and let* τ *be any* (p, q)*-interleaving in* $\Xi_{A,B}$*. Then* $i < j$ if and only if $\tau(i) < \tau(j)$ *(i.e., permutations in* $\Xi_{A,B}$ *preserve relative ranking relations).*

Example 79. *In our vegetable/fruits example, we have* $n = 6$, $p = 2$ *(two vegetables, four fruits). The set of* $(2, 4)$*-interleavings is:*

$$\Xi_{Veg,Fruit} = \left\{ \begin{array}{llll} [\mathbf{123456}], & [\mathbf{132456}], & [\mathbf{142356}], & [\mathbf{152346}], \\ [\mathbf{162345}], & [\mathbf{231456}], & [\mathbf{241356}], & [\mathbf{251346}], \\ [\mathbf{261345}], & [\mathbf{341256}], & [\mathbf{351245}], & [\mathbf{361245}], \\ [\mathbf{451236}], & [\mathbf{461235}], & [\mathbf{561234}] & \end{array} \right\},$$

or written in ordering notation,

$$\Xi_{Veg,Fruit} = \left\{ \begin{array}{llll} [\![\mathbf{VVFFFF}]\!], & [\![\mathbf{VFVFFF}]\!], & [\![\mathbf{VFFVFF}]\!], & [\![\mathbf{VFFFVF}]\!], \\ [\![\mathbf{VFFFFV}]\!], & [\![\mathbf{FVVFFF}]\!], & [\![\mathbf{FVFVFF}]\!], & [\![\mathbf{FVFFVF}]\!], \\ [\![\mathbf{FVFFFV}]\!], & [\![\mathbf{FFVVFF}]\!], & [\![\mathbf{FFVFVF}]\!], & [\![\mathbf{FFVFFV}]\!], \\ [\![\mathbf{FFFVVF}]\!], & [\![\mathbf{FFFVFV}]\!], & [\![\mathbf{FFFFVV}]\!] & \end{array} \right\}.$$

Note that the number of possible interleavings is $|\Xi_{Veg,Fruit}| = \binom{n}{p} = \binom{n}{q} = 6!/(2!4!) = 15$*. One possible riffle shuffling distribution on* S_6 *might, for example,*

assign uniform probability ($m^{unif}_{Veg,Fruit}(\sigma) = 1/15$) *to each permutation in* $\Xi_{Veg,Fruit}$ *and zero probability to everything else, reflecting indifference between vegetables and fruits. Figure 30b is a graphical example of a* $(2,4)$*-interleaving distribution.*

We now formally define our generalization of independence where a distribution which fully factors independently is allowed to undergo a single riffle shuffle.

Definition 80 (Riffled independence)**.** The subsets $A = \{1,\ldots,p\}$ and $B = \{p+1,\ldots,n\}$ are said to be *riffle independent* if

$$h = m_{p,q} * (f_A(\sigma(A)) \cdot g_B(\sigma(B))),$$

with respect to some interleaving distribution $m_{p,q}$ and distributions f_A, g_B, respectively. We will notate the riffled independence relation as $A \perp_m B$, and refer to f_A, g_B as *relative ranking factors*.

Notice that without the additional convolution, the definition of riffled independence reduces to the fully independent case given by Definition 63.

Example 81. *Consider drawing a ranking from a riffle independent model. One starts with two piles of cards,* A *and* B*, stacked together in a deck. In our fruits/vegetables setting, if we always prefer vegetables to fruits, then the vegetables occupy positions* $\{1,2\}$ *and the fruits occupy positions* $\{3,4,5,6\}$*. In the first step, rankings of each pile are drawn independent. For example, we might have the rankings:* $\sigma(Veg) = [2\,1]$ *and* $\sigma(Fruit) = [4\,6\,5\,3]$*, constituting a draw from the fully independent model described in Chapter 12. In the second stage, the deck of cards is cut and interleaved by an independently selected element* $\tau \in \Omega_{2,4}$*. For example, if:*

$$\tau = [2\,3\,1\,4\,5\,6] = [\![\text{Fruit}, \text{Veg}, \text{Veg}, \text{Fruit}, \text{Fruit}, \text{Fruit}]\!],$$

then the joint ranking is:

$$\begin{aligned}\tau(\sigma(\text{Veg}), \sigma(\text{Fruit})) &= [2\,3\,1\,4\,5\,6][2\,1\,4\,6\,5\,3] = [3\,2\,4\,6\,5\,1], \\ &= [\![\text{Grapes}, \text{Peas}, \text{Corn}, \text{Lemon}, \text{Fig}, \text{Orange}]\!].\end{aligned}$$

13.2.2 *Alternative definition of riffled independence*

It is possible to rewrite the definition of riffled independence so that it does not involve a convolution. We first define functions which map a given full ranking to relative rankings and interleavings for A and B.

Definition 82.

- (*Absolute ranks*): Given a ranking $\sigma \in S_n$, and a subset $A \subset \{1,\ldots,n\}$, $\sigma(A)$ denotes the *absolute ranks* of items in A.
- (*Relative ranking map*): Let $\phi_A(\sigma)$ denote the ranks of items in A *relative* to the set A. For example, in the ranking $\sigma = [\![\mathbf{P}, \mathbf{L}, \mathbf{F}, \mathbf{G}, \mathbf{C}, \mathbf{O}]\!]$, the relative ranks of the vegetables is $\phi_A(\sigma) = [\![\mathbf{P}, \mathbf{C}]\!] = [\![\text{Peas}, \text{Corn}]\!]$. Thus, while corn is ranked fifth in σ, it is ranked second in $\phi_A(\sigma)$. Similarly, the relative ranks of the fruits is $\phi_B(\sigma) = [\![\mathbf{L}, \mathbf{F}, \mathbf{G}, \mathbf{O}]\!] = [\![\text{Lemons}, \text{Figs}, \text{Grapes}, \text{Oranges}]\!]$.

- (*Interleaving map*): Likewise, let $\tau_{A,B}(\sigma)$ denote the way in which the sets A and B are interleaved by σ. For example, using the same σ as above, the interleaving of vegetables and fruits is $\tau_{A,B}(\sigma) = [\![Veg, Fruit, Fruit, Fruit, Veg, Fruit]\!]$. In ranking notation (as opposed to ordering notation), $\tau_{A,B}$ can be written as $(\text{sort}[\sigma(A)), \text{sort}(\sigma(B))]$. Note that for every possible interleaving, $\tau \in \Xi_{A,B}$ there are exactly $p! \times q!$ distinct permutations which are associated to τ by the interleaving map.

Using the above maps, the following lemma provides an algebraic expression for how any permutation σ can be uniquely decomposed into an interleaving composed with relative rankings of A and B, which have been "stacked" into one deck.

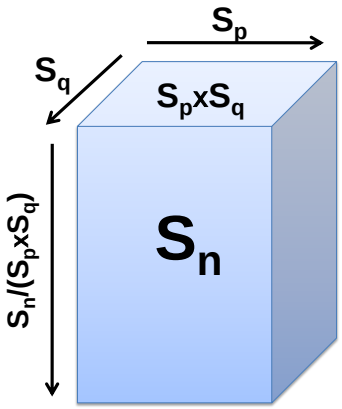


Figure 31: One way to think of S_n is as a three dimensional cube indexed by three coordinates — relative rankings of A and B, and the interleaving.

Lemma 83. *Let* $A = \{1, \ldots, p\}$, *and* $B = \{p+1, \ldots, n\}$. *Any ranking* $\sigma \in S_n$ *can be decomposed* uniquely *as an interleaving* $\tau \in \Xi_{A,B}$ *composed with a ranking of the form* $[\pi_p\ \pi_q + p]$ *(using one-line notation), where* $\pi_p \in S_p$, $\pi_q \in S_q$, *and* $\pi_q + p$ *means that the number* p *is added to every rank in* π_q. *Specifically,* $\sigma = \tau[\pi_p\ \pi_q + p]$ *with* $\tau = \tau_{A,B}(\sigma)$, $\pi_p = \phi_A(\sigma)$, *and* $\pi_q = \phi_B(\sigma)$.

Proof. Let i be an item in A (with $1 \leqslant i \leqslant p$). Since $\phi_A(\sigma) \in S_p$, $[\phi_A(\sigma)](i)$ is some number between 1 and p. By definition, for any $j \in \{1, \ldots, p\}$ the interleaving map $[\tau_{A,B}(\sigma)](j)$ returns the j^{th} largest rank in $\sigma(A)$. Thus, $[\tau_{A,B}(\sigma)](\phi_A(i))$ is the $\phi_A(i)$-th largest rank in $\sigma(A)$, which is simply the absolute rank of item i. Therefore, we conclude that $\sigma(i) = [\tau_{A,B}(\sigma)](\phi_A(i))$. Similarly, if $p+1 \leqslant i \leqslant n$, we have $\sigma(i) = [\tau_{A,B}(\sigma)](\phi_B(i) + p)$ (the added p is necessary since the indices of B are offset by p in σ), and we can conclude that $\sigma = \tau_{A,B}(\sigma)[\phi_A(\sigma)\ \phi_B(\sigma)]$. □

Lemma 83 shows that one can think of a triplet $(\tau \in \Xi_{A,B}, \sigma_p \in S_p, \sigma_q \in S_q)$ as being coordinates which uniquely specify any ranking of items in $A \cup B$ (see Figure 31 for an illustration). Using the decomposition, we can now state a second, perhaps more intuitive, definition of riffled independence in terms of the relative ranking and interleaving maps.

Definition 84. Sets A and B are said to be *riffle independent* if and only if, for every $\sigma \in S_n$, the joint distribution h factors as:

$$h(\sigma) = m(\tau_{A,B}(\sigma)) \cdot f_A(\phi_A(\sigma)) \cdot g_B(\phi_B(\sigma)). \tag{13.1}$$

Proposition 85. *Definitions 80 and 84 are equivalent.*

Proof. Assume that $A = \{1, \ldots, p\}$ and $B = \{p+1, \ldots, n\}$ are riffle independent with respect to Definition 80. We will show that Definition 84 is also satisfied (the opposite direction will be similar). Therefore, we assume that $h = m_{A,B} * (f_A(\sigma(A)) \cdot g_B(\sigma(B)))$. Note that $f(\sigma_A) \cdot g(\sigma_B)$ is supported on the subgroup $S_p \times S_q \equiv \{\sigma \in S_n : 1 \leqslant \sigma(i) \leqslant p, \text{whenever } 1 \leqslant i \leqslant p\}$.

Let $\sigma = (\sigma(A), \sigma(B))$ be any ranking. We will need to use a simple claim: consider the ranking $\tau^{-1}\sigma$ (where $\tau \in \Omega_{p,q}$). Then $\tau^{-1}\sigma$ is an element of the subgroup $S_p \times S_q$ if and only if $\tau = \tau_{A,B}(\sigma)$.

$$
\begin{aligned}
[m_{A,B} * (f_A \cdot g_B)](\sigma) &= \sum_{\sigma' \in S_n} m_{A,B}(\sigma') \cdot [f_A \cdot B](\sigma'^{-1}\sigma), \\
&= \sum_{\tau \in \Xi_{A,B}} m_{A,B}(\tau) \cdot [f_A \cdot g_B](\tau^{-1}\sigma), \\
&\qquad \text{(since } m_{A,B} \text{ is supported on } \Xi_{A,B}\text{)} \\
&= m_{A,B}(\tau_{A,B}(\sigma)) \cdot [f_A \cdot g_B]((\tau_{A,B}^{-1}(\sigma))\sigma), \\
&\qquad \text{(by the claim above and since } f_A \cdot g_B \\
&\qquad\qquad \text{is supported on } S_p \times S_q\text{)} \\
&= m_{A,B}(\tau_{A,B}(\sigma)) \cdot [f_A \cdot g_B](\phi_A(\sigma), \phi_B(\sigma)), \\
&\qquad \text{(by Lemma 83)} \\
&= m_{A,B}(\tau_{A,B}(\sigma)) \cdot f_A(\phi_A(\sigma)) \cdot g_B(\phi_B(\sigma)). \\
&\qquad \text{(by independence of } f_A \cdot g_B\text{)}
\end{aligned}
$$

Thus, we have shown that Definition 84 has been satisfied as well. □

13.2.3 *Discussion*

We have presented two ways of thinking about riffled independence. Our first formulation, in terms of convolution, is motivated by the connections between riffled independence and card shuffling theory. As we show in Section 13.4, the convolution based view is also crucial for working with Fourier coefficients of riffle independent distributions and analyzing the theoretical properties of riffled independence. Our second formulation on the other hand, shows the concept of riffled independence to be remarkably simple — that the probability of a single ranking can be computed without summing over all rankings (required in convolution) — a fact which may not have been obvious from Definition 80.

Finally, for interested readers, the concept of riffled independence also has a simple and natural group theoretic description. By a fully factorized distribution, we refer to a distribution supported on the subgroup $S_p \times S_q$, which factors along the S_p and S_q "dimensions". As we have discussed, such sparse distributions are not appropriate for ranking applications, and one would like to work with distributions capable of placing nonzero probability mass on all rankings. In the case of the symmetric group, however, there is a third "missing dimension" — the coset space, $S_n/(S_p \times S_q)$. Thus, the natural extension of full independence is to randomize over a set of coset representatives of $S_p \times S_q$, what we have referred to in the

above discussion as *interleavings* (see also Chapter 2). The draws from each set, S_p, S_q, and $S_n/(S_p \times S_q)$ are then independent in the ordinary sense, and we say that the item sets A and B are riffle independent.

SPECIAL CASES. There are a number of special case distributions captured by the riffled independence model that are useful for honing intuition. We discuss these extreme cases in the following list.

- (*Uniform and delta distributions*): Setting the interleaving distribution and both relative ranking factors to be uniform distributions yields the uniform distribution over all full rankings. Similarly, setting the same distributions to be delta distributions (which assign zero probability to all rankings but one) always yields a delta distribution.

  It is interesting to note that while A and B are always fully independent under a delta distribution, they are never independent under a uniform distribution. However, both uniform and delta distributions factor *riffle independently* with respect to any partitioning of the item set. Thus, not only is $A = \{1, \ldots, p\}$ riffle independent $B = \{p+1, \ldots, n\}$, but in fact, any set A is riffle independent of its complement.

- (*Uniform interleaving distributions*): Setting the interleaving distribution to be uniform, as we will discuss more in detail later, reflects complete indifference between the sets A and B, even if f_A and g_B encode complex preferences within each set alone.

- (*Uniform relative ranking factors*): Setting the relative ranking factors, f_A and g_B to be uniform distributions means that with respect to the joint distribution h, all items in A are completely interchangeable amongst each other (as are all items in B).

- (*Delta interleaving distributions*): Setting the interleaving distribution, $m_{A,B}$, to be a delta distribution on *any* of the (p, q)-interleavings in $\Xi_{A,B}$ recovers the definition of ordinary probabilistic independence, and thus riffled independence is a strict generalization thereof (see Figure 28). Just as in the full independence regime, where the distributions f_A and g_B are marginal distributions of absolute rankings of A and B, in the riffled independence regime, f_A and g_B can be thought of as marginal distributions of the *relative rankings* of item sets A and B.

- (*Delta relative ranking factor*): On the other hand, if one of the relative ranking factors, say f_A, is a delta distribution and the other two distributions $m_{A,B}$ and g_B are uniform, then the resulting riffle independent distribution h can be thought of as an indicator function for the set of rankings that are consistent with one particular incomplete ranking (in which only the relative ranking of A has been specified). Such distributions can be useful in practice when the input

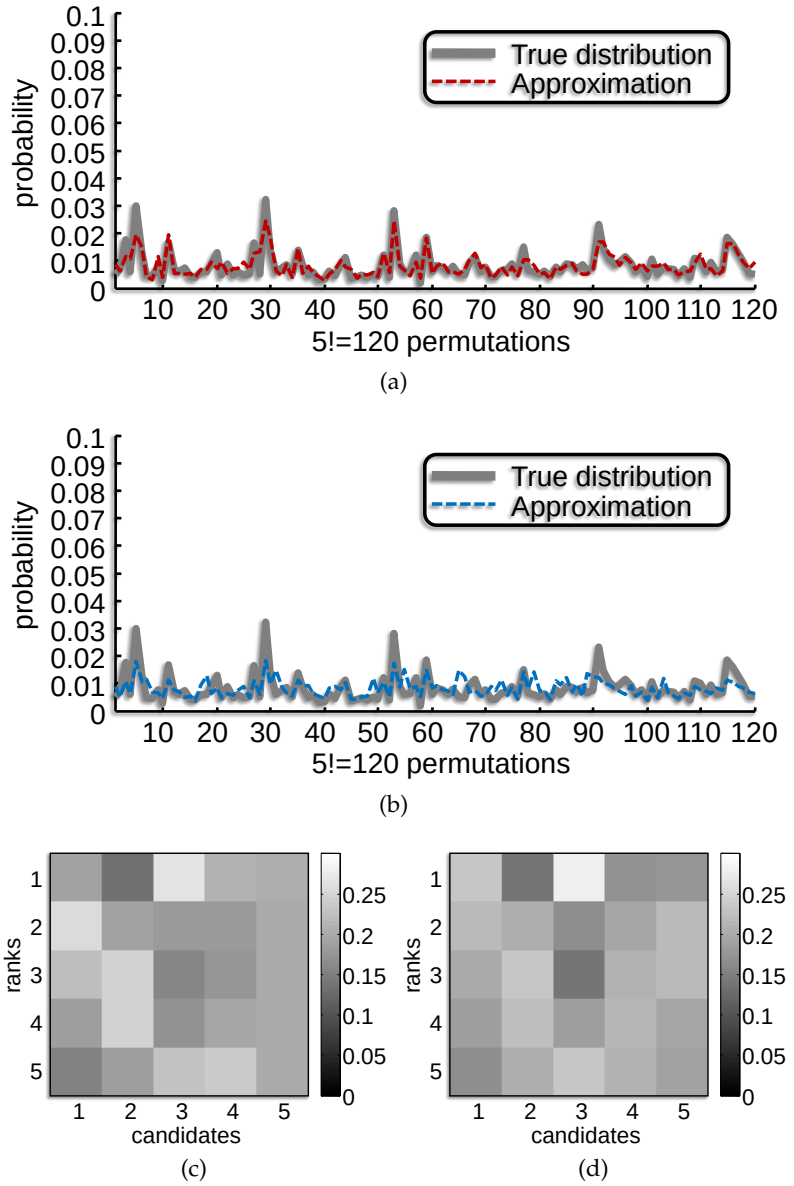


Figure 32: Approximating the APA vote distribution by riffle independent distributions. (a) approximate distribution when candidate 2 is riffle independent of remaining candidates; (b) approximate distribution when candidate 3 is riffle independent of remaining candidates; (c) and (d) corresponding first order marginals of each approximate distribution.

data comes in the form of incomplete rankings rather than full rankings. See Kondor and Barbosa [81] who use a similar factorization for efficiently computing kernel evaluations between partial rankings.

Example 86 (APA election data (continued))**.** *Like the independence assumptions commonly used in naive Bayes models, we would rarely expect riffled independence to exactly hold in real data. Instead, it is more appropriate to view riffled independence assumptions as a form of model bias that ensures learnability*

Algorithm 13.1: Recurrence for drawing $\sigma \sim m^{unif}_{p,q}$ (Base case: return $\sigma = [1]$ if $n = 1$).

DRAWRIFFLEUNIF(p, q, n) $//(p + q = n)$

with probability q/n *//drop from right pile*

$\sigma^- \leftarrow$ DRAWRIFFLEUNIF$(p, q-1, n-1)$;

foreach *i* **do** $\sigma(i) \leftarrow \begin{cases} \sigma^-(i) & \text{if } i < n \\ n & \text{if } i = n \end{cases}$;

endif

otherwise *//drop from left pile*

$\sigma^- \leftarrow$ DRAWRIFFLEUNIF$(p-1, q, n-1)$;

foreach *i* **do** $\sigma(i) \leftarrow \begin{cases} \sigma^-(i) & \text{if } i < p \\ n & \text{if } i = p \\ \sigma^-(i-1) & \text{if } i > p \end{cases}$;

endif

return (σ) ;

for small sample sizes, which as we have indicated, is almost always the case for distributions over rankings.

Can we ever expect riffled independence to be manifested in a real dataset? In Figure 32a, we plot (in dotted red) a riffle independent approximation to the true APA vote distribution (in thick gray) which is optimal with respect to KL-divergence (we will explain how to obtain the approximation in the remainder of the chapter). The approximation in Figure 32a is obtained by assuming that the candidate set $\{1, 3, 4, 5\}$ is riffle independent of $\{2\}$, and as can be seen, is quite accurate compared to the truth (with the KL-divergence from the true to the factored distribution being $d_{KL} = .0398$). Figure 32c exhibits the first order marginals of the approximating distribution, which can also visually be seen to be a faithful approximation (see Figure 10b for the empirical first-order matrix of marginals). We will discuss the interpretation of the result further in Chapter 14.

For comparison, we also display (in Figures 32b and 32d) the result of approximating the true distribution by one in which candidate $\{3\}$, the winner, is riffle independent of the remaining candidate. The resulting approximation is inferior, and the lesson to be learned in the example is that finding the correct/optimal partitioning of the item set is important in practice. We remark however, that the approximation obtained by factoring out candidate 3 is not a terrible approximation (especially on examining first order marginals), and that both approximations are far more accurate than the fully independent approximation showed earlier in Figure 29. The KL divergence from the true distribution to the factored distribution (with candidate 3 riffle independent of the remaining candidates) is $d_{KL} = .0841$.

13.3 INTERLEAVING DISTRIBUTIONS

There is, in the general case, a significant increase in storage required for riffled independence over full independence. In addition to the $O(p! + q!)$ storage required for distributions f and g, we now require $O(\binom{n}{p})$ storage

for the nonzero terms of the riffle shuffling distribution $m_{A,B}$. In this thesis we will primarily consider models with small p ($p \sim O(1)$), in which case the number of model parameters scales polynomially in n. In this section, however, we consider a family of useful riffle shuffling distributions which can be described using only a handful of parameters. The simplest riffle shuffling distribution is the *uniform riffle shuffle*, $m^{unif}_{A,B}$, which assigns uniform probability to all (p, q)-interleavings and zero probability to all other elements in S_n. Used in the context of riffled independence, $m^{unif}_{A,B}$ models potentially complex relations within A and B, but only captures the simplest possible correlations across subsets. We might, for example, have complex preference relations amongst vegetables and amongst fruits, but be completely indifferent with respect to the subsets, vegetables and fruits, as a whole.

There is a simple recursive method for drawing (p, q)-interleavings from a uniform distribution. Starting with a deck of n cards cut into a left pile ($\{1, \ldots, p\}$) and a right pile ($\{p+1, \ldots, n\}$), pick one of the piles with probability proportional to its size (p/n for the left pile, q/n for the right) and drop the bottommost card, thus mapping either card p or card n to rank n. Then recurse on the $n-1$ remaining undropped cards, drawing a $(p-1, q)$-interleaving if the right pile was picked, or a $(p, q-1)$-interleaving if the left pile was picked. See Algorithm 13.1. We have the following:

Theorem 87. *Algorithm 13.1 returns a uniformly distributed (p, q)-interleaving.*

Proof. The proof is by induction on $n = p + q$. The base case (when $n = 1$) is obvious since the algorithm can only return a single permutation.

Next, we assume for the sake of induction that for any $m < n$, the algorithm returns a uniformly distributed interleaving and we want to show this to also be the case for n.

Let τ be any interleaving in $\Xi_{A,B}$. We will show that $m(\tau) = 1/\binom{n}{p}$. Consider $\tau^- = \tau^{-1}(1 : n-1) \in S_{n-1}$. There are two cases: τ^{-1} is either a $(p, q-1)$-interleaving (in which case $\tau(n) = n$), or a $(p-1, q)$-interleaving (in which case $\tau(p) = n$).

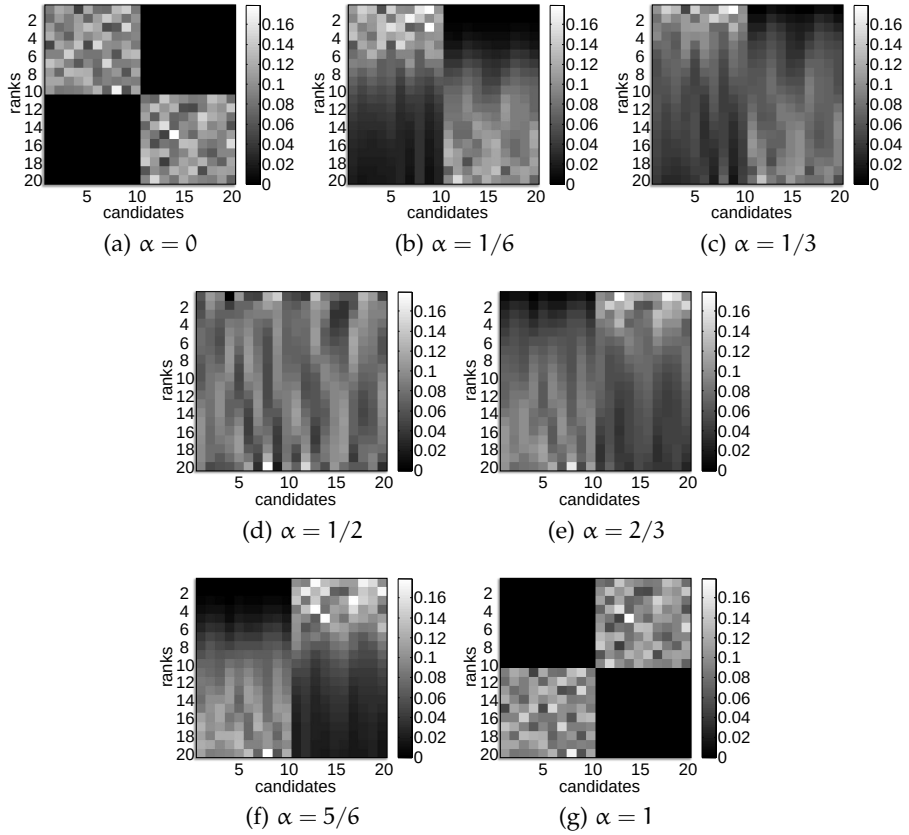
We will just consider the first case since the second is similar. τ^{-1} is uniformly distributed by the inductive hypothesis and therefore has probability $1/\binom{n-1}{p}$.

$\tau^{-1}(n)$ is set to n independently with probability q/n, so we compute the probability of the interleaving resulting from the algorithm as:

$$\frac{n-p}{n} \cdot \frac{1}{\binom{n-1}{p}} = \frac{n-p}{n} \cdot \frac{p!(n-1-p)!}{(n-1)!} = \frac{p!(n-p)!}{n!} = \frac{1}{\binom{n}{p}}.$$

□

It is natural to consider generalizations where one is preferentially biased towards dropping cards from the left hand over the right hand (or vice-versa). We model this bias using a simple one-parameter family of distributions in which cards from the left and right piles drop with probability proportional to αp and $(1-\alpha)q$, respectively, instead of p and q.



(a) $\alpha = 0$ (b) $\alpha = 1/6$ (c) $\alpha = 1/3$

(d) $\alpha = 1/2$ (e) $\alpha = 2/3$

(f) $\alpha = 5/6$ (g) $\alpha = 1$

Figure 33: First-order matrices with a deck of 20 cards, $A = \{1, \ldots, 10\}$, $B = \{11, \ldots, 20\}$, *riffle independent* and various settings of α. Compare these matrices to the fully independent first order marginal matrices of Figure 28 and note that here, the nonzero blocks are allowed to 'bleed' into zero regions. Setting $\alpha = 0$ or 1, however, recovers the fully independent case, where a subset of objects is preferred over the other with probability one.

We will refer to α as the *bias parameter*, and the family of distributions parameterized by α as the *biased riffle shuffles*.[1]

In the context of rankings, biased riffle shuffles provide a simple model for expressing groupwise preferences (or indifference) for an entire subset A over B or vice-versa. The bias parameter α can be thought of as a knob controlling the preference for one subset over the other, and might reflect, for example, a preference for fruits over vegetables, or perhaps indifference between the two subsets. Setting $\alpha = 0$ or 1 recovers the full independence assumption, preferring objects in A (vegetables) over objects in B (fruits) with probability one (or vice-versa), and setting $\alpha = .5$, recovers the uniform riffle shuffle (see Figure 33). Finally, there are a number of straightforward generalizations of the biased riffle shuffle that one can use to realize richer

1 The recurrence in Algorithm 13.1 has appeared in various forms in literature [11]. We are the first to (1) use the recurrence to Fourier transform $m_{p,q}$, and to (2) consider biased versions. The biased riffle shuffles in [41] are not similar to our biased riffle shuffles.

distributions. For example, α might depend on the number of cards that have been dropped from each pile (allowing perhaps, for distributions to prefer crunchy *fruits* over crunchy *vegetables*, but soft *vegetables* over soft *fruits*).

13.3.1 *Fourier transforming the biased riffle shuffle*

We now use the recursive generative procedure to compute the Fourier transform of $m^{unif}_{A,B}$. To make the dependence on p and q explicit for the purposes of this section, we will write $m^{unif}_{p,q}$ instead of $m^{unif}_{A,B}$ for $A = \{1, \ldots, p\}$, and $B = \{p+1, \ldots, p+q\}$. We now describe the recurrence satisfied by $m^{unif}_{p,q}$, allowing one to write $m^{unif}_{p,q}$, a distribution on S_n, in terms of $m^{unif}_{p,q-1}$ and $m^{unif}_{p-1,q}$, distributions over S_{n-1}, using EMBED and SHIFT operations. Algorithm 13.1 can be then rephrased as a recurrence relation as follows.

Proposition 88. *The uniform riffle shuffling distribution* $m^{unif}_{p,q}$ *obeys the recurrence relation:*

$$m^{unif}_{p,q} = \left[\left(\frac{p}{p+q}\right) \cdot \textsc{Shift}[\textsc{Embed}[m^{unif}_{p-1,q}], (p+1, \ldots, n)^{-1}, \epsilon]\right] + \left[\left(\frac{q}{p+q}\right) \cdot \textsc{Embed}[m^{unif}_{p,q-1}]\right], \tag{13.2}$$

with base cases: $m^{unif}_{0,n} = m^{unif}_{n,0} = \delta_\epsilon$, *where* δ_ϵ *is the delta function at the identity permutation.*

Note that by taking the support sizes of each of the functions in the above recurrence, we recover the following well known recurrence for binomial coefficients:

$$\binom{n}{p} = \binom{n-1}{p-1} + \binom{n-1}{p}, \text{ with base case } \binom{n}{0} = \binom{n}{n} = 1. \tag{13.3}$$

The *biased riffle shuffle* is defined by:

$$m^{\alpha}_{p,q} \propto \left[\left(\frac{\alpha p}{p+q}\right) \cdot \textsc{Shift}[\textsc{Embed}[m^{\alpha}_{p-1,q}], (p+1, \ldots, n)^{-1}, \epsilon]\right] + \left[\left(\frac{(1-\alpha)q}{p+q}\right) \cdot \textsc{Embed}[m^{\alpha}_{p,q-1}]\right], \tag{13.4}$$

Writing the recursion in the form of Equation 13.2 provides a construction of the uniform riffle shuffle as a sequence of operations on smaller distribution which can be performed completely with respect to Fourier coefficients. Using linearity of Fourier transforms, the convolution theorem (Proposition 51 and the fact that embeddings can be performed in the Fourier domain (Algorithm 6.4), we arrive at the equivalent Fourier-theoretic recurrence for each frequency level i.

$$\{\widehat{m^{unif}_{p,q}}_\rho\}_{\rho \in \Lambda_n} = \left(\frac{p}{p+q}\right) \cdot \textsc{Shift}[\textsc{Embed}[\{\widehat{m^{unif}_{p-1,q}}_\rho\}_{\rho \in \Lambda_{n-1}}], (p+1, \ldots, n)^{-}, \epsilon] + \left(\frac{q}{p+q}\right) \cdot \textsc{Embed}[\{\widehat{m^{unif}_{p,q-1}}_\rho\}_{\rho \in \Lambda_{n-1}}]. \tag{13.5}$$

Algorithm 13.2: Pseudocode for computing the Fourier transform of the uniform riffle shuffling distribution using dynamic programming. Input: Integers p and q. Output: The Fourier coefficient matrices of $\{\widehat{m}_\rho\}_{\rho \in \Lambda_n}$, where $m \equiv m_{A,B}^{unif}$.

```
RiffleHat(p, q):
    n ← p + q ;
    Initialize m̂^prev, m̂^curr as arrays of p + 1 Fourier transform data structures ;
    for i = 1, 2, ..., n do
        for j = max(0, p − n + i), ..., min(i, p) do
            if j == 0 or j == i then
                m̂^curr[j] ← δ̂_{ε∈S_i} ;
            end
            else
                m̂^curr[j] ← ((i−j)/i) Embed(m̂^prev[j], i − 1, i)
                + (i choose j) Convolve(Embed(m̂^prev[j − 1], i − 1, i), δ̂_{(i,i−1,...,j)});
            end
            m̂^prev ← m̂^curr ;
        end
    end
    return (m̂^curr[p]);
```

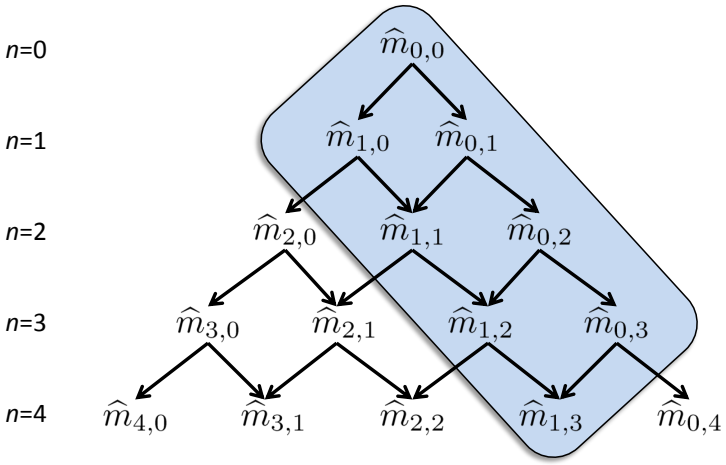


Figure 34: The flow of information in Algorithm 13.2 bears much resemblance to Pascal's triangle for computing binomial coefficients. The arrows in this diagram indicate the Fourier transforms that must be precomputed before computing the Fourier transform of a larger interleaving distribution. For example, to compute $\widehat{m}_{1,2}$, one must first compute $\widehat{m}_{1,1}$ and $\widehat{m}_{0,2}$. $\widehat{m}_{1,1}$ and $\widehat{m}_{0,2}$ in turn require $\widehat{m}_{1,0}$ and $\widehat{m}_{0,1}$. In blue, we have highlighted the collection of Fourier transforms that are computed by Algorithm 13.2 while computing $\widehat{m}_{1,3}$

Implementing the recurrence (Equation 13.5) in code can naively result in an exponential time algorithm if one is not careful. It is necessary to use dynamic programming to be sure not to recompute things that were already computed. In Algorithm 13.2, we present pseudocode of such a dynamic programming approach, which builds a 'Pascal's triangle' similar to that which might be constructed to compute a table of binomial coefficients.

The pseudocode assumes the existence of Fourier domain algorithms for convolving distributions and for embedding a distribution over S_{n-1} into S_n. See Figure 34 for a graphical illustration of the algorithm. Note that a simple modification of the scalar constants in Algorithm 13.2 allows us to compute the Fourier transform of the biased riffle shuffle as well.

13.4 ALGORITHMS FOR A FIXED PARTITIONING OF THE ITEM SET

We have thus far covered a number of intuitive examples and properties of riffled independence. Given a set of rankings drawn from some distribution h, we are now interested in estimating a number of statistical quantities, such as the parameters of a riffle independent model. In this section, we will assume a *known structure* (that the partitioning of the item set into subsets A and B is known), and given such a partitioning of the item set, we are interested in the problem of estimating parameters (which we will refer to as *RiffleSplit*), and the inverse problem of computing probabilities (or marginal probabilities) with given parameters (which we will refer to as *RiffleJoin*).

RIFFLESPLIT . In RiffleSplit (which we will also refer to as the *parameter estimation* problem), we would like to estimate various statistics of the relative ranking and interleaving distributions of a riffle independent distribution ($m_{A,B}$, f_A, and g_B). Given a set of i.i.d. training examples, $\sigma^{(1)}, \ldots, \sigma^{(m)}$, we might, for example, want to estimate each raw probability (e.g., estimate $m_{A,B}(\tau)$ for each interleaving τ). In general, we may be interested in estimating more general statistics (e.g., what are the second order relative ranking probabilities of the set of fruits?).

Since our variables are discrete, computing the maximum likelihood parameter estimates consists of forming counts of the number of training examples consistent with a given interleaving or relative ranking. Thus, the MLE parameters in our problem are simply given by the following formulas:

$$m_{A,B}^{MLE}(\tau) \propto \sum_{i=1}^{m} \mathbb{1}\left[\tau = \tau_{A,B}(\sigma^{(i)})\right], \tag{13.6}$$

$$f_A^{MLE}(\sigma_A) \propto \sum_{i=1}^{m} \mathbb{1}\left[\sigma_A = \phi_A(\sigma^{(i)})\right], \tag{13.7}$$

$$g_B^{MLE}(\sigma_B) \propto \sum_{i=1}^{m} \mathbb{1}\left[\sigma_B = \phi_B(\sigma^{(i)})\right]. \tag{13.8}$$

RIFFLEJOIN . Having estimated parameters of a riffle independent distribution, we would like to now compute various statistics of the data itself. In the simplest case, we are interested in estimating $h(\sigma)$, the joint probability of a single ranking, which can be evaluated simply by plugging parameter estimates of $m_{p,q}$, f_A, and g_B into our second definition of riffled independence (Definition 84).

Algorithm 13.3: Pseudocode for RIFFLEJOIN in the Fourier domain. Input: Fourier transforms of f_A, g_B, and $m_{A,B}$ ($\{\widehat{f}_\rho\}_{\rho\in\Lambda_p}$, $\{\widehat{g}_\rho\}_{\rho\in\Lambda_q}$, and $\{\widehat{m}_\rho\}_{\rho\in\Lambda_n}$ respectively). Output: Fourier transform of the joint distribution, $\{\widehat{h}_\rho\}_{\rho\in\Lambda_n}$.

RIFFLEJOIN($\{\widehat{f}_\rho\}_{\rho\in\Lambda_p}$,$\{\widehat{g}_\rho\}_{\rho\in\Lambda_q}$,$\{\widehat{m}_\rho\}_{\rho\in\Lambda_n}$)

$\{\widehat{h'}_\rho\}_{\rho\in\Lambda_n} = \text{JOIN}[\{\widehat{f}_\rho\}_{\rho\in\Lambda_p},\{\widehat{g}_\rho\}_{\rho\in\Lambda_q}]$;
foreach *partition* λ *of* n **do**
  $\widehat{h}_{\rho_\lambda} \leftarrow \widehat{m}_{\rho_\lambda} \cdot \widehat{h'}_{\rho_\lambda}$;
end
return $\left(\{\widehat{h}_\rho\}_{\rho\in\Lambda}\right)$;

More generally however, we may be interested in knowing the low-order statistics of the data (e.g., the first order marginals, second order marginals, etc.), or related statistics (such as $h(\sigma(i) < \sigma(j))$, the probability that object i is preferred to object j). And typically for such low-order statistics, one must compute a sum over rankings. For example, to compute the probability that item j is ranked in position i, one must sum over $(n-1)!$ rankings:

$$h(\sigma : \sigma(j) = i) = \sum_{\sigma:\sigma(j)=i} m_{A,B}(\tau_{A,B}(\sigma)) \cdot (f_A(\phi_A(\sigma)) \cdot g_B(\phi_B(\sigma))). \quad (13.9)$$

While Equation 13.9 may be feasible for small n (such as on the APA dataset), the sum quickly grows to be intractable for larger n. One of the main observations of the remainder of this section, however, is that low-order marginal probabilities of the joint distribution can always be computed directly from low-order marginal probabilities of the relative ranking and interleaving distributions without explicitly computing intractable sums.

13.4.1 *Fourier theoretic algorithms for riffled independence*

We now present algorithms for working with riffled independence (solving the RiffleSplit and RiffleJoin problems) in the Fourier theoretic framework of Chapter 8. The Fourier theoretic perspective of riffled independence presented here is valuable because it will allow us to work directly with low-order statistics instead of having to form the necessary raw probabilities first. Note that readers who are primarily interested in the structure learning can jump directly to Chapter 14.

In this section, we provide generalizations of the algorithms Chapter 12 that tackle the RiffleJoin and RiffleSplit problems. We will assume, without loss of generality that $A = \{1, \ldots, p\}$ and $B = \{p+1, \ldots, n\}$ (this assumption will be discarded in later sections), Although we begin each of the following discussions as if all of the Fourier coefficients are provided, we will be especially interested in algorithms that work well in cases where only a truncated set of Fourier coefficients are present, and where h is only *approximately* riffle independent.

Algorithm 13.4: Pseudocode for RIFFLESPLIT in the Fourier domain. Input: Fourier transform of the empirical joint distribution, $\{\widehat{h}_\rho\}_{\rho\in\Lambda_n}$. Output: Fourier transform of MLE estimates of f_A, g_B ($\{\widehat{f}_\rho\}_{\rho\in\Lambda_p}$,$\{\widehat{g}_\rho\}_{\rho\in\Lambda_q}$).

RIFFLESPLIT($\{\widehat{h}_\rho\}_{\rho\in\Lambda_n}$)

foreach *partition* λ *of* n **do**
$\widehat{h'}_{\rho_\lambda} \leftarrow \left[\widehat{m}^{unif}_{A,B}\right]^T_{\rho_\lambda} \cdot \widehat{h}_{\rho_\lambda}$;
end
$[\{\widehat{f}_\rho\}_{\rho\in\Lambda_p},\{\widehat{g}_\rho\}_{\rho\in\Lambda_q}] \leftarrow$ SPLIT($\widehat{h'}$) ;
NORMALIZE $\left(\{\widehat{f}_\rho\}_{\rho\in\Lambda_p}\right)$;
NORMALIZE $\left(\{\widehat{g}_\rho\}_{\rho\in\Lambda_q}\right)$;
return $\left(\{\widehat{f}_\rho\}_{\rho\in\Lambda_p},\{\widehat{g}_\rho\}_{\rho\in\Lambda_q}\right)$;

13.4.2 RIFFLEJOIN *in the Fourier domain*

Given the Fourier coefficients of f_A, g_B, and $m_{A,B}$, we can compute the Fourier coefficients of h using Definition 80 (our first definition) by applying JOIN (Algorithm 12.1) and the convolution theorem (Proposition 51), which tells us that the Fourier transform of a convolution can be written as a pointwise product of Fourier transforms. To compute the $\hat{h}_{\rho_\lambda}$, the Fourier theoretic formulation of the *RiffleJoin* algorithm simply calls the Join algorithm on $\{\widehat{f}_\rho\}_{\rho\in\Lambda_p}$ and $\{\widehat{g}_\rho\}_{\rho\in\Lambda_q}$, and convolves the result by $\{\widehat{m}_\rho\}_{\rho\in\Lambda_n}$, where we have dropped the A and B subscripts for succinctness (see Algorithm 13.3).

In general, it may be intractable to Fourier transform the riffle shuffling distribution m. However, there are some cases in which m can be computed. For example, if m is computed directly from a set of training examples, then one can simply compute the desired Fourier coefficients using the definition of the Fourier transform (i.e., the *direct construction* of Chapter 7). For the class of biased riffle shuffles that we discussed in Section 13.2, we have also shown an algorithm (Algorithm 13.2) for efficiently computing the low-frequency terms of $\widehat{m^\alpha_{A,B}}$ by employing the recurrence relation in Algorithm 13.1.

13.4.3 RIFFLESPLIT *in the Fourier domain*

Given the Fourier coefficients of a riffle independent distribution h, we would like to tease apart the factors. In the following, we show how to recover the relative ranking distributions, f_A and g_B, and defer the problem of recovering the interleaving distribution for Appendix 13.3.1.

From the RIFFLEJOIN algorithm, we saw that for each partition λ, $\hat{h}_{\rho_\lambda} = \widehat{m}_{\rho_\lambda} \cdot [\widehat{f \cdot g}]_{\rho_\lambda}$. The first solution to the splitting problem that might occur is to perform a deconvolution by multiplying each $\widehat{h}_\lambda$ term by the inverse of the matrix $\widehat{m}_{\rho_\lambda}$ (to form $\widehat{m}^{-1}_{\rho_\lambda} \cdot \widehat{h}_{\rho_\lambda}$) and call SPLIT (Algorithm 12.2) on the result. Unfortunately, the matrix $\widehat{m}_{\rho_\lambda}$ is, in general, non-invertible. Instead, our RIFFLESPLIT algorithm left-multiplies each $\widehat{h}_{\rho_\lambda}$ term by $[\widehat{m}^{unif}_{A,B}]^T_{\rho_\lambda}$,

which can be shown to be equivalent to convolving the distribution h by the '*dual shuffle*', m^*, defined as

$$m^*(\sigma) = m^{unif}_{A,B}(\sigma^{-1}).$$

While convolving by m^* does not produce a distribution that factors independently, the SPLIT algorithm can still be shown to recover the Fourier transforms $\{\widehat{f^{MLE}_A}_\rho\}_{\rho\in\Lambda_p}$ and $\{\widehat{g^{MLE}_B}_\rho\}_{\rho\in\Lambda_q}$ of the maximum likelihood parameter estimates:

Theorem 89. *Given a set of rankings with empirical distribution $\tilde{h}$, the maximum likelihood estimates of the relative ranking distributions over item sets A and B are given by:*

$$[f^{MLE}_A, g^{MLE}_B] \propto \textsc{Split}\left[m^*_{p,q} * \tilde{h}\right], \tag{13.10}$$

where $m^*_{p,q}$ is the dual shuffle (of the uniform interleaving distribution). Furthermore, the Fourier transforms of the relative ranking distributions are:

$$\left[\{\widehat{f^{MLE}}_\rho\}_{\rho\in\Lambda_p}, \{\widehat{g^{MLE}}_\rho\}_{\rho\in\Lambda_q}\right] \propto \textsc{Split}\left[\left\{\left(\widehat{m}^{unif}_{A,B}\right)^T_{\rho_\lambda} \cdot \hat{\tilde{h}}_{\rho_\lambda}\right\}_{\lambda\in\Lambda_n}\right].$$

Proof. We will use $\pi_p \in S_p$ and $\pi_q \in S_q$ to denote relative rankings of A and B respectively. Let us consider estimating $f^{MLE}_A(\pi_p)$. If $\tilde{h}$ is the empirical distribution of the training examples, then $f^{MLE}_A(\pi_p)$ can be computed by summing over examples in which the relative ranking of A is consistent with π_p (Equation 13.7), or equivalently, by marginalizing $\tilde{h}$ over the interleavings and the relative rankings of B. Thus, we have:

$$f^{MLE}_A(\pi_p) = \sum_{\pi_q\in S_q}\left(\sum_{\tau\in\Xi_{A,B}} \tilde{h}(\tau[\pi_p, \pi_q + p])\right), \tag{13.11}$$

where we have used Lemma 83 to decompose a ranking σ into its component relative rankings and interleaving.

The second step is to notice that the outer summation of Equation 13.11 is exactly the type of marginalization that can already be done in the Fourier domain via the Split algorithm of [60], and thus, f^{MLE}_A can be rewritten as $f^{MLE}_A = \textsc{Split}(h')$, where the function $h' : S_n \to \mathbb{R}$ is defined as $h'(\sigma) = \sum_{\tau\in\Xi_{A,B}} h(\tau\sigma)$. Hence, if we could compute the Fourier transform of the function h', then we could apply the ordinary Split algorithm to recover the Fourier transform of f^{MLE}_A.

In the third step, we observe that the function h' can be written as a convolution of the dual shuffle with h, thus establishing the first part of the theorem:

$$\begin{aligned} h'(\sigma) &= \sum_{\tau\in\Xi_{A,B}} h(\tau\sigma), \\ &\propto \sum_{\pi\in S_n} m^{unif}_{A,B}(\pi)h(\pi\sigma), \\ &\propto \sum_{\pi\in S_n} m^*_{A,B}(\pi)h(\pi^{-1}\sigma), \\ &\propto [m^*_{A,B} * h](\sigma). \end{aligned}$$

Next, we use a standard fact about Fourier transforms [29] — given a function $m^* : S_n \to \mathbb{R}$ defined as $m^*(\sigma) = m(\sigma^{-1})$, the Fourier coefficient matrices of m^* are related to those of m by the transpose. Hence, $\hat{m^*}_{\rho_\lambda}^T = \hat{m}_{\rho_\lambda}$, for every partition λ of n. Applying the convolution theorem (Proposition 51) to the Fourier coefficients of the dual shuffle and the empirical distribution establishes the final part of the theorem. □

Notice that to compute the MLE relative ranking factors in the Fourier domain, it is not necessary to know the interleaving distribution. It is necessary, however, to compute the Fourier coefficients of the *uniform* interleaving distribution ($m^{unif}_{p,q}$), which we have discussed in the previous section. It is also necessary to normalize the output of Split to sum to one, which we accomplish again by dividing each Fourier coefficient matrix by $\left[\widehat{m^{unif}_{p,q}}\right]_{(n)}$ (Equation 8.1). See Algorithm 13.4 for pseudocode.

13.4.4 *Marginal preservation*

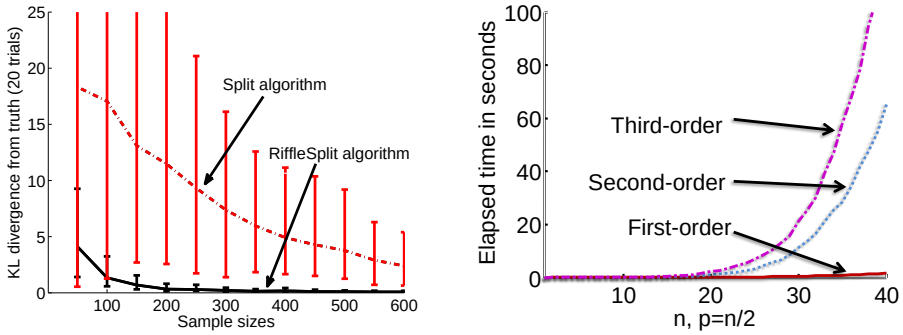
Typically, in the Fourier setting, one hopes instead to work with a set of low-order terms. For example, in the case of RiffleJoin, we might only receive the second order marginals of the parameter distributions as input. A natural question to ask then, is what is the approximation quality of the output given a bandlimited input? We now state a result below, which shows how our algorithms perform when called with a truncated set of Fourier coefficients.

Theorem 90. *Let* A *and* B *be riffle independent sets with joint distribution* h. *Given enough Fourier terms to reconstruct the* k^{th}*-order marginals of* f_A *and* g_B, *RiffleJoin returns enough Fourier terms to* exactly *reconstruct the* k^{th}*-order marginals of* h. *Likewise, given enough Fourier terms to reconstruct the* k^{th}*-order marginals of* h, *RiffleSplit returns enough Fourier terms to* exactly *reconstruct the* k^{th}*-order marginals of both* f_A *and* g_B.

Proof. This result is a simple consequence of the convolution theorem (Proposition 51) and the marginal preservation guarantees of the previous chapter. Theorem 68 states that, given s^{th}-order marginals of factors f and g, the Join algorithm can reconstruct the s^{th}-order marginals of the joint distribution $f \cdot g$, *exactly*. Since the riffle independent joint distribution is $m * (f \cdot g)$ and convolution operations are pointwise in the Fourier domain (Proposition 51), then given enough Fourier terms to reconstruct the s^{th}-order marginals of the function $m_{p,q}$, we can also reconstruct the s^{th}-order marginals of the riffle independent joint from the output of RiffleSplit. □

13.4.5 *Running time*

If the Fourier coefficient matrix for frequency level λ of a joint distribution is $d_\lambda \times d_\lambda$ then the running time complexity of the Join/Split algorithms of the previous chapter are at worst, cubic in the dimension, $O(d_\lambda^3)$. If the



(a) Parameter learning from synthetic data (b) Running times of RiffleJoin

Figure 35: Synthetic data experiments for a single partitioning of the item set

interleaving Fourier coefficients are precomputed ahead of time, then the complexity of RiffleJoin/RiffleSplit is also $O(d_\lambda^3)$.

If not, then we must Fourier transform the interleaving distribution. For RiffleJoin, we can Fourier transform the empirical distribution directly from the definition, or use the algorithms presented in Section 13.3.1 in the case of biased riffle shuffles, which has $O(n^2 d_\lambda^3)$ running time in the worst case when $p \sim O(n)$. For RiffleSplit, one must compute the Fourier transform of the uniform interleaving distribution, which, as we have shown in Section 13.3, also takes the form of a biased riffle shuffle and therefore also can be computed in $O(n^2 d_\lambda^3)$ time. In Section 9.4, we plot experimental running times.

13.5 EXPERIMENTS

In this section, we present experiments to validate our models and methods. All experiments were implemented in Matlab, except for the Fourier theoretic routines, which were written in C++. We tested on lab machines with two AMD quadcore Opteron 2.7GHz processors with 32 Gb memory.

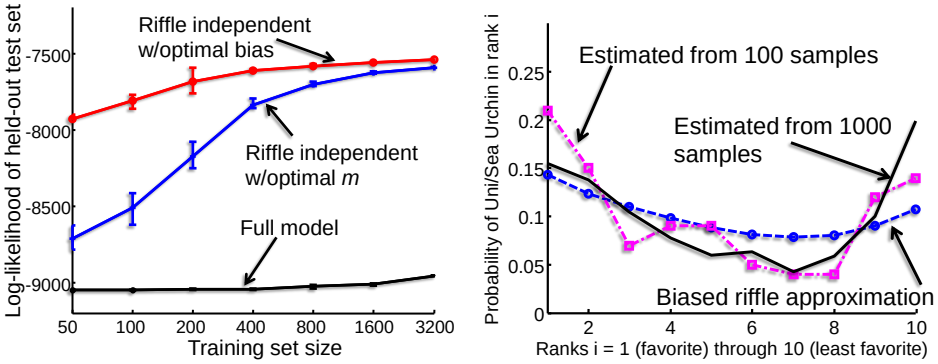
13.5.1 *Simulated data*

We begin with a discussion of our simulated data experiments. We first consider approximation quality and timing issues for a single binary partition of the item set.

To understand the behavior of RiffleSplit in approximately riffle independent situations, we drew sample sets of varying sizes from a riffle independent distribution on S_8 (with bias parameter $\alpha = .25$) and use RiffleSplit to estimate the relative ranking factors and interleaving distribution from the empirical distribution. In Figure 35a, we plot the KL-divergence between the true distribution and that obtained by applying RiffleJoin to the estimated riffle factors. With small sample sizes (far less than $8! = 40320$), we are able to recover accurate approximations despite the fact that the

1. ebi (shrimp)	2. anago (sea eel)	3. maguro (tuna)
4. ika (squid)	5. uni (sea urchin)	6. sake (salmon roe)
7. tamago (egg)	8. toro (fatty tuna)	9. tekka-maki (tuna roll)
	10. kappa-maki (cucumber roll)	

Figure 36: List of sushi types in the [70] dataset



(a) Average log-likelihood of held out test examples from the sushi dataset

(b) First-order probabilities of Uni (sea urchin) rankings

Figure 37: Sushi preference ranking experiments

empirical distributions are not exactly riffle independent. For comparison, we ran the experiment using the Split algorithm [60] to recover the parameters. Perhaps surprisingly, one can show that the Split algorithm from [60] is also an unbiased, consistent estimator of the riffle factors, but it does not return the maximum likelihood parameter estimates because it effectively ignores rankings which are not contained in the subgroup $S_p \times S_q$. Consequently, our RiffleSplit algorithm converges to the correct parameters with far fewer samples.

Next, we show that our Fourier domain algorithms are capable of handling sizeable item sets (with size n) when working with low-order terms. In Figure 35b we ran our Fourier domain RiffleJoin algorithm on various simulated distributions. We plot running times of RiffleJoin (without precomputing the interleaving distributions) as a function of n (setting $p = \lceil n/2 \rceil$, which is the worst case) scaling up to $n = 40$.

13.5.2 *Sushi preference data*

We now turn to analyzing real datasets. For our first analysis, we examine a sushi preference ranking dataset [70] consisting of 5000 full rankings of ten types of sushi. The items are enumerated in Figure 36. Note that, compared to the APA election data, the sushi dataset has twice as many items, but fewer examples.

We begin by studying our methods in the case of a single binary partitioning of the item set. Unlike the APA dataset, there is no obvious way

to naturally partition the types of sushi into two sets — in our first set of experiments, we have arbitrarily divided the item set into $A = \{1, \ldots, 5\}$ and $B = \{6, \ldots, 10\}$. In the next chapter, we consider more principled approaches for partitioning the item set.

We divided the data into training and test sets (with 500 examples) and estimated the true distribution in three ways: (1) directly from samples (with regularization), (2) using a riffle independent distribution (split evenly into two groups of five and mentioned above) with the optimal shuffling distribution m, and (3) with a biased riffle shuffle (and optimized bias α). Figure 37a plots testset log-likelihood as a function of training set size — we see that riffle independence assumptions can help significantly to lower the sample complexity of learning. Biased riffle shuffles, as can also be seen, are a useful learning bias with very small samples.

As an illustration of the behavior of biased riffle shuffles, see Figure 37b which shows the approximate first-order marginals of Uni (Sea Urchin) rankings, and the biased riffle approximation. The Uni marginals are interesting, because while many people like Uni, thus providing high rankings, many people also hate it, providing low rankings. The first-order marginal estimates have significant variance at low sample sizes, but with the biased riffle approximation, one can achieve a reasonable approximation to the distribution even with few samples at the cost of being somewhat oversmoothed.

13.6 CONCLUSIONS

In summary, we have introduced riffled independence as a novel generalization of the ordinary notion of probabilistic independence. This chapter has focused on building intuitions about the interleaving step and the development of Fourier theoretic algorithms paralleling those from Chapter 12 for joining and splitting. Finally, we provided empirical evidence on the APA dataset that riffled independence assumption can in fact approximately hold in real data. In the next chapters we will use riffled independence to analyze more datasets.

14

DISCOVERING RIFFLED INDEPENDENCE STRUCTURE IN RANKED DATA

THUS far throughout Part III, we have focused exclusively on understanding riffled independent models with a single binary partitioning of the full item set. In this section we explore a natural model simplification which comes from the simple observation that, since the relative ranking distributions f_A and g_B are again distributions over rankings, the sets A and B can further be decomposed into riffle independent subsets. We call such models *hierarchical riffle independent decompositions*. The main contributions of Chapter 14 are as follows.

- We define a simple and intuitive family of hierarchical models based on riffled independence relationships among ranked items.
- We propose an effective method for learning the hierarchical structure of such models and in particular, we propose a simple score function which evaluates the quality of a given hierarchical structure, and suggest a simple optimization algorithm for finding the best hierarchical structure for a given dataset.
- We provide a sample complexity analysis, showing that our structures can be learned with high probability in polynomial time given polynomial training examples
- Finally, on a number of datasets, we show that our methods can be applied to highlight simple latent organizational structures within the data.

14.1 HIERARCHICAL RIFFLE INDEPENDENCE DECOMPOSITIONS

For simplicity, we restrict consideration to binary hierarchies, defined as tuples of the form $H = (H_A, H_B)$, where H_A and H_B are either (1) null, in which case H is called a *leaf*, or (2) hierarchies over item sets A and B respectively. In this second case, A and B are assumed to be a nontrivial partition of the item set.

Definition 91. We say that a distribution h factors riffle independently with respect to a hierarchy $H = (H_A, H_B)$ if item sets A and B are riffle independent with respect to h, and both f_A and g_B factor riffle independently with respect to subhierarchies H_A and H_B, respectively.

Like Bayesian networks, these hierarchies represent families of distributions obeying a certain set of (riffled) independence constraints and can be parameterized locally. To draw from such a model, one generates full

rankings recursively starting by drawing rankings of the leaf sets, then working up the tree, sequentially interleaving rankings until reaching the root. The parameters of these hierarchical models are simply the interleaving and relative ranking distributions at the internal nodes and leaves of the hierarchy, respectively.

By decomposing distributions over rankings into small pieces (like Bayesian networks have done for other distributions), these hierarchical models allow for better interpretability, efficient probabilistic representation, low sample complexity, efficient MAP optimization, and, as we show in the next chapter, efficient inference.

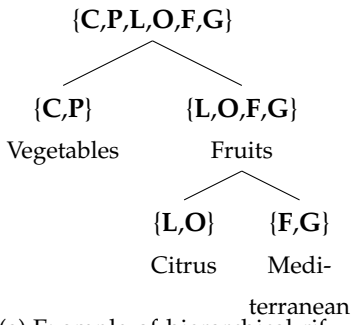
Continuing with our running example of fruits and vegetables, one can imagine that the fruits are further partitioned into two sets, a set consisting of citrus fruits ((**L**) Lemons and (**O**) Oranges) and a set consisting of mediterranean fruits ((**F**) Figs and (**G**) Grapes). To generate a full ranking, one first draws rankings of the citrus and mediterranean fruits independently ($[\![\mathbf{L}, \mathbf{O}]\!]$ and $[\![\mathbf{G}, \mathbf{F}]\!]$, for example). Secondly, the two sets are interleaved to form a ranking of all fruits ($[\![\mathbf{G}, \mathbf{L}, \mathbf{O}, \mathbf{F}]\!]$). Finally, a ranking of the vegetables is drawn ($[\![\mathbf{P}, \mathbf{C}]\!]$) and interleaved with the fruit rankings to form a full joint ranking: $[\![\mathbf{P}, \mathbf{G}, \mathbf{L}, \mathbf{O}, \mathbf{F}, \mathbf{C}]\!]$. Notationally, we can express the hierarchical decomposition as $\{\mathbf{C}, \mathbf{P}\} \perp_{m_1} (\{\mathbf{L}, \mathbf{O}\} \perp_{m_2} \{\mathbf{F}, \mathbf{G}\})$. We can also visualize hierarchies using trees (see Figure 38a for our example). The subsets of items which appear as leaves in the tree are the *leaf sets*.

A natural question to ask is: if we used a different hierarchy with the same leaf sets, would we capture the same distributions? For example, does a distribution which decomposes according to the tree in Figure 38b also decompose according to the tree in Figure 38a? The answer, in general, is no, due to the fact that distinct hierarchies impose different sets of independence assumptions, and as a result, different structures can be well or badly suited for modeling a given dataset. Consequently, it is important to use the "correct" structure if possible.

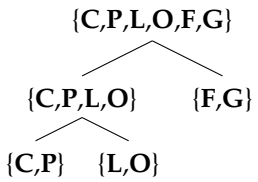
14.1.1 *Shared independence structure*

It is interesting to note, however, that while the two structures in Figures 38a and 38b encode distinct families of distributions, it is possible to identify a set of independence assumptions common to both structures. In particular since both structures have the same leaf sets, any distributions consistent with either of the two hierarchies must also be consistent with what we call a 3-*way decomposition*. We define a d-way decomposition to be a distribution with a single level of hierarchy, but instead of partitioning the entire item set into just two subsets, one partitions into d subsets, then interleaves the relative rankings of each of the d subsets together to form a joint ranking of items. Any distribution consistent with either Figure 38b or 38a must consequently also be consistent with the structure of Figure 38c. More generally, we have:

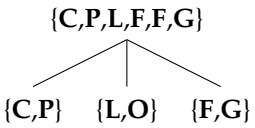
Proposition 92. *If* h *is a hierarchical riffle independent model with* d *leaf sets, then* h *can also be written as a* d*-way decomposition.*



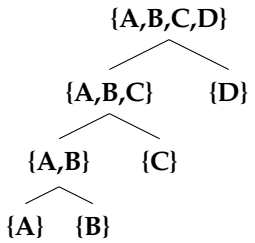
(a) Example of hierarchical riffled independence structure on S_6



(b) Another example, not equivalent to (a)



(c) 3-way decomposition for S_6 (generalizes the class of distributions parameterized by (a), (b)



(d) Hierarchical decomposition into singleton subset, where each leaf set consists of a single item (we will also refer to this particular type of tree as a 1-thin chain)

Figure 38: Examples of distinct hierarchical riffle independent structures.

Proof. We proceed by induction. Suppose the result holds for $S_{n'}$ for all $n' < n$. We want to establish that the result also holds for S_n. If h factors according to a hierarchical riffle independent model, then it can be written as $h = m \cdot f_A \cdot g_B$, where m is the interleaving distribution, and f_A, g_B themselves factor as hierarchical riffle independent distributions with, say, d_1 and d_2 leaf sets, respectively (where $d_1 + d_2 = d$). By the hypothesis, since $|A|, |B| < n$, we can factor both f_A and g_B as d_1 and d_2-way decompositions respectively. We can therefore write f_A and g_B as:

$$f_A(\pi_A) = m_A(\tau_{A_1,\ldots,A_{d_1}}) \cdot \prod_{i=1}^{d_1} f_{A_i}(\phi_{A_i}(\pi_A)),$$

$$g_B(\pi_B) = m_B(\tau_{B_1,\ldots,B_{d_2}}) \cdot \prod_{i=1}^{d_2} g_{B_i}(\phi_{B_i}(\pi_B)).$$

Substituting these decompositions into the factorization of the distribution h, we have:

$$\begin{aligned} h(\sigma) &= m(\tau_{A,B}(\sigma)) f_A(\phi_A(\sigma)) g_B(\phi_B(\sigma)), \\ &= \left(m(\tau_{A,B}(\sigma)) m_A(\tau_{A_1\ldots,A_{d_1}}) m_B(\tau_{B_1,\ldots,B_{d_2}}) \right) \end{aligned}$$

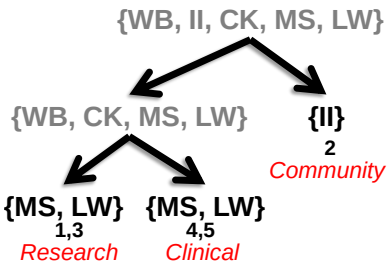


Figure 39: Hierarchical structure learned from APA data.

$$\cdot \prod_{i=1}^{d_1} f_{A_i}(\phi_{A_i}(\phi_A(\sigma))) \prod_{i=1}^{d_2} g_{B_i}(\phi_{B_i}(\phi_B(\sigma))),$$
$$= \tilde{m}(\tau_{A_1,\ldots,A_{d_1},B_1\ldots,B_{d_2}}) \cdot \prod_{i=1}^{d_1} f_{A_i}(\phi_{A_i}(\sigma)) \prod_{i=1}^{d_2} g_{B_i}(\phi_{B_i}(\sigma)),$$

where the last line follows because any legitimate interleaving of the sets A and B is also a legitimate interleaving of the sets $A_1, \ldots, A_{d_1}, B_1, \ldots, B_{d_2}$ and since $\phi_{A_i}(\phi_A(\sigma)) = \phi_{A_i}(\sigma)$. This shows that the distribution h factors as a $d_1 + d_2$-way decomposition, and concludes the proof. □

In general, knowing the hierarchical decomposition of a model is more desirable than knowing its d-way decomposition which may require many more parameters $\left(O(\frac{n!}{\prod_i d_i!}), \text{where } i \text{ indexes over leaf sets}\right)$. For example, the extreme case of the n-way decomposition requires $O(n!)$ parameters and captures every distribution over permutations.

14.1.2 *Thin chain models*

There is a class of particularly simple hierarchical models which we will refer to as k-thin chain models. By a k-thin chain model, we refer to a hierarchical structure in which the size of the smaller set at each split in the hierarchy is fixed to be a constant and can therefore be expressed as:

$$(A_1 \perp_m (A_2 \perp_m (A_3 \perp_m \ldots))), \; |A_i| = k, \text{for all } i.$$

See Figure 38d for an example of 1-thin chain. We view thin chains as being somewhat analogous to thin junction tree models [7], in which cliques are never allowed to have more than k variables. When $k \sim O(1)$, for example, the number of model parameters scales polynomially in n. To draw rankings from a thin chain model, one sequentially inserts items independently, one group of size k at a time, into the full ranking.

Theorem 93. *The k^{th} order marginals are sufficient statistics for a k-thin chain model.*

Proof. Corollary of Theorem 90 □

Example 94 (APA election data (continued)). *The APA, as described by [30], is divided into "*academicians and clinicians who are on uneasy terms*". In 1980, candidates* $\{1,3\}$ *(W. Bevan and C. Kiesler who were research psychologists) and* $\{4,5\}$ *(M.Siegle and L. Wright, who were clinical psychologists) fell on opposite ends of this political spectrum with candidate 2 (I. Iscoe) being somewhat independent. Diaconis conjectured that voters choose one group over the other, and then choose within. We are now able to verify Diaconis' conjecture using our riffled independence framework. After removing candidate 2 from the distribution, we perform a search within candidates* $\{1,3,4,5\}$ *to again find* nearly *riffle independent subsets. We find that* $A = \{1,3\}$ *and* $B = \{4,5\}$ *are very nearly riffle independent (with respect to KL divergence) and thus are able to verify that candidate sets* $\{2\}$, $\{1,3\}$, $\{4,5\}$ *are indeed grouped in a riffle independent sense in the APA data. We remark that in a later work, [92] identified candidate 2 (I. Iscoe) as belonging to yet a third group of psychologists called* community psychologists*. The hierarchical structure that best describes the APA data is shown in Figure 39 and the KL-divergence from the true distribution to the hierarchical model is* $d_{KL} = .0676$.

Finally for the two main opposing groups within the APA, the riffle shuffling distribution for sets $\{1,3\}$ *and* $\{4,5\}$ *is not well approximated by a biased riffle shuffle. Instead, since there are two coalitions, we fit a mixture of two biased riffle shuffles to the data and found the bias parameters of the mixture components to be* $\alpha_1 \approx .67$ *and* $\alpha_2 \approx .17$*, indicating that the two components oppose each other (since* α_1 *and* α_2 *lie on either side of* $.5$*).*

14.1.3 *Mallows models*

We now review the well known *Mallows models* which can be seen as a subset of the 1-thin chain model family. We show in particular that, under a Mallows distribution, items are riffle independent with respect to a particular chain hierarchy. Exploiting this riffle independent structure yields an efficient method for conditioning on arbitrary partial rankings.

Let $d_K : S_n \times S_n \to \mathbb{R}$ be the *Kendall's tau* distance metric on rankings (see Section 2.3.2 for a definition). Given two rankings σ_1, σ_2, $d_\tau(\sigma_1, \sigma_2)$ is defined as the minimum number of adjacent transpositions necessary to convert one argument σ_1 into the other, σ_2. The *Mallows model* is a distribution over rankings defined as:

$$h^{Mallows}(\sigma; \phi, \sigma_0) \propto \phi^{-d_\tau(\sigma,\sigma_0)}, \tag{14.1}$$

where σ_0 represents a *central or reference* ranking and ϕ is a spread parameter. For simplicity, we will assume that σ_0 is the identity ranking mapping item 1 to rank 1, item 2 to rank 2, and so on.

For a given ranking σ and each item j of the item set, define:

$$V_j(\sigma) = \#\{i \,:\, j+1 \leqslant i \leqslant n,\ \sigma(i) < \sigma(j)\},$$

which is simply the number of items in the itemset $\{j+1, \ldots, n\}$ which are ranked before item j with respect to σ. The collection of V_js fully determines the ranking σ, and the procedure given in Algorithm 14.1 can be used to

Algorithm 14.1: Reconstruct σ from the collection of V_js. Note that V_n is always zero and hence is not used in the algorithm. Input: $V_1, \ldots, V_{n-1}$. Output: A permutation $\sigma \in S_n$ corresponding to the tuple $(V_1, \ldots, V_{n-1})$.

RECONSTRUCTSIGMA$(V_1, \ldots, V_{n-1})$

Initialize σ to be a ranking of $\{n\}$, mapping n to 1 ;
for $j = n-1, n-2, \ldots, 1$ **do**
    Insert item j in rank $V_j + 1$;
end
return (σ);

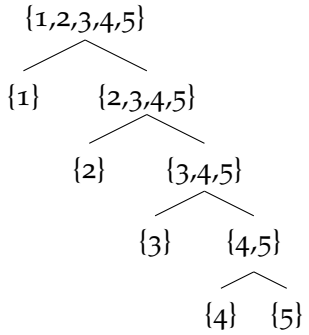


Figure 40: An example of a hierarchical structure over five food items

reconstruct σ ([97]). Fligner & Verducci first showed [36] (see also [97]) that a ranking can be sampled from the Mallows model by drawing the V_j independently, each according to a particular exponentially parameterized distribution. In particular, set each V_j to be a value r drawn from the set $\{0, \ldots, n-j\}$ with probability proportional to ϕ^r (where, again, ϕ is the Mallows spread parameter). Using Algorithm 14.1 to reconstruct σ from the drawn values of V_j yields an independent draw from a Mallows model with spread parameter ϕ.

This generative procedure of drawing the V_j independently is exactly the same as that of a riffle independent hierarchy in which a single item is partitioned out of the hierarchy at each level of the hierarchy, with exponentially parameterized interleaving distributions. For example, on $n = 5$ items, the hierarchy encoding the factorization of the Mallows model is given in Figure 40 with item 1 being partitioned out at the topmost level, then item 2 partitioned out at the second layer, and so on. Since each leaf node consists of a single item, there are no relative ranking parameters. At each internal node of the hierarchy, the interleaving distribution which determines the position where item j is inserted into the item subset $\{j+1, \ldots, n\}$ is given by $m(\tau_{AB} = B|B|\ldots|A|\ldots|B|B) \propto \phi^r$, where r is the position of the A item in τ.

As a side note, we remark that these interleaving distributions are very similar to (but not exactly the same as) the biased riffle shuffles introduced in the previous chapter, where the interleaving step is likened to the *riffle shuffle* for cards, in which one drops cards one by one, selected from the left or right deck after each drop with some probability.

14.2 OBJECTIVE FUNCTIONS FOR STRUCTURE LEARNING

Since different hierarchies impose different independence assumptions, we would like to find the structure that is best suited for modeling a given ranking dataset. On some datasets, a natural hierarchy might be available — for example, if one were familiar with the typical politics of APA elections, then it may have been possible to "guess" the optimal hierarchy. However, for general ranked data, it is not always obvious what kind of groupings riffled independence will lead to, particularly for large n. Should fruits really be riffle independent of vegetables? Or are green foods riffle independent of red foods?

Over the next three sections, we address the problem of automatically discovering hierarchical riffle independent structures from training data. Key among our observations is the fact that while absolute item ranks cannot be independent due to mutual exclusivity, relative ranks between sets of items are not subject to the same constraints. More than simply being a 'clustering' algorithm, however, our procedure can be thought of as a structure learning algorithm, like those from the graphical models literature [74], which find the optimal (riffled) independence decomposition of a distribution.

The base problem that we address in this current section is how to find the best structure if there is only one level of partitioning and two leaf sets, A, B. Alternatively, we want to find the topmost partitioning of the tree. In Section 14.3, we use this base case as part of a top-down approach for learning a full hierarchy.

14.2.1 *Problem statement*

Given then, a training set of rankings, $\sigma^{(1)}$, $\sigma^{(2)}$, $\ldots, \sigma^{(m)} \sim h$, drawn i.i.d. from a distribution in which a subset of items, $A \subset \{1, \ldots, n\}$, is riffle independent of its complement, B, the problem which we address in this section is that of automatically determining the sets A and B. If h does not *exactly* factor riffle independently, then we would like to find the riffle independent approximation which is *closest* to h in some sense. Formally, we would like to solve the problem:

$$\arg\min_{A} \min_{m,f,g} \; D_{KL}(\hat{h}(\sigma) \,\|\, m(\tau_{A,B}(\sigma)) f(\phi_A(\sigma)) g(\phi_B(\sigma))), \tag{14.2}$$

where $\hat{h}$ is the empirical distribution of training examples and D_{KL} is the Kullback-Leibler divergence measure. Equation 14.2 is a seemingly reasonable objective since it can also be interpreted as maximizing the likelihood of the training data. In the limit of infinite data, Equation 14.2 can be shown via the Gibbs inequality to attain its minimum, zero, at the subsets A and B, if and only if the sets A and B are truly riffle independent of each other.

For small problems, one can actually solve Problem 14.2 using a single computer by evaluating the approximation quality of each subset A and taking the minimum, which was the approach taken in Example 94. However,

for larger problems, one runs into time and sample complexity problems since optimizing the globally defined objective function (Equation 14.2) requires relearning all model parameters (m, f_A, and g_B) for each of the exponentially many subsets of $\{1, \ldots, n\}$. In fact, for large sets A and B, it is rare that one would have enough samples to estimate the relative ranking parameters f_A and g_B without already having discovered the hierarchical riffle independent decompositions of A and B. We next propose a more locally defined objective function, reminiscent of clustering, which we will use instead of Equation 14.2. As we show, our new objective will be more tractable to compute and have lower sample complexity for estimation.

14.2.2 *Proposed objective function*

The approach we take is to minimize a different measure that exploits the observation that *absolute ranks of items in A are fully independent of relative ranks of items in B, and vice versa* (which we prove in Proposition 95). With our vegetables and fruits, for example, knowing that Figs is ranked first among all six items (the absolute rank of a fruit) should give no information about whether Corn is preferred to Peas (the relative rank of vegetables). More formally, given a subset $A = \{a_1, \ldots, a_\ell\}$, recall that $\sigma(A)$ denotes the vector of (absolute) ranks assigned to items in A by σ (thus, $\sigma(A) = (\sigma(a_1), \sigma(a_2), \ldots, \sigma(a_\ell))$). We propose to minimize an alternative objective function:

$$\mathcal{F}(A) \equiv I(\sigma(A)\ ;\ \phi_B(\sigma)) + I(\sigma(B)\ ;\ \phi_A(\sigma)), \tag{14.3}$$

where I denotes the mutual information (defined between two variables X_1 and X_2 by $I(X_1; X_2) \equiv D_{KL}(P(X_1, X_2) \| P(X_1)P(X_2))$.

The function $\mathcal{F}$ does not have the same likelihood interpretation as the objective function of Equation 14.2. However, it can be thought of as a composite likelihood of two models, one in which the relative rankings of A are independent of absolute rankings of B, and one in which the relative rankings of B are independent of absolute rankings of A (see Appendix E.2). With respect to distributions which satisfy (or approximately satisfy) both models (i.e., the riffle independent distributions), minimizing $\mathcal{F}$ *is* equivalent to (or approximately equivalent to) maximizing the log likelihood of the data. Furthermore, we can show that $\mathcal{F}$ is guaranteed to detect riffled independence:

Proposition 95. *$\mathcal{F}(A) = 0$ is a necessary and sufficient criterion for a subset $A \subset \{1, \ldots, n\}$ to be riffle independent of its complement, B.*

Proof. Suppose A and B are riffle independent. We first claim that $\sigma(A)$ and $\phi_B(\sigma)$ are independent. To see this, observe that the absolute ranks of A, $\sigma(A)$, are determined by the relative rankings of A, $\phi_A(\sigma)$ and the interleaving $\tau_{A,B}(\sigma)$. By the assumption that A and B are riffle independent, we know that the relative rankings of A and B ($\phi_A(\sigma)$ and $\phi_B(\sigma)$), and the interleaving $\tau_{A,B}(\sigma)$ are independent, establishing the claim. The argument

that $\sigma(B)$ and $\phi_A(\sigma)$ are independent is similar, thus establishing one direction of the proposition.

To establish the reverse direction, assume that Equation 14.3 evaluates to zero on sets A and B. It follows that $\sigma(A) \perp \phi_B(\sigma)$ and $\phi_A(\sigma) \perp \sigma(B)$. Now, as a converse to the observation from above, note that the absolute ranks of A *determine* the relative ranks of A, $\phi_A(\sigma)$, as well as the interleaving $\tau_{A,B}(\sigma)$. Similarly, $\sigma(B)$ determines $\phi_B(\sigma)$ and $\tau_{A,B}(\sigma)$. Thus, $(\phi_A(\sigma), \tau_{A,B}(\sigma)) \perp \phi_B(\sigma)$ and $\phi_A(\sigma) \perp (\tau_{A,B}(\sigma), \phi_B(\sigma))$. It then follows that $\phi_A(\sigma) \perp \tau_{A,B}(\sigma) \perp \phi_B(\sigma)$. □

As with Equation 14.2, optimizing $\mathcal{F}$ is still intractable for large n. However, $\mathcal{F}$ motivates a natural proxy, in which we replace the mutual informations defined over all n variables by a sum of mutual informations defined over just three variables at a time.

Definition 96 (Tripletwise mutual informations)**.** Given any triplet of distinct items, (i, j, k), we define the tripletwise mutual information term, $I_{i;j,k} \equiv I(\sigma(i) ; \sigma(j) < \sigma(k))$.

The tripletwise mutual information $I_{i;j,k}$ can be computed as follows:

$$I(\sigma(i) ; \sigma(j) < \sigma(k)) = \sum_{\sigma(i)} \sum_{\sigma(j)<\sigma(k)} h(\sigma(i), \sigma(j) < \sigma(k)) \log \frac{h(\sigma(i), \sigma(j) < \sigma(k))}{h(\sigma(i))h(\sigma(j) < \sigma(k))},$$

where the inside summation runs over two values, true/false, for the binary variable $\sigma(j) < \sigma(k)$. To evaluate how riffle independent two subsets A and B are, we want to examine the triplets that straddle the two sets.

Definition 97 (Internal and Cross triplets)**.** We define $\Omega^{cross}_{A,B}$ to be the set of triplets which "cross" from set A to set B: $\Omega^{cross}_{A,B} \equiv \{(i; j, k) : i \in A, j, k \in B\}$. $\Omega^{cross}_{B,A}$ is similarly defined. We also define Ω^{int}_{A} to be the set of triplets that are internal to A: $\Omega^{int}_{A} \equiv \{(i; j, k) : i, j, k \in A\}$, and again, Ω^{int}_{B} is similarly defined.

Our proxy objective function can be written as the sum of the mutual information evaluated over all of the crossing triplets:

$$\tilde{\mathcal{F}}(A) \equiv \sum_{(i,j,k)\in\Omega^{cross}_{A,B}} I_{i;j,k} + \sum_{(i,j,k)\in\Omega^{cross}_{B,A}} I_{i;j,k}. \tag{14.4}$$

$\tilde{\mathcal{F}}$ can be viewed as a low order version of $\mathcal{F}$, involving mutual information computations over triplets of variables at a time instead of n-tuples. The mutual information $I_{i;j,k}$, for example, reflects how much the rank of a vegetable (i) tells us about how two fruits (j, k) compare. If A and B are riffle independent, then we know that $I_{i;j,k} = 0$ for any (i, j, k) such that $i \in A$, $j, k \in B$ (and similarly for any (i, j, k)) such that $i \in B$, $j, k \in A$. Given that fruits and vegetables are riffle independent sets, knowing that Grapes is preferred to Figs should give no information about the absolute rank of Corn, and therefore $I_{Corn;Grapes,Figs}$ should be zero. Note that such tripletwise independence assertions bear resemblance to assumptions

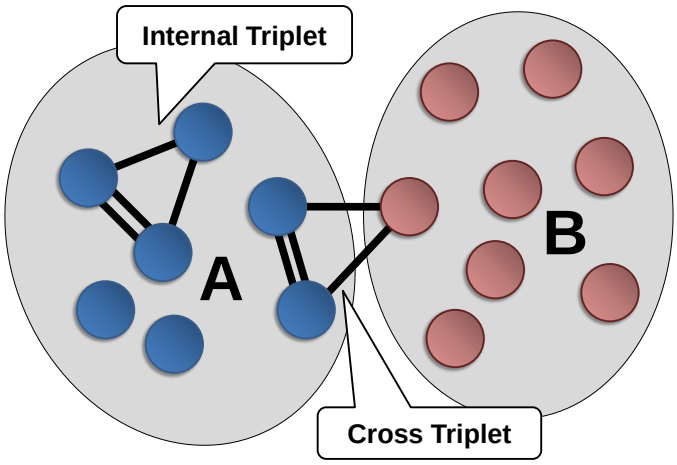


Figure 41: This diagram shows a graphical depiction of the problem of finding riffle independent subsets. A triangle with vertices (i, j, k) represents the term $I_{i;j,k}$ Since the $I_{i;j,k}$ are not invariant with respect to a permutation of the indices i, j, and k, the triangles are directed, and we therefore use double bars represent the nodes j, k for the term $I_{i;j,k}$. Note that if the tripletwise terms were instead replaced by edgewise terms, the problem would simply be a standard clustering problem.

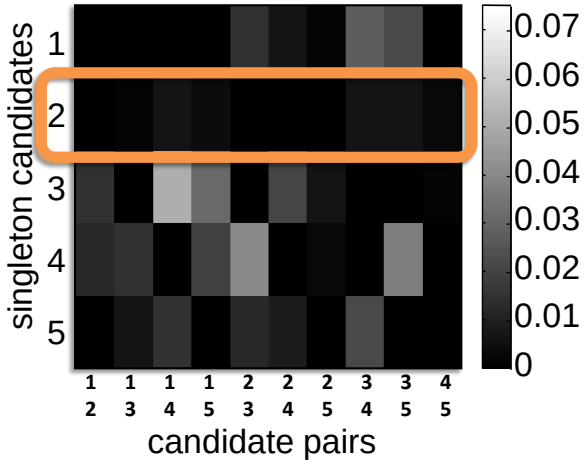


Figure 42: Here we show the matrix of tripletwise mutual informations computed from the APA dataset (see Example 98).

sometimes made in social choice theory, commonly referred to as *Independence of Irrelevant Alternatives* [6], where the addition of a third element i, is assumed to not affect whether one prefers an element j over k.

The objective $\tilde{\mathcal{F}}$ is somewhat reminiscent of typical graphcut and clustering objectives. Instead of partitioning a set of nodes based on sums of pairwise similarities, we partition based on sums of tripletwise affinities. We show a graphical depiction of the problem in Figure 41, where cross triplets (in $\Omega^{cross}_{A,B}$, $\Omega^{cross}_{B,A}$) have low weight and internal triplets (in Ω^{int}_{A}, Ω^{int}_{B}) have high weight. The objective is to find a partition such that the sum over cross triplets is low. In fact, the problem of optimizing $\tilde{\mathcal{F}}$ can be

seen as an instance of the weighted, directed hypergraph cut problem [42]. Note that the word *directed* is significant for us, because, unlike typical clustering problems, our triplets are not symmetric (for example, $I_{i;jk} \neq I_{j;ik}$), resulting in a nonstandard and poorly understood optimization problem.

Example 98 (APA election data (continued))**.** *Figure 41 visualizes the tripletwise mutual informations computed from the APA dataset. Since there are five candidates, there are* $\binom{5}{2} = 10$ *pairs of candidates. The* $(i,(j,k))$ *entry in the matrix corresponds to* $I(\sigma(i);\sigma(j) < \sigma(k))$*. For easier visualization, we have set entries of the form* $(i,(i,k))$ *and* $(i,(j,i))$ *to be zero since they are not counted in the objective function.*

The highlighted row corresponds to candidate 2, in which all of the mutual information terms are close to zero. We see that the tripletwise mutual information terms tell a story consistent with the conclusion of Example 86, in which we showed that candidate 2 was approximately riffle independent of the remaining candidates.

Finally, it is also interesting to examine the $(3,(1,4))$ *entry. It is the largest mutual information in the matrix, a fact which should not be surprising since candidates 1 and 3 are politically aligned (both research psychologists). Thus, knowing, for example, that candidate 3 was ranked first is a strong indication that candidate 1 was preferred over candidate 4.*

14.2.3 *Encouraging balanced partitions*

In practice, like the minimum cut objective for graphs, the tripletwise objective of Equation 14.4 has a tendency to "prefer" small partitions (either $|A|$ or $|B|$ very small) to more balanced partitions ($|A|, |B| \approx n/2$) due to the fact that unbalanced partitions have fewer triplets that cross between A and B. The simplest way to avoid this bias is to optimize the objective function over subsets of a fixed size k. As we discuss in the next section, optimizing with a fixed k can be useful for building thin hierarchical riffle independent models. Alternatively, one can use a modified objective function that encourages more balanced partitions. For example, we have found the following *normalized cut* [119] inspired variation of our objective to be useful for detecting riffled independence when the size k is unknown:

$$\mathcal{F}^{balanced}(A) \equiv \frac{\sum_{\Omega^{cross}_{A,B}} I_{i;j,k}}{\sum_{\Omega^{cross}_{A,B}} I_{i;j,k} + \sum_{\Omega^{int}_{A}} I_{i;j,k}} + \frac{\sum_{\Omega^{cross}_{B,A}} I_{i;j,k}}{\sum_{\Omega^{cross}_{B,A}} I_{i;j,k} + \sum_{\Omega^{int}_{B}} I_{i;j,k}}. \tag{14.5}$$

Intuitively, the denominator in Equation 14.5 penalizes subsets whose interiors have small weight. Note that there exist many variations on the objective function that encourage balance, but $\mathcal{F}^{balanced}$ is the one that we have used in our experiments.

14.2.4 *Low-order detectability assumptions*

When does $\tilde{\mathcal{F}}$ detect riffled independence? It is not difficult to see, for example, that $\tilde{\mathcal{F}} = 0$ is a necessary condition for riffled independence, since $A \perp_m B$ implies $I_{a;b,b'} = 0$. We have:

Proposition 99. *If* A *and* B *are riffle independent sets, then* $\tilde{\mathcal{F}}(A) = 0$.

However, the converse of Proposition 99 is not true in full generality without accounting for dependencies that involve larger subsets of variables. Just as the pairwise independence assumptions that are commonly used for randomized algorithms [100][1] do not imply full independence between two sets of variables, there exist distributions which "look" riffle independent from tripletwise marginals but do not factor upon examining higher-order terms. Nonetheless, in most practical scenarios, we expect $\tilde{\mathcal{F}} = 0$ to imply riffled independence.

14.2.5 *Quadrupletwise objective functions for riffled independence*

A natural variation of our method is to base the objective function on the following quantities, defined over quadruplets of items instead of triplets:

$$I_{ij;kl} \equiv I(\sigma(i) < \sigma(j)\,;\, \sigma(k) < \sigma(\ell)). \tag{14.6}$$

Intuitively, $I_{ij;kl}$ measures how much knowing that, say, Peas is preferred to Corn, tells us about whether Grapes are preferred to Oranges. Again, if the fruits and vegetables are riffle independent, then the mutual information should be zero. Summing over terms which cross between the cut, we obtain a quadrupletwise objective function defined as: $\mathcal{F}^{quad}(A) \equiv \sum_{(i,j)\in A,(k,\ell)\in B} I_{ij;kl}$. If A and B are riffle independent with $i, j \in A$ and $k, \ell \in B$, then the mutual information $I_{ij;kl}$ is zero. Unlike their tripletwise counterparts, however, the $I_{ij,kl}$ do not arise from a global measure that is both necessary and sufficient for detecting riffled independence. In particular, $I(\phi_A(\sigma); \phi_B(\sigma)) = 0$ is insufficient to guarantee riffled independence. For example, if the interleaving depends on the relative rankings of A and B, then riffled independence is not satisfied, yet $\mathcal{F}^{quad}(A) = 0$. Moreover, it is not clear how one would detect riffle independent subsets consisting of a single element using a quadrupletwise measure. As such, we have focused on tripletwise measures in our experiments. Nonetheless, quadrupletwise measures may potentially be useful in practice (for detecting larger subsets) and have the significant advantage that the $I_{ij;kl}$ can be estimated with fewer samples and using almost any imaginable form of partially ranked data.

1 A pairwise independent family of random variables is one in which any two members are marginally independent. Subsets with larger than two members may not necessarily factor independently, however.

14.2.6 *Estimating the objective from samples*

We have so far argued that $\tilde{\mathcal{F}}$ is a reasonable function for finding riffle independent subsets. However, since we only have access to samples rather than the true distribution h itself, it will only be possible to compute an approximation to the objective $\tilde{\mathcal{F}}$. In particular, for every triplet of items, (i, j, k), we must compute an estimate of the mutual information $I_{i;j,k}$ from i.i.d. samples drawn from h, and the main question is: how many samples will we need in order for the approximate version of $\tilde{\mathcal{F}}$ to remain a reasonable objective function?

In the following, we denote the estimated value of $I_{i;j,k}$ by $\hat{I}_{i;j,k}$. For each triplet, we use a regularized procedure due to [51] to estimate mutual information. We adapt his sample complexity bound to our problem below.

Lemma 100. *For any fixed triplet (i, j, k), the mutual information $I_{i;j,k}$ can be estimated to within an accuracy of Δ with probability at least $1 - \gamma$ using $S(\Delta, \gamma) \equiv O\left(\frac{n^2}{\Delta^2} \log^2 \frac{n}{\Delta} \log \frac{n}{\gamma}\right)$ i.i.d. samples and the same amount of time.*

The approximate objective function is therefore:

$$\hat{\mathcal{F}}(A) \equiv \sum_{(i,j,k) \in \Omega^{\text{cross}}_{A,B}} \hat{I}_{i;j,k} + \sum_{(i,j,k) \in \Omega^{\text{cross}}_{B,A}} \hat{I}_{i;j,k}.$$

What we want to now show is that, if there exists a unique way to partition $\{1, \ldots, n\}$ into riffle independent sets, then given enough training examples, our approximation $\hat{\mathcal{F}}$ uniquely singles out the correct partition as its minimum with high probability. A class of riffle independent distributions for which the uniqueness requirement is satisfied consists of the distributions for which A and B are *strongly connected* according to the following definition.

Definition 101. A subset $A \subset \{1, \ldots, n\}$ is called *ϵ-third-order strongly connected* if, for every triplet $i, j, k \in A$ with i, j, k distinct, we have $I_{i;j,k} > \epsilon$.

If a set A is riffle independent of B and both sets are third order strongly connected, then we can ensure that riffled independence is detectable from third-order terms and that the partition is unique. We have the following probabilistic guarantee.

Theorem 102. *Let A and B be ϵ-third order strongly connected riffle independent sets, and suppose $|A| = k$. Given $S(\Delta, \epsilon) \equiv O\left(\frac{n^4}{\epsilon^2} \log^2 \frac{n}{\epsilon} \log \frac{n}{\gamma}\right)$ i.i.d. samples, the minimum of $\hat{\mathcal{F}}$ is achieved at exactly the subsets A and B with probability at least $1 - \gamma$.*

See Appendix E for details. Finally, we remark that the strong connectivity assumptions used in Theorem 102 are stronger than necessary — and with respect to certain interleaving distributions, it can even be the case that the estimated objective function singles out the correct partition when all of internal triplets belonging to A and B have zero mutual information. Moreover, in some cases, there are multiple valid partitionings of the item

set. For example the uniform distribution is a distribution in which every subset $A \subset \{1, \ldots, n\}$ is riffle independent of its complement. In such cases, multiple solutions are equally good when evaluated under $\tilde{\mathcal{F}}$, but not its sample approximation, $\hat{\mathcal{F}}$.

14.3 ALGORITHMS FOR STRUCTURE DISCOVERY

Having now designed a function that is tractable to estimate from both perspectives of computational and sample complexity, we turn to the problem of learning the hierarchical riffle independence structure of a distribution from training examples. Instead of directly optimizing an objective in the space of possible hierarchies, we take a simple top-down approach in which the item sets are recursively partitioned by optimizing $\hat{\mathcal{F}}$ until some stopping criterion is met (for example, when the leaf sets are smaller than some k, or simply stopping after a fixed number of splits).

EXHAUSTIVE OPTIMIZATION. Optimizing the function $\hat{\mathcal{F}}$ requires searching through the collection of subsets of size $|A| = k$, which, when performed exhaustively, requires $O\left(\binom{n}{k}\right)$ time. An exhaustive approach thus runs in exponential time, for example, when $k \sim O(n)$.

However, when the size of k is known and small ($k \sim O(1)$), the optimal partitioning of an item set can be found in polynomial time by exhaustively evaluating $\hat{\mathcal{F}}$ over all k-subsets.

Corollary 103. *Under the conditions of Theorem 102, one needs at most* $S(\Delta, \epsilon) \equiv O\left(\frac{n^2}{\epsilon^2} \log^2 \frac{n}{\epsilon} \log \frac{n}{\gamma}\right)$ *samples to recover the exact riffle independent partitioning with probability* $1 - \gamma$.

When k is small, we can therefore use exhaustive optimization to learn the structure of k-thin chain models (Section 14.1.2) in polynomial time. The structure learning problem for thin chains is to discover how the items are partitioned into groups, which group is inserted first, which group is inserted second, and so on. To learn the structure of a thin chain, we can use exhaustive optimization to learn the topmost partitioning of the item set, then recursively learn a thin chain model for the items in the larger subset.

14.3.1 *Handling arbitrary partitions using* ANCHORS

When k is large, or even unknown, $\hat{\mathcal{F}}$ cannot be optimized using exhaustive methods. Instead, we propose a simple algorithm for finding A and B based on the following observation. If an oracle could identify any two elements of the set A, say, a_1, a_2, in advance, then the quantity $I_{x;a_1,a_2} = I(x; a_1 < a_2)$ indicates whether the item x belongs to A or B since $I_{x;a_1,a_2}$ is nonzero in the first case, and zero in the second case.

For finite training sets, when I is only known approximately, one can sort the set $\{I_{x;a_1,a_2} ; x \neq a_1, a_2\}$ and if k is known, take the k items closest

Algorithm 14.2: Pseudocode for partitioning using the *Anchors* method. Input: a training set of rankings $\{\sigma^{(1)}, \ldots, \sigma^{(m)}\}$, and $k \equiv |A|$ (the size of A). Output:. a riffle independent partitioning of item set, (A_{best}, B_{best})

ANCHORSPARTITION($\{\sigma^{(1)}, \ldots, \sigma^{(m)}\}$, $k \equiv |A|$):

Fix a_1 to be any item ;
forall $a_2 \in \{1, \ldots, n\}$, $a_1 \neq a_2$ **do**
  Estimate $\hat{I}_{x;a_1,a_2}$ for all $x \neq a_1, a_2$;
  $\hat{I}^k \leftarrow k^{th}$ smallest item in $\{\hat{I}_{x;a_1,a_2}; x \neq a_1, a_2\}$;
  $A_{a_1,a_2} \leftarrow \{x : \hat{I}_{x;a_1,a_2} \leqslant \hat{I}^k\}$;
end
$A_{best} \leftarrow \arg\min_{a_1,a_2} \hat{\mathcal{F}}(A_{a_1,a_2})$;
$B_{best} \leftarrow \{1, \ldots, n\} \backslash A_{best}$;
return $([A_{best}, B_{best}])$;

to zero to be the set B (when k is unknown, one can use a threshold to infer k). Since we compare all items against a_1, a_2, we refer to these two fixed items as "anchors".

Of course a_1, a_2 are not known in advance, but by fixing a_1 to be an arbitrary item, one can repeat the above method for all $n-1$ settings of a_2 to produce a collection of $O(n^2)$ candidate partitions. Each partition can then be scored using the approximate objective $\hat{\mathcal{F}}$, and a final optimal partition can be selected as the minimum over the candidates. See Algorithm 14.2. In cases when k is not known a priori, we evaluate partitions for all possible settings of k using $\hat{\mathcal{F}}$.

Since the Anchors method does not require searching over subsets, it can be significantly faster than an exhaustive optimization of $\hat{\mathcal{F}}$. Moreover, by assuming ϵ-third order strong connectivity as in the previous section, one can use similar arguments to derive sample complexity bounds.

Corollary 104 (of Theorem 102)**.** *Let A and B be ϵ-third order strongly connected riffle independent sets, and suppose $|A| = k$. Given $S(\Delta, \epsilon)$ i.i.d. samples, the output of the Anchors algorithm is exactly $[A, B]$ with probability $1 - \gamma$. In particular, the Anchors estimator is consistent.*

We remark, however, that there are practical differences that can at times make the Anchors method somewhat less robust than an exhaustive search. Conceptually, anchoring works well when there exists two elements that are strongly connected with all of the other elements in its set, which can then be used as the anchor elements a_1, a_2. An exhaustive search can work well in weaker conditions such as when items are strongly connected through longer paths. We show in our experiments that the Anchors method can nonetheless be quite effective for learning hierarchies.

14.3.2 *Running time.*

We now consider the running time of our structure learning procedures. In both cases, it is necessary to precompute the mutual information quantities $I_{i;j,k}$ for all triplets i, j, k from m samples. For each triplet, we can compute

$I_{i;j,k}$ in linear time with respect to the sample size. The set of all triplets can therefore be computed in $O(mn^3)$ time.

The exhaustive method for finding the k-subset which minimizes $\hat{\mathcal{F}}$ requires evaluating the objective function at $\binom{n}{k} = O(n^k)$ subsets. What is the complexity of evaluating $\hat{\mathcal{F}}$ at a particular partition A, B? We need to sum the precomputed mutual informations over the number of triangles that cross between A and B. If $|A| = k$ and $|B| = n - k$, then we can bound the number of such triangles by $k(n-k)^2 + k^2(n-k) = O(kn^2)$. Thus, we require $O(n^k + kn^2)$ optimization time, leading to a bound of $O(kn^{k+2} + mn^3)$ total time.

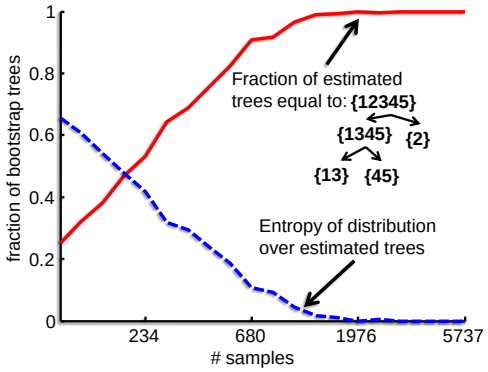
The Anchors method requires us to (again) precompute mutual informations. The other seeming bottleneck is the last step, in which we must evaluate the objective function $\hat{\mathcal{F}}$ at $O(n^2)$ partitions. In reality, if $|A|$ and $|B|$ are both larger than 1, then a_1 can be held fixed at any arbitrary element, and we must only optimize over $O(n)$ partitions. When $|A| = |B| = 1$, then $n = 2$, in which case the two sets are trivially riffle independent (independent of the actual distribution). As we showed in the previous paragraph, evaluating $\hat{\mathcal{F}}$ requires $O(kn^2)$ time, and thus optimization using the Anchors method = $O(n^3(k+m))$ total time. Since k is much smaller than m (in any meaningful training set), we can drop it from the big-O notation to get $O(mn^3)$ time complexity, showing that the Anchors method is dominated by the time that is required to precompute and cache mutual informations.

14.4 QUANTIFYING STABILITY OF STRUCTURE RECOVERY

Given a hierarchy estimated from data, we now discuss how one might practically quantify how confident one should be about the hypothesized structure. We might like to know if the amount of data that was used for estimating the structure was adequate to support the learned structure, and, if the the data looked slightly different, would the hypothesis change?

Bootstrapping [33] offers a simple approach — repeatedly resample the data with replacement, and estimate a hierarchical structure for each resampling. The difference between our setting and typical bootstrapping settings, however, is that our structures lie in a large discrete set. Thus, unlike continuous parameters, whose confidence we can often summarize with intervals or ellipses, it is not clear how one might compactly summarize a collection of many hierarchical clusterings of items.

The simplest way to summarize the collection of hierarchies obtained via the bootstrap is to measure the fraction of the estimated structures which are identical to the structure estimated from the original unperturbed dataset. If, for small sets of resampled data, the estimated hierarchy is consistently identical to that obtained from the original data, then we can be confident that the data supports the hypothesis. We show in the following example that, for the structure which was learned from the APA dataset, a far smaller dataset would have sufficed.



(a)

	{12345} {1345} {2} {13} {45}	{12345} {1234} {5} {234} {1} {2} {34}	{12345} {245} {13} {24} {5}	{12345} {2345} {1} {34} {25}
948	237	3	1	0
1487	245	3	1	1
2333	249	0	1	0
3658	250	0	0	0
5737	250	0	0	0

samples | Bootstrap distribution over trees (top k=4 trees)

(b)

Figure 43: We show the distribution of structures estimated from bootstrapped samples of the APA data (with varying sample sizes): (a) plots (in solid red) the fraction of bootstrapped trees for each sample size which agree exactly with the hierarchy given in Figure 39; In (b), we summarize the boostrap distribution for the largest sample sizes.

Example 105 (APA Election data (continued))**.** *As our final APA related example, we show the results of bootstrap resampling in Figure 43. To generate the plots, we resampled the APA dataset with replacement 200 times each for varying sample sizes, and ran our Anchors algorithm on each resulting sample. Figure 43a plots (in solid red) the fraction of bootstrapped trees for each sample size which agree exactly with the hierarchy given in Figure 39. Given that we forced sets to be partitioned until they had at most 2 items, there are 120 possible hierarchical structures for the APA dataset.*

It is interesting to see that the hierarchies returned by the algorithm are surprisingly stable even given fewer than 100 samples, with about 25% of bootstrapped trees agreeing with the optimal hierarchy. At 1000 samples, almost all trees agree with the optimal hierarchy. In Figure 43b, we show a table of the bootstrap distribution for the largest sample sizes (which were concentrated at only a handful of trees).

For larger item sets n, however, it is rarely the case that there is enough data to strongly support the hierarchy in terms of the above measure. In these cases, instead of asking whether entire structures agree with each other *exactly*, it makes sense to ask whether estimated *substructures*

agree. For example, a simple measure might amount to computing the fraction of structures estimated from resampled datasets which agreed with the original structure at the topmost partition. Another natural measure is to count the fraction of structures which correctly recovered all (or a subset of) leaf sets for the original dataset, but not necessarily the correct hierarchy. By Proposition 92, correctly discovering the leaf set partitioning is probabilistically meaningful, and corresponds to correctly identifying the d-way decomposition corresponding to a distribution, but failing to identifying the specific hierarchy.

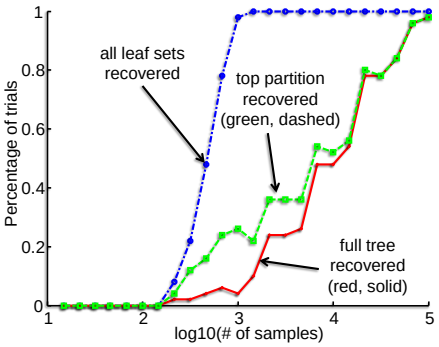
We remark that sometimes, there is no one unique structure corresponding to a distribution. The uniform distribution, for example, is consistent with any hierarchical riffle independent structure, and so bootstrapped hierarchies will not concentrate on any particular structure or even substructure. Moreover, even when there *is* true unique structure corresponding to the generating distribution, it may be the case that other simpler structures perform better when there is not much available training data.

14.5 EXPERIMENTS

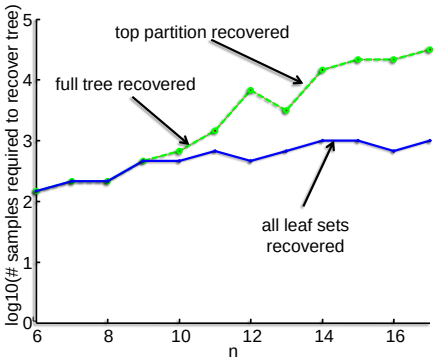
In this section, we present a series of experiments to validate our models and methods. All experiments were implemented in Matlab, except for the Fourier theoretic routines, which were written in C++. We tested on lab machines with two AMD quadcore Opteron 2.7GHz processors with 32 Gb memory. We have already analyzed the APA data extensively throughout Part III. Here, we demonstrate our algorithms on simulated data as well as other real datasets, namely, sushi preference data, and Irish election data.

14.5.1 *Simulated data*

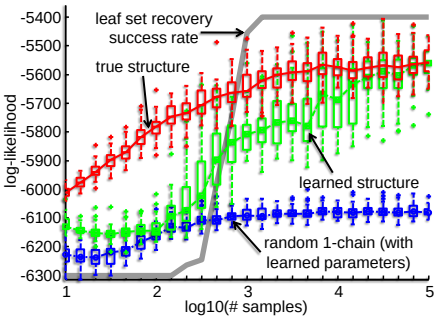
We first applied our methods to synthetic data to show that, given enough samples, our algorithms *do* effectively recover the optimal hierarchical structures which generated the original datasets. For various settings of n, we simulated data drawn jointly from a k-thin chain model (for $k = 4$) with a random parameter setting for each structure and applied our exact method for learning thin chains to each sampled dataset. First, we investigated the effect of varying sample size on the proportion of trials (out of fifty) for which our algorithms were able to (a) recover the full underlying tree structure *exactly*, (b) recover the topmost partition correctly, or (c) recover all leaf sets correctly (but possibly out of order). Figure 44a shows the result for an itemset of size $n = 16$. Figure 44b, shows, as a function of n, the number of samples that were required in the same experiments to (a) *exactly* recover the full underlying structure or (b) recover the correct leaf sets, for at least 90% of the trials. What we can observe from the plots is that, given enough samples, reliable structure recovery *is* indeed possible. It is also interesting to note that recovery of the correct leaf sets can be done with much fewer samples than are required for recovering the full hierarchical structure of the model.



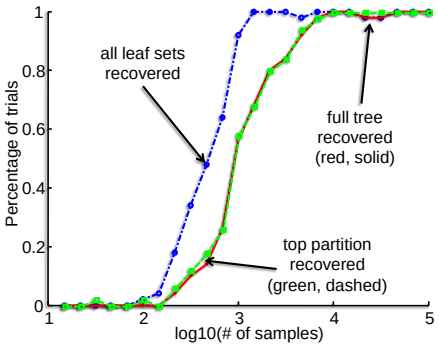
(a) Success rate for structure recovery vs. sample size ($n = 16, k = 4$)



(b) Number of samples required for structure recovery vs. number of items n



(c) Test set log-likelihood comparison



(d) Anchors algorithm success rate ($n = 16$, unknown k)

Figure 44: Structure discovery experiments on synthetic data

After learning a structure for each dataset, we learned model parameters and evaluated the log-likelihood of each model on 200 test examples drawn from the true distributions. In Figure 44c, we compare log-likelihood performance when (a) the true structure is given (but not parameters), (b) a k-thin chain is learned with known k, and (c) when we use a random generated 1-chain structure. As expected, knowing the true structure results in the best performance, and the 1-chain is overconstrained. However, our structure learning algorithm is eventually able to catch up to the performance of the true structure given enough samples. It is also interesting to note that the jump in performance at the halfway point in the plot coincides with the jump in the success rate of discovering all leaf sets correctly — we conjecture that performance is sometimes less sensitive to the actual hierarchy used, as long as the leaf sets have been correctly discovered.

To test the Anchors algorithm, we ran the same simulation using Algorithm 14.2 on data drawn from hierarchical models with no fixed k. We generated roughly balanced structures, meaning that item sets were recursively partitioned into (almost) equally sized subsets at each level of the hierarchy. From Figure 44d, we see that the Anchors algorithm can also discover the true structure given enough samples. Interestingly, the difference in sample complexity for discovering leaf sets versus discovering

1. ebi (shrimp)	2. anago (sea eel)	3. maguro (tuna)
4. ika (squid)	5. uni (sea urchin)	6. sake (salmon roe)
7. tamago (egg)	8. toro (fatty tuna)	9. tekka-maki (tuna roll)
	10. kappa-maki (cucumber roll)	

Figure 45: List of sushi types in the [70] dataset

the full tree is not nearly as pronounced as in Figure 44a. We believe that this is due to the fact that the balanced trees have less depth than the thin chains, leading to fewer opportunities for our greedy top-down approach to commit errors.

14.5.2 *Sushi preference data*

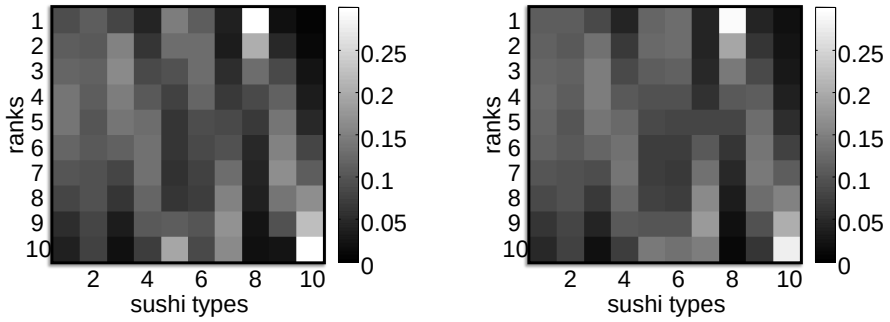
We now turn to analyzing real datasets. For our first analysis, we examine the sushi preference ranking dataset [70] consisting of 5000 full rankings of ten types of sushi. The items are enumerated (again) in Figure 45. Note that, compared to the APA election data, the sushi dataset has twice as many items, but fewer examples.

Figure 48 shows the hierarchical structure that we learn using the entire sushi dataset. Since the sushi are not prepartitioned into distinct coalitions, it is somewhat more difficult than with, say, the APA data, to interpret whether the estimated structure makes sense. However, parts of the tree certainly seem like reasonable groupings. For example, all of the tuna related sushi types have been clustered together. Tamago and kappa-maki (egg and cucumber rolls) are "safer", vegetarian choices, while uni and sake (sea urchin and salmon roe) are the somewhat more daring selections (the uni/sake category can also be described as the non-vegetarian/non-meat choices). Anago (sea eel), is the odd man out in the estimated hierarchy, being partitioned away from the remaining items at the top of the tree.

To understand the behavior of our algorithm with smaller sample sizes, we looked for features of the tree from Figure 48 which remained stable even when learning with smaller sample sizes. Figure 47 summarizes the results of our bootstrap analysis for the sushi dataset, in which we resample from the original training set 200 times at each of different sample sizes and plot the proportion of learned hierarchies which, (a) recover 'sea eel' as the topmost partition, (b) recover all leaf sets correctly, (c), recover the entire tree correctly, (d) recover the tuna-related sushi leaf set, (e) recover the {tamago, kappa-maki} leaf set, and (f) recover the {uni, sake} leaf set.

14.5.3 *Irish election data*

We next applied our algorithms to a larger Irish House of Parliament (Dáil Éireann) election dataset from the Meath constituency in Ireland (Figure 49a). The Dáil Éireann uses the *single transferable vote* (STV) election system, in which voters rank a subset of candidates. In the Meath con-



(a) First-order marginals of sushi preference data

(b) Riffle independent approximation of first-order marginals with learned hierarchy

Figure 46: Sushi preference dataset: exact first-order marginals and riffle independent approximation

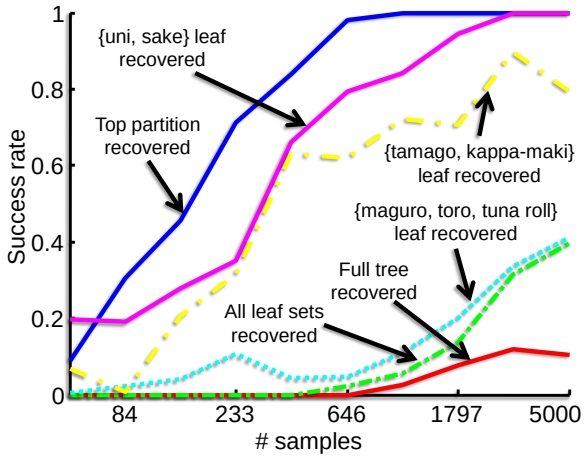


Figure 47: Stability of bootstrapped tree 'features' of the sushi dataset

stituency, there were 14 candidates in the 2002 election, running for five allotted seats. The candidates identified with the two major rival political parties, Fianna Fáil and Fine Gael, as well as a number of smaller parties (Figure 49b). See [44] for more election details (including candidate names) as well as an alternative analysis. In our experiments, we used a subset of roughly 2500 fully ranked ballots from the election.

To summarize the dataset, Figure 50a shows the matrix of first-order marginals estimated from the dataset. Candidates $\{1, 2, 4, 5, 6, 13\}$ form the set of "major" party candidates belonging to either Fianna Fáil or Fine Gael, and as shown in the figure, fared much better in the election than the other seven minor party candidates. Notably, candidates 11 and 12 (belonging to the Christian Solidary Party and Sinn Féin, respectively) received on average, the lowest ranks in the 2002 election. One of the differences between the two candidates, however, is that a significant portion of the electorate also ranked the Sinn Féin candidate very high.

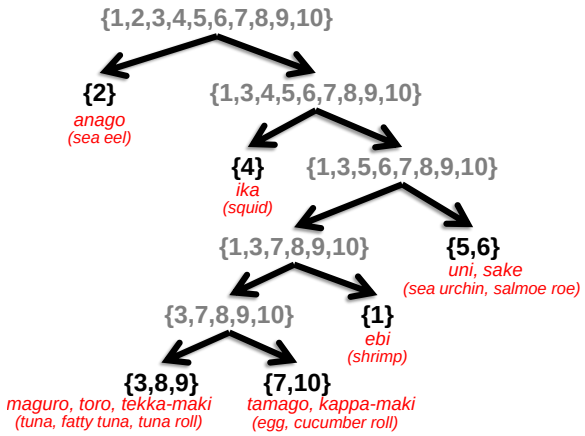
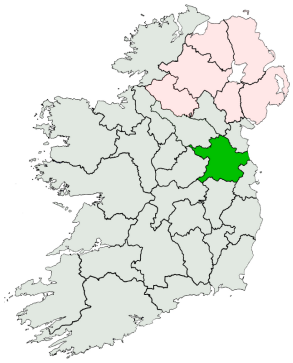


Figure 48: Learned hierarchy for sushi dataset using all 5000 rankings



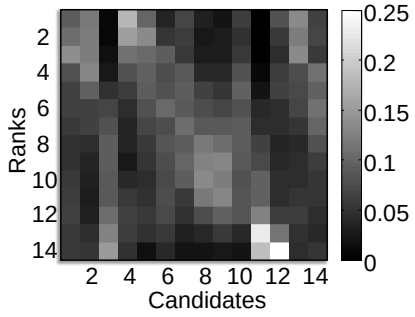
(a) The Meath constituency in Ireland, shown in green, was one of three constituencies to have electronic voting in 2002. (Map from Wikipedia)

	Candidate	Party
1	Brady, J.	Fianna Fáil
2	Bruton, J.	Fine Gael
3	Colwell, J.	Independent
4	Dempsey, N.	Fianna Fáil
5	English, D.	Fine Gael
6	Farrelly, J.	Fine Gael
7	Fitzgerald, B.	Independent
8	Kelly, T.	Independent
9	O'Brien, P.	Independent
10	O'Byrne, F.	Green Party
11	Redmond, M.	Christian Solidarity
12	Reilly, J.	Sinn Féin
13	Wallace, M.	Fianna Fáil
14	Ward, P.	Labour

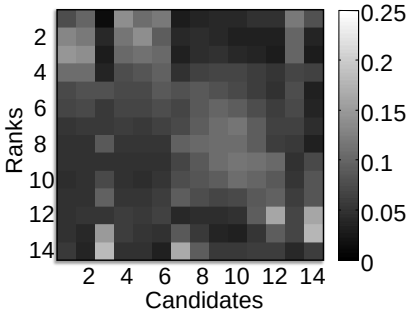
(b) List of candidates from the Meath constituency election in 2002 for five seats in the Dáil Éireann (reproduced from [44])

Figure 49: Irish election dataset summary

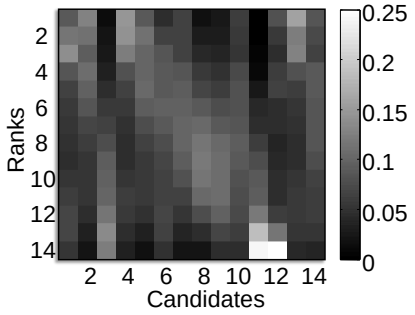
Though it may not necessarily be clear how one might partition the candidates, a natural idea might be to assume that the major party candidates (A) are riffle independent of the minor party candidates (B). In Figure 50b, we show the first-order marginals corresponding to an approximation in which A and B are assumed to be riffle independent. Visually, the approximate first-order marginals can be seen to be roughly similar to the exact first-order marginals, however there are significant features of the matrix which are not captured by the approximation — for example, the columns belonging to candidates 11 and 12 are not well approximated. In Figure 50c, we plot a more principled approximation corresponding to a learned hierarchy, which we discuss next. As can be seen, the first-order



(a) First-order marginals of Irish election data



(b) Riffle independent approximation of first-order marginals with $A = \{\text{Fianna Fáil, Fine Gael}\}$, and $B = \{\text{everything else}\}$



(c) Riffle independent approximation of first-order marginals with learned hierarchy in Figure 51

Figure 50: Irish Election dataset: exact first-order marginals and riffle independent approximations

marginals obtained via structure learning is visually much closer to the exact marginals.

As with the APA data, both the exhaustive optimization of $\hat{\mathcal{F}}$ and the Anchors algorithm returned the same tree, with running times of 69.7 seconds and 2.1 seconds respectively (not including the 3.1 seconds required for precomputing mutual informations). The resulting tree, with candidates enumerated alphabetically from 1 through 14, is shown (only up to depth 4), in Figure 51. As expected, the candidates belonging to the two major parties, Fianna Fáil and Fine Gael, are neatly partitioned into their own leaf sets. The topmost leaf is the Sinn Fein candidate, indicating that voters tended to insert him into the ranking independently of all of the other 13 candidates.

To understand the behavior of our algorithm with smaller sample sizes, we looked for features of the tree from Figure 51 which remained stable even when learning with smaller sample sizes. In Figure 52a, we resampled from the original training set 200 times at different sample sizes and plot the proportion of learned hierarchies which, (a) recover the Sinn Fein candidate as the topmost leaf, (b) partition the two major parties into leaf sets, and (c) agree with the original tree on all leaf sets, and (d) recover the entire

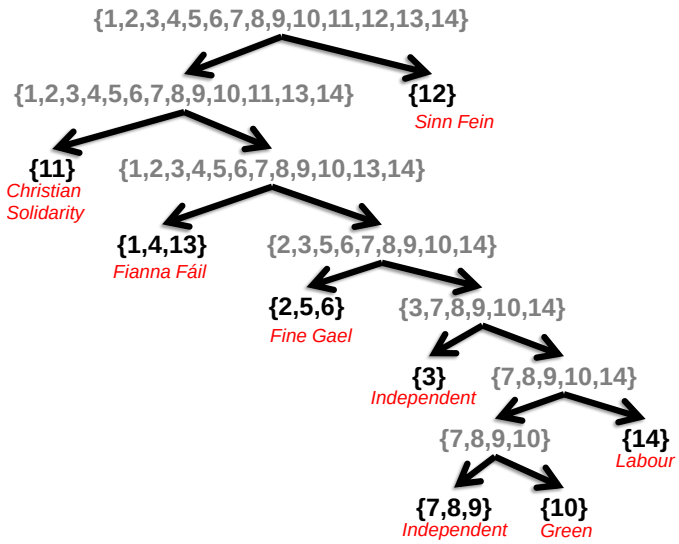
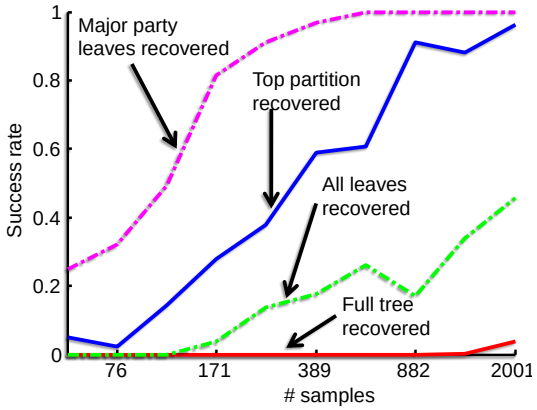


Figure 51: Learned hierarchy for Irish Election dataset using all 2500 ballots

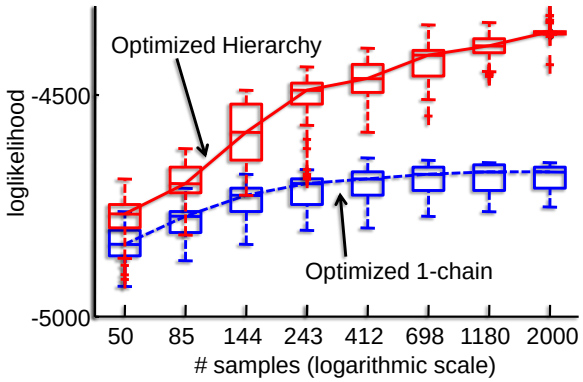
tree. Note that while the dataset is insufficient to support the entire tree structure, even with about 100 training examples, candidates belonging to the major parties are consistently grouped together indicating strong party influence in voting behavior.

We compared the results between learning a general hierarchy (without fixed k) and learning a 1-thin chain model on the Irish data. Figure 52b shows the log-likelihoods achieved by both models on a held-out test set as the training set size increases. For each training set size, we subsampled the Irish dataset 100 times to produce confidence intervals. Again, even with small sample sizes, the hierarchy outperforms the 1-chain and continually improves with more and more training data. One might think that the hierarchical models, which use more parameters are prone to overfitting, but in practice, the models learned by our algorithm devote most of the extra parameters towards modeling the correlations among the two major parties. As our results suggest, such intraparty ranking correlations are crucial for achieving good modeling performance.

Finally, we ran our structure learning algorithm on two similar but smaller election datasets from the other constituencies in the 2002 election which supported electronic voting, the Dublin North and West constituencies. Figure 53 shows the resulting hierarchies learned from each dataset. As with the Meath constituency, the Fianna Fáil and Fine Gael are consistently grouped together in leaf sets in the Dublin datasets. Interestingly, the Sinn Féin and Socialist parties are also consistently grouped in the Dublin datasets, potentially indicating some latent similarities between the two parties.



(a) Stability of bootstrapped tree 'features' of the Irish dataset



(b) Log-likelihood of held out test examples from the Irish dataset using optimized structures

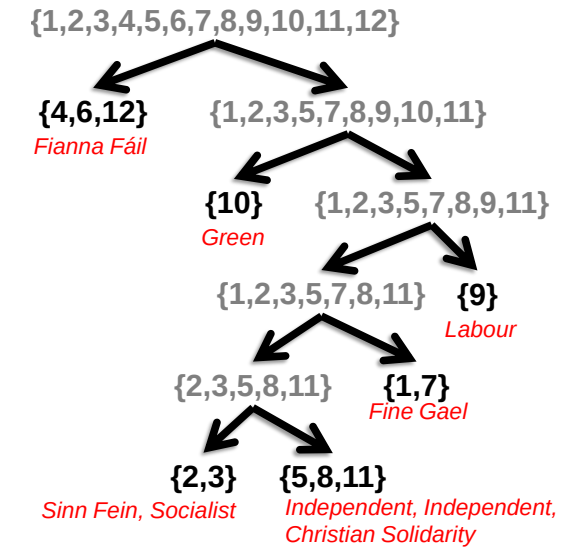
Figure 52: Structure Discovery Experiments: Irish Election dataset

14.6 CONCLUSION

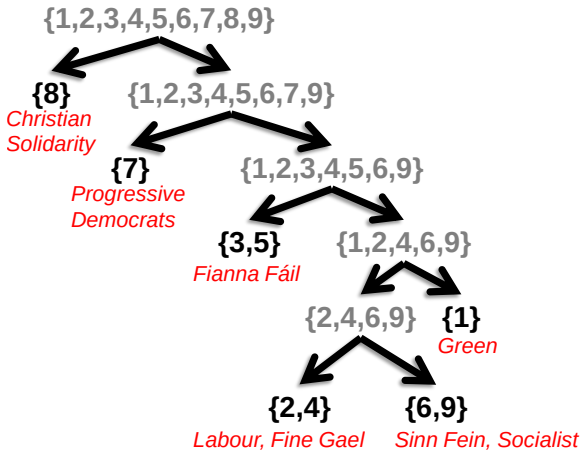
Exploiting independence structure for efficient inference and low sample complexity is a simple yet powerful idea, pervasive throughout the machine learning literature, showing up in the form of Bayesian networks, Markov random fields, and more. For rankings, independence can be problematic due to mutual exlusivity constraints, and we began Chapter 13 by indicating a need for a useful generalization of independence.

The main contribution of Chapters 13 and 14 is the definition of such a generalized notion, namely, riffled independence. There are a number of natural questions that immediately follow any such definition, such as:

- Does the generalization retain any of the computational advantages of probabilistic independence?
- Can we find evidence that such generalized independence relations hold (or approximately hold) in real datasets?
- Can we design scalable algorithms which can exploit the generalized independence structure of a model?



(a) Dublin North



(b) Dublin West

Figure 53: Learned hierarchies for Irish election data from Dublin (north) and Dublin (west) constituencies

- If subsets of items in a ranking dataset indeed satisfy the generalized independence assumption, or approximately so, how could we algorithmically determine what these subsets should be from samples?

We have shown that for riffled independence, the answer to each of the above questions lies in the affirmative. We next explored hierarchical riffle independent decompositions. Our model, in which riffle independent subsets are recursively chained together, leads to a simple, interpretable model whose structure we can estimate from data, and we have successfully applied our learning algorithms to several real datasets.

Currently, the success of our structure learning methods depends on the existence of a fairly sizeable dataset of full rankings. However, ranking datasets are more typically composed of partial or incomplete rankings, which are often far easier to elicit from a multitude of users. For example,

top-k type rankings, or even rating data (in which a user/judge provides a rating of an item between, say, 1 and 5) are common. This problem of extending our parameter and structure learning algorithms for handling such partially ranked data is the topic of the next chapter.

Many other possible extensions are possible. In our work, we have developed algorithms for estimating maximum likelihood parameters. For small training set sizes, a Bayesian approach would be more appropriate, where a prior is placed on the parameter space. However, if the prior distribution ties parameters together (i.e., if the prior does not factor across parameters), then the structure learning problem can be considerably more complicated, since we would not be able to simply identify independence relations.

Riffled independence is a new tool for analyzing ranked data and as we have shown, has the potential to give new insights into ranking datasets. We strongly believe that it will be crucial in developing fast and efficient inference and learning procedures for ranking data, and perhaps other forms of permutation data.

15

EXPLOITING RIFFLED INDEPENDENCE FOR PROBABILISTIC INFERENCE

We have thus far developed a simple interpretable model based on riffled independence relationships among ranked items. In Chapters 13 and 14, we focused primarily on the *representation problem*, in which we proposed simple and efficient algorithms for learning the structure and parameters of such a hierarchical model.

In this chapter, we turn to the *inference problem* in the context of riffled independence. While in Part II we argued that Fourier based representations are well suited to handling low-order observations involving small subsets of items at a time, we will contend in this chapter that *the structural assumption of riffled independence is particularly well suited to answering probabilistic queries about partial rankings*. We show in particular, that when riffled independence assumptions are made about a prior distribution, partial ranking observations decompose in a way that allows for efficient conditioning.

The main contributions of our work are as follows:

- When items satisfy the riffled independence relationship, we show that conditioning on partial rankings can be done efficiently, with running time linear in the number of model parameters.
- We show that in general, it is impossible to exploit riffled independence structure to efficiently condition on observations that do not take the form of partial rankings.
- We propose the first algorithm that is capable of efficiently estimating the structure and parameters of riffle independent models from heterogeneous collections of partially ranked data.
- We show results on real voting and preference data evidencing the effectiveness of our methods.

15.1 DECOMPOSABLE OBSERVATIONS

We first revisit the conditioning operation. Given a prior distribution, h, over rankings and an observation $\mathcal{O}$, recall Bayes rule (Equations 3.5), which tells us that the posterior distribution, $h(\sigma|\mathcal{O})$, is proportional to $L(\mathcal{O}|\sigma) \cdot h(\sigma)$, where $L(\mathcal{O}|\sigma)$ is the likelihood function. This operation of *conditioning* h on an observation $\mathcal{O}$ is typically computationally intractable since it requires multiplying two $n!$ dimensional functions, unless one can exploit structural decompositions of the problem. In this section, we describe a decomposition for a certain class of likelihood functions over the space of rankings in which the observations are 'factored' into simpler parts.

When an observation $\mathcal{O}$ is decomposable in this way, we show that one can efficiently condition a riffle independent prior distribution on $\mathcal{O}$. For simplicity in this chapter, we focus on *subset observations* whose likelihood functions encode membership with some subset of rankings in S_n.

Definition 106 (Observations)**.** A *subset observation* $\mathcal{O}$ is a binary observation whose likelihood is proportional to the indicator function of some subset of S_n.

As a running example, we will consider the class of *first place observations* throughout the chapter. The first place observation $\mathcal{O}$ ="Corn is ranked first", for example, is associated with the collection of rankings placing the item Corn in first place ($\mathcal{O} = \{\sigma : \sigma(\mathtt{Corn}) = 1\}$). We are interested in computing $h(\sigma|\sigma \in \mathcal{O})$. In the first place scenario, we are given a voter's top choice and we would like to infer his preferences over the remaining candidates.

Given a partitioning of the item set Ω into two subsets A and B, it is sometimes possible to *decompose* (or *factor*) a subset observation involving items in Ω into smaller subset observations involving A, B and the interleavings of A and B independently. Such decompositions can often be exploited for efficient inference.

Example 107.

- *The first place observation*

  $$\mathcal{O} = \text{"Corn is ranked first"}$$

  can be decomposed into two independent observations — an observation on the relative ranking of Vegetables,

  $$\mathcal{O}_A = \text{"Corn is ranked first among Vegetables"},$$

  and an observation on the interleaving of Vegetables and Fruits,

  $$\mathcal{O}_{A,B} = \text{"First place is occupied by a Vegetable"}.$$

  To condition on $\mathcal{O}$ in this case, one updates the relative ranking distribution over Vegetables (A) by zeroing out rankings of vegetables which do not place Corn in first place, and updates the interleaving distribution by zeroing out interleavings which do not place a Vegetable in first place, then normalizes the resulting distributions.
- *An example of a* nondecomposable *observation is the observation*

  $$\mathcal{O} = \text{"Corn is in third place"}.$$

  To see that $\mathcal{O}$ does not decompose (with respect to Vegetables and Fruits), it is enough to notice that the interleaving of Vegetables and Fruits is not independent of the relative ranking of Vegetables. If, for example, an element $\sigma \in \mathcal{O}$ interleaves A (Vegetables) and B (Fruits) as $\tau_{AB}(\sigma) = A|B|A|B$, then since $\sigma(\mathtt{Corn}) = 3$, the relative ranking of Vegetables is constrained to be $\phi_A(\sigma) = \mathtt{Peas}|\mathtt{Corn}$. Since the interleavings and relative rankings are not independent, we see that $\mathcal{O}$ cannot be decomposable.

Formally, we use riffle independent factorizations to define decomposability with respect to a hierarchy H of the item set.

Definition 108 (Decomposability)**.** Given a hierarchy H over the item set, a subset observation $\mathcal{O}$ *decomposes* with respect to H if its likelihood function $L(\mathcal{O}|\sigma)$ factors riffle independently with respect to H.

When subset observations and the prior decompose according to the same hierarchy, we can show (as in Example 107) that the posterior also decomposes.

Proposition 109. *Let H be a hierarchy over the item set. Given a prior distribution h and an observation $\mathcal{O}$ which both decompose with respect to H, the posterior distribution $h(\sigma|\mathcal{O})$ also factors riffle independently with respect to H.*

Proof. Denote the likelihood function corresponding to $\mathcal{O}$ by L (in this proof, it does not matter that $\mathcal{O}$ is assumed to be a subset observation and the result holds for arbitrary likelihoods).

We use induction on the size of the item set $n = |\Omega|$. The base case $n = 1$ is trivially true. We next examine the general case where $n > 1$. The posterior distribution, by Bayes rule, can be written $h(\sigma|\mathcal{O}) \propto L(\sigma) \cdot h(\sigma)$. There are now two cases. If H is a leaf node, then the posterior h' trivially factors according to H, and we are done. Otherwise, L and h both factor, by assumption, according to $H = (H_A, H_B)$ in the following way:

$$L(\sigma) = m_L(\tau_{AB}(\sigma)) \cdot f_L(\phi_A(\sigma)) \cdot g_L(\phi_B(\sigma)), \text{ and}$$
$$h(\sigma) = m_h(\tau_{AB}(\sigma)) \cdot f_h(\phi_A(\sigma)) \cdot g_h(\phi_B(\sigma)).$$

Multiplying and grouping terms, we see that the posterior factors as:

$$h(\sigma|\mathcal{O}) = [m_L \cdot m_h](\tau_{AB(\sigma)}) \cdot [f_L \cdot f_h](\phi_A(\sigma)) \cdot [g_L \cdot g_h](\phi_B(\sigma)).$$

To show that $h(\sigma|\mathcal{O})$ factors with respect to H, we need to demonstrate (by Definition 91) that the distributions $[f_L \cdot f_h]$ and $[g_L \cdot g_h]$ (after normalizing) factor with respect to H_A and H_B, respectively.

Since f_L and f_h both factor according to the hierarchy H_A by assumption and $|A| < n$ since H is not a leaf, we can invoke the inductive hypothesis to show that the posterior distribution, which is proportional to $f_L \cdot f_h$ must also factor according to H_A. Similarly, the distribution proportional to $g_L \cdot g_h$ must factor ordering to H_B. □

In fact, we can show that when the prior and observation both decompose with respect to the same hierarchy, *inference operations can always be performed in time linear in the number of model parameters*.

15.2 COMPLETE DECOMPOSABILITY

The condition of Proposition 109, that the prior and observation must decompose with respect to *exactly* the same hierarchy is a sufficient one for efficient inference, but it might at first glance seem so restrictive as

to render the proposition useless in practice. To overcome this limitation of "hierarchy specific" decomposability, we explore a special family of observations (which we call *completely decomposable*) for which the property of decomposability does not depend specifically on a particular hierarchy, implying in particular that for these observations, efficient inference is *always* possible (as long as efficient representation of the prior distribution is also possible).

To illustrate how an observation can decompose with respect to multiple hierarchies over the item set, consider again the first place observation $\mathcal{O}$ ="Corn is ranked first". We argued in Example 107 that $\mathcal{O}$ is a decomposable observation. Notice however that decomposability for this particular observation *does not* depend on how the items are partitioned by the hierarchy. Specifically, if instead of Vegetables and Fruits, the sets $A = \{\text{Corn}, \text{Lemons}\}$ and $B = \{\text{Peas}, \text{Oranges}\}$ are riffle independent, a similar decomposition of $\mathcal{O}$ would continue to hold, with $\mathcal{O}$ decomposing as an observation on the relative ranking of items in A ("Corn is first among items in A"), and an observation on the interleaving of A and B ("First place is occupied by some element of A").

To formally capture this notion that an observation can decompose with respect to *arbitrary* underlying hierarchies, we define *complete decomposability*:

Definition 110 (Complete decomposability)**.** We say that an observation $\mathcal{O}$ is *completely decomposable* if it decomposes with respect to *every* possible hierarchy over the item set Ω.

Finally, we denote the collection of all possible completely decomposable observations as $\mathcal{CRJ}$. See Figure 54 for an illustration of the set $\mathcal{CRJ}$.

The property of complete decomposability is a *guarantee* for an observation $\mathcal{O}$, that one can always exploit any available factorized structure of the prior distribution in order to efficiently condition on $\mathcal{O}$.

Proposition 111. *Given a prior* h *which factorizes with respect to a hierarchy* H*, and a completely decomposable observation* $\mathcal{O}$*, the posterior* $h(\sigma|\mathcal{O})$ *also decomposes with respect to* H *and can be computed in time linear in the number of model parameters of* h*.*

Proof. Proposition 109 requires that the prior and likelihood decompose with respect to the same hierarchy. However, the property of complete decomposability means that the observation $\mathcal{O}$ decomposes with respect to all hierarchies, and so we see that Proposition 111 follows as a simple corollary to Proposition 109. □

Example 112. *The* simplest *example of a completely decomposable observation is the* uniform observation $\mathcal{O}_{\mathtt{unif}} = S_\Omega$*, which includes all possible rankings and corresponds to a uniform indicator function* $\delta_{\mathtt{unif}}$ *over rankings. Given* any *hierarchy* H*,* $\delta_{\mathtt{unif}}$ *can be shown to decompose riffle independently with respect to* H*, where each factor is also uniform, and hence* $\mathcal{O}_{\mathtt{unif}}$ *is completely decomposable.*

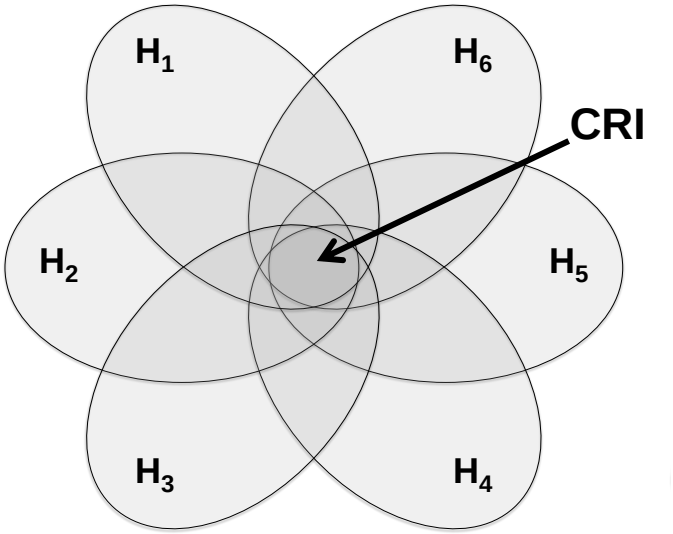


Figure 54: A diagram illustrating $\mathcal{CRI}$.

The uniform observation is of course not particularly interesting in the context of Bayesian inference, but on the other hand, given the stringent conditions in Definition 110, it is not obvious that nontrivial completely decomposable observations can even exist. Nonetheless, there do exist nontrivial examples (such as the first place observations), and in the next section, we exhibit a rich and general class of completely decomposable observations.

15.3 COMPLETE DECOMPOSABILITY OF PARTIAL RANKING OBSERVATIONS

In this section we discuss the mathematical problem of identifying all completely decomposable observations. Our main contribution in this section is to show that *completely decomposable observations correspond exactly to partial rankings of the item set*.

PARTIAL RANKINGS. We begin our discussion by introducing *partial rankings*, which allow for items to be tied with respect to a ranking σ by 'dropping' verticals from the vertical bar representation of σ.

Definition 113 (Partial ranking observation)**.** Let $\Omega_1, \Omega_2, \ldots, \Omega_k$ be an ordered collection of subsets which partition Ω (i.e., $\cup_i \Omega_i = \Omega$ and $\Omega_i \cap \Omega_j = \emptyset$ if $i \neq j$). The *partial ranking observation*[1] corresponding to this partition is the collection of rankings which rank items in Ω_i before items

1 As in [1], we note that "The term *partial ranking* used here should not be confused with two other standard objects: (1) Partial order, namely, a reflexive, transitive anti-symmetric binary relation; and (2) A ranking of a subset of Ω . In search engines, for example, although only the top-k elements of Ω are returned, the remaining $n - k$ are implicitly assumed to be ranked behind."

in Ω_j if $i < j$. We denote this partial ranking as $\Omega_1|\Omega_2|\dots|\Omega_k$ and say that it has type $\gamma = (|\Omega_1|, |\Omega_2|, \dots, |\Omega_k|)$.

We denote the collection of all partial rankings (over n items) as $\mathcal{P}$.

Each partial ranking as defined above can be viewed as a coset of the subgroup $S_\gamma = S_{\gamma_1} \times S_{\gamma_2} \times \cdots \times S_{\gamma_k}$ (see also Section 2.4). Given the type γ and any full ranking $\pi \in S_\Omega$, there is only one partial ranking of type γ containing π, thus we will therefore equivalently denote the partial ranking $\Omega_1|\Omega_2|\dots|\Omega_k$ as $S_\gamma\pi$, where π is any element of $\Omega_1|\Omega_2|\dots|\Omega_k$. Note that this *coset notation* allows for multiple rankings σ to refer to the same partial ranking $S_\gamma\sigma$.

The space of partial rankings as defined above captures a rich and natural class of observations. In particular, partial rankings encompass a number of commonly occurring special cases, which have traditionally been modeled in isolation, but in our work (as well as recent works such as [86, 84]) can be used in a unified setting.

Example 114. *Partial rankings observations include:*

- (First place, or Top-1 observations): *First place observations correspond to partial rankings of type* $\gamma = (1, n-1)$. *The observation that Corn is ranked first can be written as Corn|Peas,Lemons,Oranges.*
- (Top-k observations): *Top-*k *observations are partial rankings with type* $\gamma = (1, \dots, 1, n-k)$. *These generalize the first place observations by specifying the items mapping to the first* k *ranks, leaving all* $n-k$ *remaining items implicitly ranked behind.*
- (Desired/less desired dichotomy): *Partial rankings of type* $\gamma = (k, n-k)$ *correspond to a subset of* k *items being preferred or desired over the remaining subset of* $n-k$ *items. For example, partial rankings of type* $(k, n-k)$ *might arise in* approval voting *in which voters mark the subset of "approved" candidates, implicitly indicating disapproval of the remaining* $n-k$ *candidates.*

To show how partial rankings observations decompose, we will exhibit an explicit factorization with respect to a hierarchy H over items. For simplicity, we begin by considering the single layer case, in which the items are partitioned into two leaf sets A and B. Our factorization depends on the following notions of *consistency* of relative rankings and interleavings with a partial ranking.

Definition 115 (Restriction consistency)**.** Given a partial ranking $S_\gamma\pi = \Omega_1|\Omega_2|\dots|\Omega_k$ and any subset $A \subset \Omega$, we define the *restriction* of $S_\gamma\pi$ to A as the partial ranking on items in A obtained by intersecting each Ω_i with A. Hence the restriction of $S_\gamma\pi$ to A is:

$$[S_\gamma\pi]_A = \Omega_1 \cap A|\Omega_2 \cap A|\dots|\Omega_k \cap A.$$

Given a ranking, σ_A of items in A, we say that σ_A is *consistent* with the partial ranking $S_\gamma\pi$ if σ_A is a member of $[S_\gamma\pi]_A$.

Definition 116 (Interleaving consistency)**.** Given an interleaving τ_{AB} of two sets A, B which partition Ω, we say that τ_{AB} is *consistent* with a partial ranking $S_\gamma\pi = \Omega_1|\ldots|\Omega_k$ (with type γ) if the first γ_1 entries of τ_{AB} contain the same number of As and Bs as Ω_1, and the second γ_2 entries of τ_{AB} contain the same number of As and Bs as Ω_2, and so on. Given a partial ranking $S_\gamma\pi$, we denote the collection of consistent interleavings as $[S_\gamma\pi]_{AB}$.

For example, consider the partial ranking

$$S_\gamma\pi = \text{Corn}, \text{Lemons}|\text{Peas}, \text{Oranges},$$

which places a single vegetable and a single fruit in the first two ranks, and a single vegetable and a single fruit in the last two ranks. Alternatively, $S_\gamma\pi$ partially specifies an interleaving $AB|AB$. The full interleavings $A|B|B|A$ and $B|A|B|A$ are consistent with $S_\gamma\pi$ (by dropping vertical lines) while $A|A|B|B$ is *not* consistent (since it places two vegetables in the first two ranks).

Using the notions of consistency with a partial ranking, we show that partial ranking observations are decomposable with respect to any binary partitioning (i.e., single layer hierarchy) of the item set.

Proposition 117 (Single layer hierarchy)**.** *For any partial ranking observation $S_\gamma\pi$ and any binary partitioning of the item set (A, B), the indicator function of $S_\gamma\pi$, $\delta_{S_\gamma\pi}$, factors riffle independently as:*

$$\delta_{S_\gamma\pi}(\sigma) = m_{AB}(\tau_{AB}(\sigma)) \cdot f_A(\phi_A(\sigma)) \cdot g_B(\phi_B(\sigma)), \tag{15.1}$$

where the factors m_{AB}, f_A and g_B are the indicator functions for consistent interleavings and relative rankings, $[S_\gamma\pi]_{AB}$, $[S_\gamma\pi]_A$ and $[S_\gamma\pi]_B$, respectively.

The single layer decomposition of Proposition 117 can be turned into a recursive decomposition for partial ranking observations over arbitrary binary hierarchies, which establishes our main result. In particular, given a partial ranking $S_\gamma\pi$ and a prior distribution which factorizes according to a hierarchy H, we first condition the topmost interleaving distribution by zeroing out all parameters corresponding to interleavings which are not consistent with $S_\gamma\pi$, and normalizing the distribution.

We then need to condition the subhierarchies H_A and H_B on relative rankings of A and B which are consistent with $S_\gamma\pi$, respectively. Since these consistent sets, $[S_\gamma\pi]_A$ and $[S_\gamma\pi]_B$, are partial rankings themselves, the same algorithm for conditioning on a partial ranking can be applied recursively to each of the subhierarchies H_A and H_B.

Theorem 118. *Every partial ranking is completely decomposable ($\mathcal{P} \subset \mathcal{CRJ}$).*

To prove Theorem 118 (as well as later results), we will refer to *rank sets*.

Definition 119. Given a partial ranking of type γ, we denote the *rank set* occupied by Ω_i by R_i^γ. Note that R_i^γ depends only on γ and can be written as $R_1^\gamma = \{1, \ldots, \gamma_1\}$, $R_2^\gamma = \{\gamma_1 + 1, \ldots, \gamma_1 + \gamma_2\}$, $\ldots$, $R_k^\gamma = \{\sum_{i=1}^{k-1} \gamma_i + 1, \ldots, n\}$.

And we will refer to the following basic fact regarding rank sets:

Proposition 120. $\sigma \in S_\gamma\pi = \Omega_1|\ldots|\Omega_k$ *if and only if for each* i, $\sigma(\Omega_i) = R_i^\gamma$.

Proof. (of Theorem 118) We use induction on the size of the itemset. The cases $n = 1, 2$ are trivial since every distribution on S_1 or S_2 factors riffle independently. We now consider the more general case of $n > 2$.

Fix a partial ranking $S_\gamma\pi = \Omega_1|\Omega_2|\ldots|\Omega_k$ of type γ and a binary partition of the item set into subsets A and B. We will show that the indicator function $\delta_{S_\gamma\pi}$ factors as:

$$\delta_{S_\gamma\pi}(\sigma) = m(\tau_{AB}(\sigma)) \cdot f(\phi_A(\sigma)) \cdot g(\phi_B(\sigma)), \tag{15.2}$$

where factors m, f and g are the indicator functions for the set of consistent interleavings, $[S_\gamma\sigma]_{AB}$, and the sets of consistent relative rankings, $[S_\gamma\sigma]_A$ and $[S_\gamma\sigma]_B$, respectively. If Equation 15.2 is true, then we will have shown that $\delta_{S_\gamma\pi}$ must decompose with respect to the top layer of H. To show that $\delta_{S_\gamma\pi}$ decomposes hierarchically, we must also show that the relative ranking factors f_A and g_B decompose with respect to H_A and H_B, the subhierarchies over the item sets A and B. To establish this second step (assuming that Equation 15.2 holds), note that f_A and g_B are indicator functions for the restricted partial rankings, $[S_\gamma\sigma]_A$ and $[S_\gamma\sigma]_B$, which themselves are partial rankings over smaller item sets A and B. The inductive hypothesis (and the fact that A and B are assumed to be strictly smaller sets than Ω) then shows that the functions f_A and g_B both factor according to their respective subhierarchies.

We now turn to establishing Equation 15.2. It suffices to prove that the following two statements are equivalent:

I. The ranking σ is consistent with the partial ranking $S_\gamma\pi$ (i.e., $\sigma \in S_\gamma\pi$).

II. The following three conditions hold:
   a) The interleaving $\tau_{AB}(\sigma)$ is consistent with $S_\gamma\pi$ (i.e., $\tau_{AB}(\sigma) \in [S_\gamma\pi]_{AB}$), and
   b) The relative ranking $\phi_A(\sigma)$ is consistent with $S_\gamma\pi$ (i.e., $\phi_A(\sigma) \in [S_\gamma\pi]_A$), and
   c) The relative ranking $\phi_B(\sigma)$ is consistent with $S_\gamma\pi$ (i.e., $\phi_B(\sigma) \in [S_\gamma\pi]_B$).

- *(I ⇒ II):* We first show that $\sigma \in S_\gamma\pi$ implies conditions (a), (b) and (c).

  (a) If $\sigma \in S_\gamma\pi$, then for each i,

$$\begin{aligned} |j \in R_i^\gamma : \tau_{AB}^{-1}(j) \in A| &= |j \in R_i^\gamma : \sigma^{-1}(j) \in A|, \\ &\qquad \text{(by Definition 82)} \\ &= |k \in \Omega_i : k \in A|, \\ &\qquad \text{(by Proposition 120)} \\ &= |\Omega_i \cap A|. \end{aligned}$$

The same argument (replacing A with B) shows that for each i, we have $|j \in R_i^\gamma : \tau_{AB}(j) = 1| = |\Omega_i \cap B|$. These two conditions (by Definition 116) show that τ_{AB} is consistent with $S_\gamma\pi$.

(b) If $\sigma \in S_\gamma\pi$, then (by Definition 113) σ ranks items in Ω_i before items in Ω_j for any $i < j$. Intersecting each Ω_i with A, we also see that σ ranks any item in $\Omega_i \cap A$ before any item in $\Omega_j \cap A$ for all i, j. By Definition 82, $\phi_A(\sigma)$ also ranks any item in $\Omega_i \cap A$ before any item in $\Omega_j \cap A$ for all i, j. And finally by Definition 116 again, we see that $\phi_A(\sigma)$ is consistent with the partial ranking $S_\gamma\pi = \Omega_1 \cap A | \ldots | \Omega_k \cap A$.

(c) (Same argument as (b)).

- (II $\Rightarrow$ I): We now assume conditions (a), (b), and (c) to hold, and show that $\sigma \in S_\gamma\pi$. By Proposition 120 it is sufficient to show that if an item $k \in \Omega_i$, then $\sigma(k) \in R_i^\gamma$. To prove this claim, we show by induction on i that if an item $k \in \Omega_i \cap A$, then $\sigma(k) \in R_i^\gamma$ (and similarly if $k \in \Omega_i \cap B$, then $\sigma(k) \in R_i^\gamma$).

  Base case. In the base case ($i = 1$), we assume that $k \in \Omega_1 \cap A$, and the goal is to show that $\sigma(k) \in R_1$. By condition (a), we have that $\tau_{AB}(\sigma) \in [S_\gamma\pi]_{AB}$. By Definition 116, this means that: $|\Omega_1 \cap A| = \{j \in R_1 : [\tau_{AB}^{-1}(\sigma)](j) \in A\} = \{j \in R_1 : \sigma^{-1}(j) \in A\}$. In words, there are $m = |\Omega_1 \cap A|$ items from A which lie in rank set $R_1 = \{1, \ldots, \gamma_1\}$. To show that an item $k \in A$ maps to a rank in R_1, we now must show that in the relative ranking of elements in A, k is among the first m. By condition (b), $\phi_A(\sigma) \in [S_\gamma\pi]_A$, implying that the item subset $\Omega_1 \cap A$ occupy the first m positions in the relative ranking of A. Since $k \in \Omega_1 \cap A$, item k is among the first m items ranked by $\phi_A(\sigma)$ and therefore $\sigma(k) \in R_1$. A similar argument shows that $k \in \Omega_1 \cap B$ implies that $\sigma(k) \in R_1$.

  Inductive case. We now show that if $k \in \Omega_i \cap A$, then $\sigma(k) \in R_i$. By condition (b), $\phi_A(\sigma) \in [S_\gamma\pi]_A$, implying that the item subset $\Omega_i \cap A$ (and hence, item k) occupies the first $m = |\Omega_i \cap A|$ positions in the relative ranking of A beyond the items $\cup_{j=1}^{i-1}(\Omega_j \cap A)$. By the inductive hypothesis and mutual exclusivity, these items, together with $\cup_{j=1}^{i-1}(\Omega_j \cap B)$ occupy ranks $\cup_{j=1}^{i-1} R_j$, and therefore $\sigma(k) \in R_\ell$ for some $\ell \geqslant i$. On the other hand, condition (a) assures us that $|\Omega_i \cap A| = \{j \in R_i : \sigma^{-1}(j) \in A\}$ — or in other words, that the ranks in R_i are occupied by exactly m items of A. Therefore, $\sigma(k) \in R_i$. Again, a similar argument shows that $k \in \Omega_i \cap B$ implies that $\sigma(k) \in R_i$.

□

See Algorithm 15.1 for details on our recursive conditioning algorithm. As a consequence of Theorem 118 and Proposition 111, conditioning on partial ranking observations can be performed in linear time with respect to the number of model parameters.

Algorithm 15.1: Pseudocode for PRCONDITION, an algorithm for recursively conditioning a hierarchical riffle independent prior distribution on partial ranking observations. See Definitions 115 and 116 for $[S_\gamma\sigma]_A$, $[S_\gamma\sigma]_B$, and $[S_\gamma\sigma]_{AB}$. The runtime of PRCONDITION is linear in the number of model parameters. Input: All parameter distributions of the prior h_{prior} represented in explicit tabular form, and an observation $S_\gamma\pi$ in the form of a partial ranking. Output: All parameter distributions of the posterior h_{post} represented in explicit tabular form.

PRCONDITION (Prior h_{prior}, Hierarchy H, Observation $S_\gamma\pi = \Omega_1|\Omega_2|\dots|\Omega_k$)
  if *isLeaf*(H) **then**
    forall σ **do**

$$h_{post}(\sigma) \leftarrow \begin{cases} h_{prior}(\sigma) & \text{if } \sigma \in S_\gamma\pi \\ 0 & \text{otherwise} \end{cases};$$

    end
    NORMALIZE (h_{post}) ;
    return (h_{post});
  end
  else
    forall τ **do**

$$m_{post}(\tau) \leftarrow \begin{cases} m_{prior}(\tau) & \text{if } \tau \in [S_\gamma\pi]_{AB} \\ 0 & \text{otherwise} \end{cases};$$

    end
    NORMALIZE (m_{post}) ;
    $f(\phi_A) \leftarrow$PRCONDITION ($f_{prior}, H_A, [S_\gamma\pi]_A$) ;
    $g(\phi_B) \leftarrow$PRCONDITION ($g_{prior}, H_B, [S_\gamma\pi]_B$) ;
    return ($m_{post}, f_{post}, g_{post}$);
  end

15.3.1 *An impossibility result*

It is interesting to consider what completely decomposable observations exist beyond partial rankings. One of our main contributions is to show that there are no such observations.

Theorem 121 (Converse of Theorem 118)**.** Every *completely decomposable observation takes the form of a partial ranking* ($\mathcal{CRI} \subset \mathcal{P}$).

Together, Theorems 118 and 121 form a significant insight into the nature of rankings, showing that the notions of partial rankings and complete decomposability *exactly* coincide. In fact, our result shows that it is even possible to define partial rankings via complete decomposability!

As a practical matter, our results show that there is no algorithm based on simple multiplicative updates to the parameters which can exactly condition on observations which do *not* take the form of partial rankings. If one is interested in conditioning on such observations, Theorem 121 suggests that a slower or approximate inference approach might be necessary.

15.3.2 *Proof of the impossiblity result (Theorem 121)*

We now turn to proving Theorem 121. Since this proof is significantly longer than the one for its converse, we sketch the main ideas that drive the proof and refer interested readers to details in Appendix F.

Recall that the definition of the *linear span* of a set of vectors in a vector space is the intersection of all linear subspaces containing that set of vectors. To prove Theorem 121, we introduce analogous concepts of the *span* of a set of rankings.

Definition 122 (RSPAN and PSPAN)**.** Let $X \subset S_n$ be any collection of rankings. We define $\text{PSPAN}(X)$ to be the intersection of all partial rankings containing X. Similarly, we define $\text{RSPAN}(X)$ to be the intersection of all completely decomposable observations containing X. More formally,

$$\text{PSPAN}(X) = \bigcap_{S_\gamma\sigma : X \subset S_\gamma\sigma} S_\gamma\sigma, \text{ and}$$

$$\text{RSPAN}(X) = \bigcap_{\mathcal{O} : X \subset \mathcal{O},\ \mathcal{O} \in \mathcal{CRI}} \mathcal{O}.$$

Our proof strategy is to establish two claims: (1) that the PSPAN of any set is always a partial ranking, and (2) that in fact, the RSPAN and PSPAN of a set X are exactly the same sets. Since claim (1) is a fact about partial rankings and does not involve riffled independence, we defer all related proofs to the appendix. Thus we have:

Lemma 123. *For any $X \subset S_n$, $\text{PSPAN}(X)$ is a partial ranking.*

Proof. See Appendix. □

The following discussion will instead sketch proofs of claim (2). If claims (1) and (2) indeed hold, we first show that Theorem 121 must hold.

Proof. (of Theorem 121): Let $\mathcal{O} \in \mathcal{CRI}$ and let $X = \mathcal{O}$. We have $\mathcal{O} = \text{RSPAN}(\mathcal{O})$. Since $\text{RSPAN}(X) = \text{PSPAN}(X)$ by claim (2),we have that $X = \text{PSPAN}(X)$. By Lemma 178, we know that $\text{PSPAN}(X)$ is a partial ranking, and therefore $X = \mathcal{O}$ must also be a partial ranking. □

We now proceed to establish the claim that $\text{RSPAN}(X) = \text{PSPAN}(X)$. The following proposition lists several basic properties of the RSPAN that we will use in several of the proofs. They all follow directly from definition so we do not write out the proofs.

Proposition 124.

I. (Monotonicity) For any X, $X \subset \text{RSPAN}(X)$.

II. (Subset preservation) For any X, X' such that $X \subset X$, $\text{RSPAN}(X) \subset \text{RSPAN}(X')$.

III. (Idempotence) For any X, $\text{RSPAN}(\text{RSPAN}(X)) = \text{RSPAN}(X)$.

One inclusion follows directly from the fact that $\mathcal{P} \subset \mathcal{CRI}$ (Theorem: 118):

Lemma 125. *For any subset of orderings, X, $\text{RSPAN}(X) \subset \text{PSPAN}(X)$.*

Proof. Lemma 125 follows almost directly from the fact that $\mathcal{P} \subset \mathcal{CRI}$ (Theorem 118). Fix a subset $X \subset S_n$ and let π be any element of $\text{RSPAN}(X)$. Consider any partial ranking indicator function $\delta \in \mathcal{P}$ such that $\delta(\sigma) > 0$ for all $\sigma \in X$. We want to see that $\delta(\pi) > 0$. By Theorem 118, $\delta \in \mathcal{CRI}$. Moreover, since $\pi \in \text{RSPAN}(X)$, and $\delta(\sigma) > 0$ for all $\sigma \in X$, we conclude that $\delta(\pi) > 0$ (by Definition 122). □

Algorithm 15.2: Pseudocode for computing PSPAN(X). FORMPSPAN(X) takes a set of partial rankings (or full rankings) X as input and outputs a partial ranking. This algorithm iteratively deletes vertical bars from elements of X until they are in agreement. Note that it is not necessary to keep track of t, but we do so here to ease notation in the proofs. Nor is this algorithm the most direct way of computing PSPAN(X), but again, it simplifies the proof of our main theorem.

```
FORMPSPAN(X)
    X_0 ← X; t ← 0;
    while ∃S_γ π, S_γ' π' ∈ X_t which disagree on the relative ordering of items a_1, a_2 do
        X_t ← ∅ ;
        foreach S_γ σ ∈ X_t do
            Add any partial ranking obtained by deleting a vertical bar from S_γ σ
            between items a_1 and a_2 to X_t;
        end
        t ← t + 1;
    end
    return (any element of X_t) ;
```

We now consider the problem of computing the partial ranking span (PSPAN) of a given set of rankings X. In Algorithm 15.2, we show a simple procedure that provably outputs the correct result.

Proposition 126. *Given a set of rankings X as input, Algorithm 15.2 outputs* PSPAN(X).

Proof. See Appendix. □

As a final step before being able to prove our second main claim, that RSPAN(X) = PSPAN(X) for any X, we prove the following two technical lemmas about Algorithm 15.2 which form the heart of our argument. In particular, for a completely decomposable observation $\mathcal{O} \in \mathcal{CRI}$, Lemma 127 below shows a ranking contained in $\mathcal{O}$ can "force" other rankings to be contained in $\mathcal{O}$.

Lemma 127. *Let $\mathcal{O} \in \mathcal{CRI}$ and suppose there exist $\pi_1, \pi_2 \in S_n$ which disagree on the relative ranking of items $i, j \in \Omega$ such that $\pi_1, \pi_2 \in \mathcal{O}$. Then the ranking obtained by swapping the relative ranking of items i, j within any $\pi_3 \in \mathcal{O}$ must also be contained in $\mathcal{O}$.*

Proof. Let h be the indicator distribution corresponding to the observation $\mathcal{O}$. We will show that swapping the relative ranking of items i, j in π_3 will result in a ranking which is assigned nonzero probability by h, thus showing that this new ranking is contained in $\mathcal{O}$.

Let $A = \{i, j\}$ and $B = \Omega \backslash A$. Since $\mathcal{O} \in \mathcal{CRI}$, h must factor riffle independently according to the partition (A, B). Thus,

$$h(\pi_1) = m(\tau_{AB}(\pi_1)) \cdot f(\phi_A(\pi_1)) \cdot g(\phi_B(\pi_1)) > 0, \text{ and}$$
$$h(\pi_2) = m(\tau_{AB}(\pi_2)) \cdot f(\phi_A(\pi_2)) \cdot g(\phi_B(\pi_2)) > 0.$$

Since π_1 and π_2 disagree on the relative ranking of items in A, this factorization implies in particular that both $f(\phi_A = i|j) > 0$ and $f(\phi_A = j|i) > 0$.

Since $h(\pi_3) > 0$, it must also be that each of $m(\tau_{AB}(\pi_3))$, $f(\phi_A(\pi_3))$, and $g(\phi_B(\pi_3))$ have positive probability. We can therefore swap the relative ranking of A, ϕ_A, to obtain a new ranking which has positive probability since all of the terms in the decomposition of this new ranking have positive probability. □

Lemma 128 provides conditions under which removing a vertical bar from one of the rankings in X will not change the support of a completely riffle independent distribution. The key strategy in this proof is to argue that large subsets of rankings must be contained in a completely decomposable observation $\mathcal{O}$ by decomposing rankings into transpositions (using Lemma 8) and invoking the technical lemma from above (Lemma 127) repeatedly.

Lemma 128. *Let $S_\gamma\pi = \Omega_1|\ldots|\Omega_i|\Omega_{i+1}|\ldots|\Omega_k$ be a partial ranking on item set Ω, and $S_{\gamma'}\pi' = \Omega_1|\ldots|\Omega_i \cup \Omega_{i+1}|\ldots|\Omega_k$, the partial ranking in which the sets Ω_i and Ω_{i+1} are merged. Let $a_1 \in \cup_{j=1}^{i}\Omega_j$ and $a_2 \in \cup_{j=i+1}^{k}\Omega_j$. If $\mathcal{O}$ is any element of $\mathcal{CRI}$ such that $S_\gamma\pi \subset \mathcal{O}$ and there additionally exists a ranking $\tilde{\pi} \in \mathcal{O}$ which disagrees with $S_\gamma\pi$ on the relative ordering of a_1, a_2, then $S_{\gamma'}\pi' \subset \mathcal{O}$.*

Proof. See Appendix. □

Recall that Lemma 125 showed that $\text{RSPAN}(X) \subset \text{PSPAN}(X)$. We now use Lemma 128 to show the reverse inclusion also holds, establishing that the two sets are in fact equal and therefore proving the desired result, that $\mathcal{CRI} \subset \mathcal{P}$.

Proposition 129. *For any subset of orderings, X, $\text{RSPAN}(X) \supset \text{PSPAN}(X)$.*

Proof. At each iteration t, Algorithm 15.2 produces a set of partial rankings, X_t. We denote the union of all partial rankings at time t as $\tilde{X}_t \equiv \bigcup_{S_\gamma\sigma \in X_t} S_\gamma\sigma$. Note that $\tilde{X}_0 = X$ and $\tilde{X}_T = \text{PSPAN}(X)$. The idea of our proof will be to show that at each iteration t, the following set inclusion holds: $\text{RSPAN}(\tilde{X}_t) \subset \text{RSPAN}(\tilde{X}_{t-1})$. If indeed this holds, then after the final iteration T, we will have shown that:

$$
\begin{aligned}
\text{PSPAN}(X) &= \tilde{X}_T, && \text{(Proposition 126)} \\
&\subset \text{RSPAN}(\tilde{X}_T), \\
&\quad \text{(Monotonicity, Proposition 124)} \\
&\subset \text{RSPAN}(\tilde{X}_0), \\
&\quad (\text{RSPAN}(\tilde{X}_t) \subset \text{RSPAN}(\tilde{X}_{t-1}), \text{ shown below}), \\
&\subset \text{RSPAN}(X) && (\tilde{X}_0 = X, \text{ see Algorithm 15.2})
\end{aligned}
$$

which would prove the Proposition.

It remains now to show that $\text{RSPAN}(\tilde{X}_t) \subset \text{RSPAN}(\tilde{X}_{t-1})$. We claim that $\tilde{X}_t \subset \text{RSPAN}(\tilde{X}_{t-1})$. Let $\sigma \in \tilde{X}_t$. If $\sigma \in \tilde{X}_{t-1}$, then since $\tilde{X}_{t-1} \subset \text{RSPAN}(\tilde{X}_{t-1})$, we have $\sigma \in \text{RSPAN}(\tilde{X}_{t-1})$ and the proof is done. Otherwise, $\sigma \in \tilde{X}_t \backslash \tilde{X}_{t-1}$. In this second case, we use the fact that at iteration t, the vertical bar between Ω_i and Ω_{i+1} was deleted from the partial ranking

$S_\gamma\pi = \Omega_1|\ldots|\Omega_i|\Omega_{i+1}|\ldots|\Omega_k$ (which is a subset of $\tilde{X}_{t-1}$) to form the partial ranking $S_{\gamma'}\pi' = \Omega_1|\ldots|\Omega_i \cup \Omega_{i+1}|\ldots|\Omega_k$. (which is a subset of $\tilde{X}_t$). Furthermore, in order for the vertical bar to have been deleted by the algorithm, there must have existed some partial ranking (and therefore some full ranking ω') that disagreed with $S_\gamma\pi$ on the relative ordering of items a_1, a_2 on opposite sides of the bar. Since $\sigma \in \tilde{X}_t \backslash \tilde{X}_{t-1}$ we can assume that $\sigma \in S_{\gamma'}\pi'$.

We now would like to apply Lemma 128. Note that for any $\mathcal{O} \in \mathcal{CRI}$ such that $\tilde{X}_{t-1} \subset \mathcal{O}$, we also have $S_\gamma\pi \subset \mathcal{O}$, since $S_\gamma\pi \subset \tilde{X}_{t-1}$. An application of Lemma 128 then shows that $S_{\gamma'}\pi' \subset \mathcal{O}$ and therefore that $\sigma \in \mathcal{O}$.

We have shown in fact that $\sigma \in \mathcal{O}$ holds for *any* $\mathcal{O} \in \mathcal{CRI}$ such that $\tilde{X}_{t-1} \subset \mathcal{O}$, and therefore taking the intersection of supports over all $\mathcal{O} \in \mathcal{CRI}$, we see that $\tilde{X}_t \subset \text{RSPAN}(\tilde{X}_{t-1})$. Taking the RSPAN of both sides yields:

$$\begin{aligned}\text{RSPAN}(\tilde{X}_t) &\subset \text{RSPAN}(\text{RSPAN}(\tilde{X}_{t-1})),\\ &\qquad \text{(Subset preservation, Proposition 124)}\\ &\subset \text{RSPAN}(\tilde{X}_{t-1}).\\ &\qquad \text{(Idempotence, Proposition 124)}\end{aligned}$$

□

15.4 MODEL ESTIMATION FROM PARTIALLY RANKED DATA

In many ranking based applications, datasets are predominantly composed of partial rankings rather than full rankings due to the fact that for humans, partial rankings are typically easier and faster to specify. In addition, many datasets are heterogenous, containing partial ranking of different types. For example, in Irish House of Parliament elections, voters are allowed to specify their top-k candidate choices for any value of k (see Figure 56). In this section we use the efficient inference algorithm proposed in Section 15.3 for estimating a riffle independent model from partially ranked data. Because estimating a model using partially ranked data is typically considered to be more difficult than estimating one using only full rankings, a common practice (see for example [55]) has been to simply ignore the partial rankings in a dataset. The ability of a method to incorporate *all* of the available data however, can lead to significantly improved model accuracy as well as wider applicability of that method. In this section, *we propose the first efficient method for estimating the structure and parameters of a hierarchical riffle independent model from heterogeneous datasets consisting of arbitrary partial ranking types*. Central to our approach is the idea that given someone's partial preferences, we can use the efficient algorithms developed in the previous section to infer his full preferences and consequently apply previously proposed algorithms which are designed to work with full rankings.

CENSORING INTERPRETATIONS OF PARTIAL RANKINGS. The model estimation problem for full rankings is stated as follows. Given i.i.d. training examples $\sigma^{(1)}, \ldots, \sigma^{(m)}$ (consisting of full rankings) drawn from a

hierarchical riffle independent distribution h, recover the structure and parameters of h.

In the partial ranking setting, we again assume i.i.d. draws, but that each training example $\sigma^{(i)}$ undergoes a censoring process producing a partial ranking consistent with $\sigma^{(i)}$. For example, censoring might only allow for the ranking of the top-k items of $\sigma^{(i)}$ to be observed. While we allow for arbitrary types of partial rankings to arise via censoring, we make a common assumption that the partial ranking type resulting from censoring $\sigma^{(i)}$ does not depend on $\sigma^{(i)}$ itself.

ALGORITHM. We treat the model estimation from partial rankings problem as a missing data problem. As with many such problems, if we could determine the full ranking corresponding to each observation in the data, then we could apply algorithms which work in the completely observed data setting. Since full rankings are not given, we utilize an Expectation-Maximization (EM) approach in which we use inference to compute a posterior distribution over full rankings given the observed partial ranking. In our case, we then apply the algorithms from [55, 56] which were designed to estimate the hierarchical structure of a model and its parameters from a dataset of full rankings.

Given an initial model h, our EM-based approach alternates between the following two steps until convergence is achieved.

- **(E-step)**: For each partial ranking, $S_\gamma\pi$, in the training examples, we use inference to compute a posterior distribution over the full ranking σ that could have generated $S_\gamma\pi$ via censoring, $h(\sigma|\mathcal{O} = S_\gamma\pi)$. Since the observations take the form of partial rankings, we use the efficient algorithms in Section 15.3 to perform the E-step.
- **(M-step)**: In the M-step, one maximizes the expected log-likelihood of the training data with respect to the model. When the hierarchical structure of the model has been provided, or is known beforehand, our M-step can be performed using standard methods for optimizing parameters. When the structure is *unknown*, we use a *structural EM* approach, which is analogous to methods from the graphical models literature for structure learning from incomplete data [39, 40].

  Unfortunately, the (riffled independence) structure learning algorithm of [55] is unable to directly use the posterior distributions computed from the E-step. Instead, observing that sampling from riffle independent models can be done efficiently and exactly (as opposed to, for example, MCMC methods), we simply sample full rankings from the posterior distributions computed in the E-step and pass these full rankings into the structure learning algorithm of [55]. The number of samples that are necessary, instead of scaling factorially, scales according to the number of samples required to detect riffled independence (which under mild assumptions is polynomial in n, [55]).

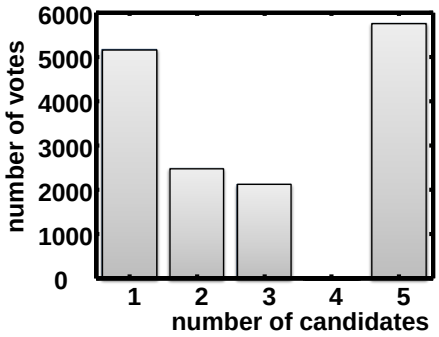


Figure 55: Histogram of top-k ballot lengths in the APA election data. Over half of voters provide only their top-3 or top-4 choices

15.5 EXPERIMENTS

We demonstrate our algorithms on simulated data as well as real datasets taken from different domains.

In addition to roughly 5000 full rankings, the APA dataset has over 10,000 top-k rankings of 5 candidates. In previous chapters, we had used only the full rankings of the APA data, but now we are able to use the entire dataset. Figure 55 plots, for each $k \in \{1, \ldots, 5\}$, the number of ballots in the Meath data of length k.

Likewise, the *Meath dataset* [44] which was taken from the 2002 Irish Parliament election has over 60,000 top-k rankings of 14 candidates. As with the APA data, we had used only the full rankings of the Meath data in previous chapters, but here we use the entire dataset. Figure 56 plots, for each $k \in \{1, \ldots, 14\}$, the number of ballots in the Meath data of length k. In particular, note that the vast majority of ballots in the dataset consist of partial rather than full rankings. We can run inference on over 5000 top-k examples for the Meath data in 10 seconds on a dual 3.0 GHz Pentium machine with an unoptimized Python implementation. Using 'brute force' inference, we estimate that the same job would require roughly one hundred years.

We extracted a third dataset from a database of *searchtrails* collected by [134], in which browsing sessions of roughly 2000 users were logged during 2008-2009. In many cases, users are unlikely to read articles about the same story twice, and so it is often possible to think of the order in which a user reads through a collection of articles as a top-k ranking over articles concerning a particular story/topic. The ability to model visit orderings would allow us to make long term predictions about user browsing behavior, or even recommend 'curriculums' over articles for users. We ran our algorithms on roughly 300 visit orderings for the eight most popular posts from `www.huffingtonpost.com` concerning 'Sarah Palin', a popular subject during the 2008 U.S. presidential election. Since no user visited every article, there are no full rankings in the data and thus the method of 'ignoring' partial rankings does not work.

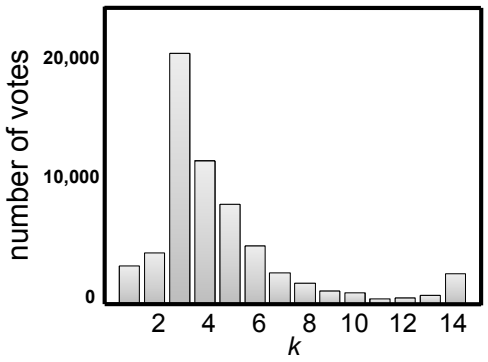


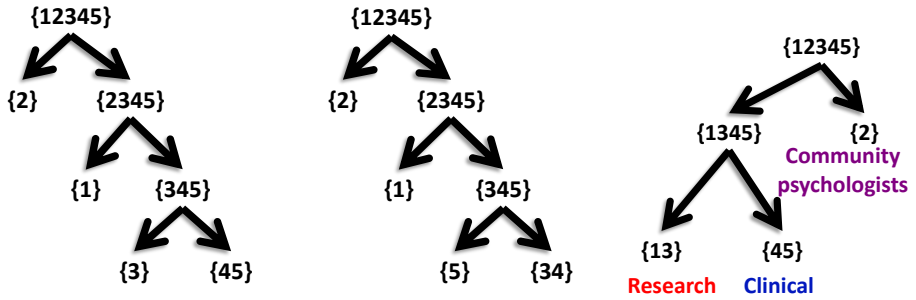
Figure 56: Histogram of top-k ballot lengths in the Irish election data. Over half of voters provide only their top-3 or top-4 choices

APA STRUCTURE LEARNING RESULTS. We now complete our long running APA dataset example. Because of the large number of full rankings in the APA data, we subsampled a dataset of 300 training examples from the complete APA dataset. Performed structure learning using *only* the full rankings of these 300 training examples, one obtains the structure in Figure 57a, which can be seen to not match the 'correct' structure of Figure 39 which was learned using 5000 full rankings. Figures 57b and 57c plot the results of our EM algorithm with the former showing the result after just a single EM iteration and the latter showing the result after structural convergence, which occurs by the third iteration, showing that our method can learn the 'correct' structure given 300 training examples.

We compared our EM algorithm against two alternative approaches that we call *FlatEM* and *Uniform Fill-in*. The FlatEM algorithm is the same as the EM algorithm above except for two details: (1) it performs conditioning exhaustively instead of exploiting the factorized model structure, and (2) it performs the M-step without sampling. The Uniform Fill-in approach treats every top-k ranking in the tranining set as a uniform collection of votes for all of the full rankings consistent with that top-k ranking, and is accomplished by using just one iteration of our EM algorithm.

In Figure 58a we plot test set loglikelihoods corresponding to each approach, with EM and FlatEM having almost identical results and both performing much better than the Uniform Fill-in approach. On the other hand, Figure 58b which plots running times shows that FlatEM can be far more costly (for most datasets, it cannot even be run in a reasonable amount of time).

To verify that partial rankings *do* indeed make a difference in the APA data, we plot the results of estimating a model from the subsets of APA training data consisting of top-k rankings with length larger than some fixed k. Figures 58c and 58d show the likelihood and running times for $k = 0, 1, 2, 3$ with $k = 0$ being the entire training set and $k = 3$ being the subset of training data consisting only of full rankings. As our results show, including partial rankings does indeed help on average for improving test log-likelihood (with diminishing returns).

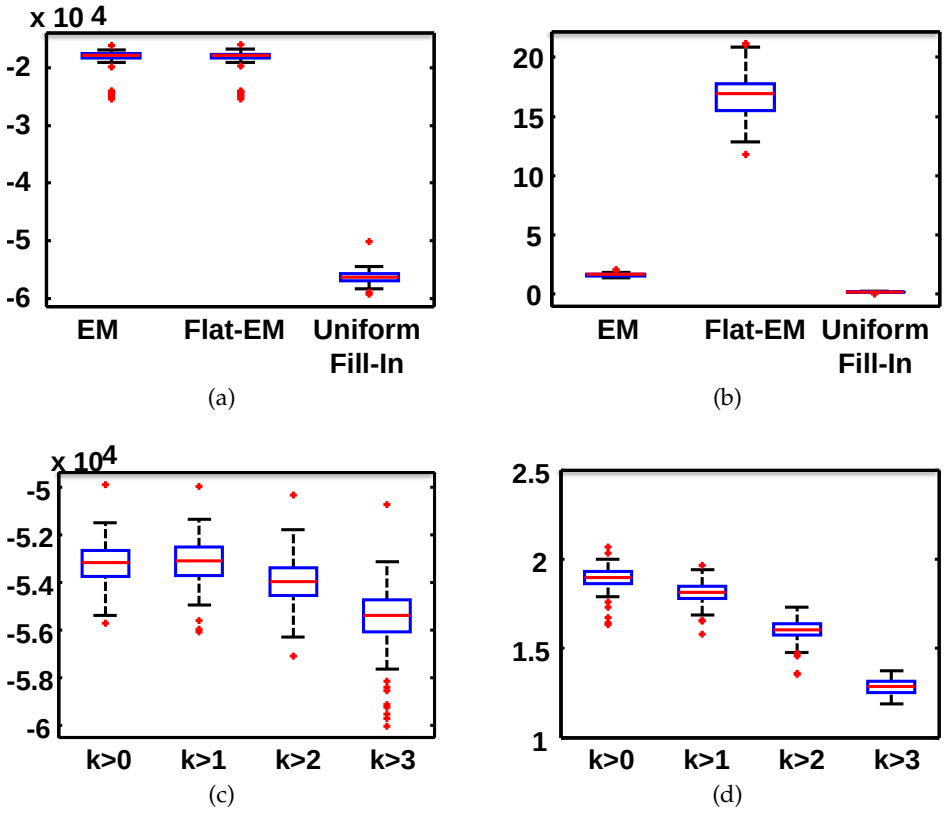


(a) Structure learned using only the subset of full rankings (out of the 300 given training examples)

(b) Structure learned using all training examples after 1 iteration of EM

(c) Structure learned using all training examples after structural convergence (3 iterations)

Figure 57: Structure learning with a subset of the APA dataset (300 rankings, randomly sampled)



(a) (b)

(c) (d)

Figure 58: APA experimental results — each experiment repeated with 200 bootstrapped resamplings of the data

STRUCTURE DISCOVERY WITH EM WITH LARGER n. In all experiments, we initialize distributions to be uniform, and do not use random restarts. Our experiments have led to several observations about using EM for learning with partial rankings. First, we observe that typical runs converge to a fixed structure quickly, with no more than three EM iterations. Fig-

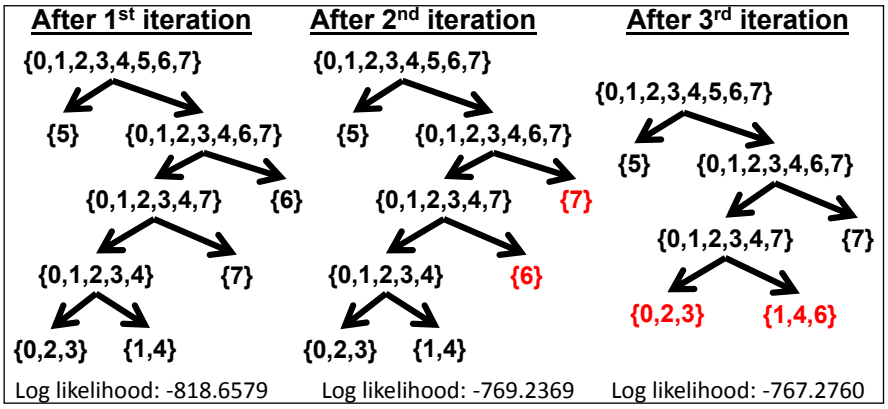
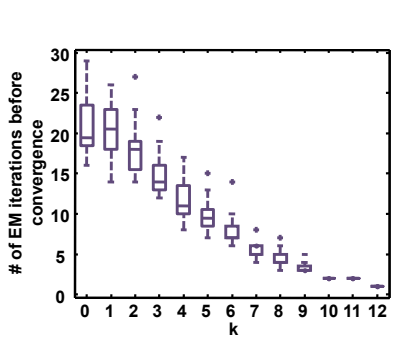
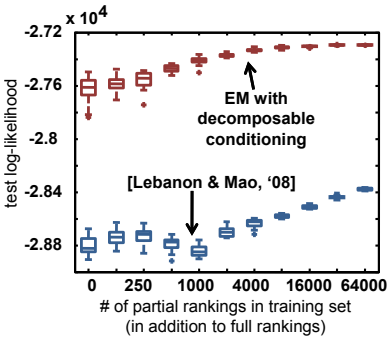


Figure 59: Iterations of Structure EM for the Sarah Palin data with structural changes at each iteration highlighted in red. This figure is best viewed in color.



(a) # of EM iterations required for convergence if the training set only contains rankings of length longer than k.



(b) Density estimation from synthetic data. We plot test loglikelihood when learning from 343 full rankings and between 0 and 64,000 additional partial rankings

Figure 60: Synthetic data results

ure 59 shows the progress of EM on the Sarah Palin data, whose structure converges by the third iteration. As expected, the log-likelihood increases at each iteration, and we remark that the structure becomes more interpretable — for example, the leaf set $\{0, 2, 3\}$ corresponds to the three posts about Palin's wardrobe before the election, while the posts from the leaf set $\{1, 4, 6\}$ were related to verbal gaffes made by Palin during the campaign.

Secondly, the number of EM iterations required to reach convergence in log-likelihood depends on the types of partial rankings observed. We ran our algorithm on subsets of the Meath dataset, each time training on $m = 2000$ rankings all with length larger than some fixed k. Figure 60a shows the number of iterations required for convergence as a function of k (with 20 bootstrap trials for each k). We observe fastest convergence for datasets consisting of almost-full rankings and slowest convergence for those consisting of almost-empty rankings, with almost 25 iterations necessary if one trains using rankings of all types. Finally we remark that

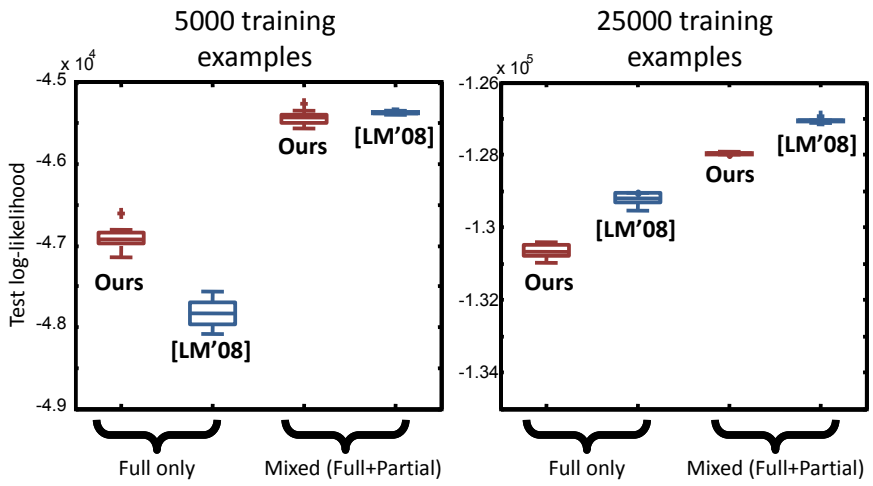


Figure 61: Density estimation from small (5000 examples) and large subsets (25000 examples) of the Meath data. We compare our method against [86] training (1) on all available data and (2) on the subset of full rankings.

the model obtained after the first iteration of EM is interesting and can be thought of as the result of pretending that each voter is completely ambivalent regarding the $n - k$ unspecified candidates.

THE VALUE OF PARTIAL RANKINGS. We now show that using partial rankings in addition to full rankings allows us to achieve better density estimates. We first learned models from synthetic data drawn from a hierarchy, training using 343 full rankings plus varying numbers of partial ranking examples (ranging between 0-64,000). We repeat each setting with 20 bootstrap trials, and for evaluation, we compute the log-likelihood of a testset with 5000 examples. For speed, we learn a structure H only once and fix H to learn parameters for each trial. Figure 60b, which plots the test log-likelihood as a function of the number of partial rankings made available to the training set, shows that we are indeed able to learn more accurate distributions as more and more data are made available.

COMPARING TO A NONPARAMETRIC MODEL. Comparing the performance of riffle independent models to other approaches was not possible in previous chapters since we had not been able to handle partial rankings. Using the methods developed in this chapter, however, we compare riffle independent models with the state-of-the-art nonparametric Lebanon-Mao (LM08) estimator of [86] on the same data (setting their regularization parameter to be $C = 1,2,5$, or 10 via a validation set). Figure 60b shows (naturally) that when the data are drawn synthetically from a riffle independent model, then our EM method significantly outperforms the LM08 estimator.

For the Meath data, which is only approximately riffle independent, we trained on subsets of size 5,000 and 25,000 (testing on remaining data). For each subset, we evaluated our EM algorithm for learning a riffle indepen-

dent model against the LM08 estimator when (1) using only full ranking data, and (2) using all data. As before, both methods do better when partial rankings are made available.

For the smaller training set, the riffle independent model performs as well or better than the LM'08 estimator. For the larger training set of 25,000, we see that the nonparametric method starts to perform slightly better on average, the advantage of a nonparametric model being that it is guaranteed to be consistent, converging to the correct model given enough data. The advantage of riffle independent models, however, is that they are simple, interpretable, and can highlight global structures hidden within the data.

15.6 CONCLUSION

In probabilistic reasoning problems, it is often the case that certain data types suggest certain distribution representations. For example, sparse dependency structure in the data often suggests a Markov random field (or other graphical model) representation [39, 40]. For low-order permutation observations (depending on only a few items at a time), we have shown (in Part II) that a Fourier domain representation is appropriate. Our work in this chapter shows, on the other hand, that when the observed data takes the form of partial rankings, then hierarchical riffle independent models are a natural representation.

As with conjugate priors, we showed that a riffle independent model is guaranteed to retain its factorization structure after conditioning on a partial ranking (which can be performed in linear time). Most surprisingly, our work shows that observations which do not take the form of partial rankings are not amenable to simple multiplicative update based conditioning algorithms. Finally, we showed that it is possible to learn hierarchical riffle independent models from partially ranked data, significantly extending the applicability of previous work.

An interesting future research direction along these lines is to consider what approximations might be appropriate for conditioning riffle independent priors on *non*-partial ranking observations. Perhaps a discouraging conclusion for our study is that, despite being so ubiquitous, pairwise comparisons are not decomposable with respect to every riffle independent model. Further work is necessary to investigate whether, for example, a single fruit can be compared to a single vegetable, even if approximately. For models that are *not* riffle independent but are used in applications in which we care about partial ranking inference, we believe that the framework developed in this chapter could be used as a principled approximation.

16

RELATED WORK

WHILE our definition of riffled independence is original, many parts of our work in Part III draw from a variety of different literatures, such as card shuffling research due primarily to Persi Diaconis and collaborators [11, 41], papers about Fourier theoretic probabilistic inference over permutations from the machine learning community[77, 59, 60], as well as graphical model structure learning research.

16.0.1 *Card shuffling theory*

Bayer and Diaconis [11] provided a a convergence analysis of repeated riffle shuffles, showing famously that the number of shuffles required to sufficiently randomize a standard 52 card deck is seven. Our novelty lies in the combination of shuffling theory with probabilistic independence, which was first exploited in our paper (Huang et al. [60]) for scaling inference operations to large problems. As discussed in Section 13.2, Fulman [41] introduced a class of shuffles known as biased riffle shuffles which are not the same as the biased riffle shuffles discussed in Chapter 13. The fact that the uniform riffle shuffling can be realized by dropping card with probability proportional to the number of cards remaining in each hand has been observed in a number of papers [11], but we are the first to (1) formalize this in the form of the recurrence given in Equation 13.2, and (2) to compute the Fourier transform of the uniform and biased riffle shuffling distributions. Recently, Kondor and Barbosa [81] has used a similar interleaving distribution in the context of evaluating kernel functions between partial rankings.

16.0.2 *Fourier analysis on permutations*

Our dynamic programming approach for computing the Fourier transform of a biased riffle shuffle (Section 13.3) bears some similarities to the FFT (Fast Fourier Transform) algorithm proposed by Clausen and Baum [20], and in particular, relies on the same branching rule recursions [114] that we introduced in Chapter 6. While the Clausen FFT requires $O(n! \log(n!))$ time, since our biased riffle shuffles are parameterized by a single α, we can use the recurrence to compute low-frequency Fourier terms in polynomial time.

16.0.3 *Learning structured representations*

Our insights for the structure learning problem described in Chapter 14 are inspired by some of the recent approaches in the machine learning liter-

ature for learning the structure of thin junction trees (Bach and Jordan [7]). In particular, the idea of using a low order proxy objective with a graph-cut like optimization algorithm is similar to an idea which was recently introduced in Shahaf et al. [118], which determines optimally thin separators with respect to the Bethe free energy approximation (of the entropy) rather than a typical log-likelihood objective. Our sample analysis is based on the mutual information sample complexity bounds derived in Hoffgen [51], which was also used in Chechetka and Guestrin [16] for developing a structure learning algorithm for thin junction trees with provably polynomial sample complexity. Our EM based methods for learning structure from partially observed data is directly inspired by the structural EM approaches which were pioneered by Friedman [39]. Finally, the bootstrap methods which we have employed in our experiments for verifying robustness bear much resemblance to some of the common bootstrapping methods which have been used in bioinformatics for analyzing phylogenetic trees (Holmes [52, 53]).

16.0.4 *Mallows models*

As we have discussed (Section 14.1.3), our riffled independence models generalize the popular class of Mallows models and generalized Mallows models. Mallows models (as well as other similar distance based models) have the advantage that they can compactly represent distributions for very large n, and admit conjugate prior distributions (Meila et al. [97]). Estimating parameters has been a popular problem for statisticians — recovering the optimal σ_0 from data is known as the *consensus ranking* or *rank aggregation* problem and is known to be NP-hard (Bartholdi et al. [10]). Many authors have focused on approximation algorithms instead.

Like Gaussian distributions, Mallows models also tend to lack flexibility, and so Lebanon and Mao [86] propose a nonparametric model of ranked (and partially ranked) data based on placing weighted Mallows kernels on top of training examples, which, as they show, can realize a far richer class of distributions, and can be learned efficiently. However, they do not address the inference problem, and it is not immediately clear in many Mallows models papers whether one can efficiently perform inference operations like marginalization and conditioning in such models. Our work on the other hand typically leads to a class of distributions which is both rich as well as interpretable, and additionally, we have identified precise conditions under which efficient conditioning is possible (the conditions being that the observations take the form of partial rankings).

There are in fact several recent works to model partial rankings using Mallows based models. Busse et al. [15] learned finite mixtures of Mallows models from top-k data (also using an EM approach). Lebanon and Mao [86] developed a nonparametric model based (also) on Mallows models which can handle arbitrary types of partial rankings. In both settings, a central problem is to marginalize a Mallows model over all full rankings which are consistent with a particular partial ranking. To do so efficiently,

both papers rely on the fact (first shown in Fligner and Verducci [36]) that this marginalization step can be performed in closed form. This closed form equation of Fligner and Verducci [36], however, can be seen as a very special case of our setting since Mallows models can always be shown to factor riffle independently according to a chain structure.Specifically, to compute the sum over rankings which are consistent with a partial ranking $S_\gamma \sigma$, it is necessary to condition on $S_\gamma \sigma$, and to compute the normalization constant of the resulting function. The conditioning step can be performed using the methods in Chapter 15, and the normalization constant can be computed by multiplying the normalization constant of each factor of the hierarchical decomposition. Thus, instead of resorting to the more complicated mathematics based on inversion combinatorics, our theory of complete decomposability offers a simple conceptual way to understand why Mallows models can be conditioned efficiently on partial ranking observations.

Finally in recent related work, Lu and Boutilier [90] considered an even more general class of observations based on DAG (directed acycle graph) based observations in which probabilities of rankings which are not consistent with a DAG of relative ranking relations are set to zero. Lu and Boutilier [90] show in particular that the conditioning problem for their DAG based class of observations is #P-hard. They additionally propose an efficient rejection sampling method for performing probabilistic inference within the general class of DAG observations and prove that the sampling method is *exact* for the class of partial rankings that we have discussed in Chapter 15.

Part IV

FUTURE DIRECTIONS AND DISCUSSION

17

EXTENSIONS AND OPEN QUESTIONS

In this chapter, we propose a number of interesting extensions and open questions that we consider to be promsing for future investigation.

17.1 FEATURE BASED GENERALIZATION ABILITY.

In real applications, items are related to each other in complex ways. For example, if a user reports a preference for the movie "Indiana Jones" over "Amores Perros", he is most likely actually reporting something more general, more complex, and perhaps more nebulous, perhaps that he has a preference for a particular genre, or director, or language, etc.

Probabilistic models that account for feature based relationships among items would allow for us to more accurately capture preference models over real item sets as well as to draw more information out of a single training example. Such models would allow for generalization ability to previously unencountered items and allow one to scale to ranking problems over much larger item collections. The question is: how can we model features without sacrificing the compactness of our representations and efficiency of our inference techniques?

It would be particularly interesting to integrate features into both of our additive and multiplicative based decompositions. For example, one might consider using linear regression to relate feature vectors with Fourier coefficients.

17.2 FOURIER ANALYSIS: OPEN QUESTIONS AND EXTENSIONS

APPROXIMATE FOURIER BASED INFERENCE GUARANTEES. We have some basic results about the errors that can be introduced by bandlimiting and how those errors can be propagated by typical inference operations. Currently, what is missing is a Boyen/Koller-like result ([13]) which would presumably bound the deviation from the true distribution at all future timesteps assuming certain conditions on the mixing distribution. While it is true that the KL-divergence between the true distribution and the approximate distribution can be shown to decrease monotonically at each timestep, we do not have similar results yet for the conditioning step and it is unclear how one should work with the KL-divergence functional in the Fourier domain. Any theoretical bound is more likely to be stated in terms of an L_2 related distance.

OPTIMIZATION IN THE FOURIER DOMAIN. A crucial problem which we have ignored up until now has been that of optimizing a function over permutations. A natural question that one might ask is whether it is easy

to maximize a bandlimited function? In particular, for a fixed bandlimiting level, we would like to know if it is possible to maximize functions within the bandlimited class in polynomial time. Things seem rosy given that the maximum value of a first-order function on permutation can be found in polynomial time ($O(n^3)$) using the popular Hungarian (Kuhn-Munkres) algorithm [101]. Unfortunately, beyond first-order, functions become far more difficult to optimize, and in fact we have shown (in Theorem 57) that the problem of optimizing a second-order function is NP-hard. To make matters worse, the reduction given in Theorem 57 is an approximation preserving reduction and there are no constant factor approximation algorithms for the general TSP case. However, it would be useful in many situations to have optimization routines which account for higher order effects. To this end, we plan to explore various strategies for optimizing these bandlimited functions that work well in practice. Another avenue to explore is to search for problem structure which can be exploited to guarantee optimality or near-optimality in certain cases (like convexity and submodularity, respectively).

FINDING ONLY THE SIGNIFICANT FOURIER COEFFICIENTS As we have discussed in previous chapters, the polynomial sized growth of low-order Fourier coefficient matrices are still too unwieldy for tractable inference, particularly after $n > 30$ or so. While we have addressed scaling problems in this thesis by additionally exploiting probabilistic independence, another tantalizing approach is to attempt to maintain only the Fourier coefficients which contribute significantly toward a distribution's energy.

How to discovery such "significant" Fourier coefficients for distributions over the symmetric group is, as of yet, an open problem. One possible starting point might lie in the work of Akavia [2] who studied a similar problem for functions over abelian groups, showing that it is possible to deterministically (and in sublinear time) pinpoint the Fourier coefficients that contribute greater than a τ-fraction (say, 1%) of the total energy of the function.

OTHER GROUPS AND ALGEBRAIC STRUCTURES. The Fourier theoretic inference routines that we presented in Chapter 8, as we mentioned, is not specific to the symmetric group, with most of its results carrying over to finite and compact Lie groups. As an example, the noncommutative group of rotation operators in three dimensions, $SO(3)$, appears in settings which model the pose of a three dimensional object. Elements in $SO(3)$ might be used to represent the pose of a robot arm in robotics, or the orientation of a mesh in computer graphics; In many settings, it would be useful to have a compact representation of uncertainty over poses. We believe that there are many other application domains with algebraic structure where similar probabilistic inference algorithms might apply, and in particular, that noncommutative settings offer a particularly challenging but exciting opportunity for machine learning research.

In a similar vein, we are also considering sets which do not have group structure but are acted upon by some group. One such structure is known as the *rook monoid* which is the set of partial matchings between two collections A and B. We believe that studying distributions over the rook monoid will prove to be useful for matching problems in computer vision which need to deal with spurious measurements that need not be matched with anything.

QUANTIFYING UNCERTAINTY. It would be useful to measure the uncertainty of a distribution over permutations. For example, the entropy functional is one common such measure. The challenging part of writing entropy in terms of Fourier coefficients, however, is that logarithms are not easily expressed using Fourier coefficients. Furthermore, with a truncated set of coefficients, it is impossible to hope for an exact measure, and the best one could do would be to derive upper/lower bounds on the uncertainty measure. We have considered Taylor approximations to the entropy functional, but so far there has not been much success with the current approach due to poor performance in bandlimited settings. It seems more useful to look for other measures of uncertainty that are more naturally expressible with Fourier coefficients.

ALTERNATIVE PROJECTION BASES FOR FUNCTIONS ON PERMUTATIONS. In Part II, we focused exclusively on projections to the Gel'fand-Tsetlin basis (Chapter 6) which has many useful properties particularly in the context of developing efficient algorithms. However, it is possible that alternative basis sets may also lead to interesting algorithms.

For example, wavelet bases which have been applied ([27]) ubiquitously in engineering may also be useful for permutations. Wavelets functions, which can be localized both in frequency and space, typically are better at capturing both large scale diffuse effects as well as small scale local effects (with which low-frequency Fourier representations have difficulties). There are a number of open questions, however, including: (1) Which wavelet bases are appropriate for functions on permutations? (2) How can we efficiently exploit sparse wavelet structure for probabilistic inference? And (3) how can one recover fundamental quantities (such as marginal probabilities) without explicitly tabulating the function values of basis elements?

17.3 RIFFLED INDEPENDENCE: OPEN QUESTIONS AND EXTENSIONS

TRACTABLE INFERENCE WITH INCOMPLETE RANKINGS. We showed in Chapter 15 that one can exploit riffled independence structure to condition on an observation if and only if it takes the form of a partial ranking. While the space of partial rankings is both rich and useful in many settings, it does not cover an important class of observations: that of *incomplete rankings*, which are defined to be a ranking (or partial ranking) of a subset of the itemset Ω. For example, Theorem 121 shows that the conditioning problem for pairwise observations of the form "Apples are preferred over

Bananas" is nondecomposable. Note that Top-k rankings are considered to be "complete" rankings since they implicitly rank all other items in the last $n - k$ positions.

How then, can we tractably condition on incomplete rankings? One possible approach is to convert to a Fourier representation using Algorithm 13.3, then conditioning on a pairwise ranking observation using Kronecker conditioning (Algorithm 8.4). This Fourier domain approach would be useful if one were particularly interested in low-order marginals of the posterior distributions.

One example in particular may be the calculation of the tripletwise marginals $h(\sigma(i)|\sigma(j) < \sigma(k))$ which arise in the structure learning problem (Chapter 14). Here, after conditioning on the pairwise ranking $\sigma(j) < \sigma(k)$, we are interested only in the first-order marginal probabilities associated with item i.

When the Fourier approach is not viable, another option may be to assume that the posterior distribution takes on a particular riffle independent structure (in the same way that mean field methods would assume a factorized posterior). The research question of interest is: which hierarchical structure should be used for the purposes of approximating the posterior?

STRUCTURE LEARNING EXTENSIONS. Our study of structure learning in the context of riffled independence can be extended in a number of directions. In this thesis, we have focused on a *top-down* clustering based approach for structure discovery. In some circumstances, however, it may be more useful to pursue a *bottom-up* approach, where instead of recursively partitioning sets of items, one merges items, then merges subsets of items together recursively. For example, with large itemsets, there may not be enough samples to discover an entire hierarchy, but the training set might be adequate for learning small hierarchies among subsets of items. The main open question here is how to quantify the "affinity" for two subsets (which may not necessarily cover the entire item set) to be merged, as opposed to quantifying the "discord" between two subsets as we have done in Chapter 14

Another interesting open question is whether there exists an algorithm with a constant factor approximation guarantees for the structure learning problem. While our sample complexity guarantees hold for data that is drawn from some underlying distribution which truly factors as a riffle independent hierarchy, it is not known whether our ANCHORS algorithm is guaranteed to optimize or approximately optimize the desired objective function. One potential starting point would be to examine the randomized greedy approach of Mathieu and Schudy [96] who consider a similar setting and establish approximation bounds.

PARAMETRIC INTERLEAVING MODELS. In Part III, we considered, on the one hand, interleaving models in which the probability of each possible interleaving is explicitly represented and tabulated. On the other hand we

have also considered the highly parameterized biased riffle shuffles with only one parameter which make strong Markovian assumptions.

The space between these two extremes is worth exploring, particularly in the context of real ranking data. For example, there are simple generalizations of the biased riffle shuffle which make the same Markovian assumptions, but allow for different bias probabilities depending on the bottom card of each the left and right hands, reflecting, perhaps, that crunchy *fruits* are preferred over crunchy *vegetables*, but soft *vegetables* are preferred over soft *fruits*. Additionally, one can consider relaxing the Markovian assumptions on card drops to account for longer range dependencies, imitating a "sticky" card effect.

LEARNING FROM DEPENDENT DATA. In this thesis, we have assumed throughout that training examples are independent and identically distributed. However in practice these are not always safe assumptions as a number of factors such as the user interface can impact the validity of both. For example, in an internet survey in which a user must perform a series of preference ranking tasks in sequence, a worry is that prior ranking tasks may bias the results of his future rankings.

Another source of bias lies in the reference ranking that may be displayed, in which the user is asked to rearrange items by 'dragging and dropping'. On the one hand, showing everyone the same reference ranking may bias the resulting data. But on the other hand, showing every user a different reference ranking may mean that the training examples are not exactly identically distributed.

Understanding, identifying, and finally, learning in spite of the different types of biases that may occur in eliciting preference data remains a fundamental problem in ranking.

STUDYING STRATEGIC VOTING. It is interesting to consider the differences between the actual vote distributions considered in this thesis against the approximate riffle independent distributions. Take the APA dataset, for example, in which the optimal approximation by a riffle independent hierarchy reflects the underlying political coalitions within the organization. Upon comparison between the approximation and the empirical distribution, however, some marked difference arise. For example, the riffle independent approximation underestimates the number of votes obtained by candidate 3 (a research psychologist) who ultimately won the election.

One possible explanation for the discrepancy may lie in the idea that voters tend to vote strategically in APA elections, placing stronger candidates of opposing political coalitions lower in the ranking, rather than revealing their true preferences. An interesting line of future work lies in detecting and studying the presence of such strategic voting in election datasets. Open questions include (1) verifying mathematically whether strategic voting does indeed exist in, say, the APA election data, and (2) if so, why the strategic voting effect is not strong enough to overwhelm our riffle independence structure learning algorithms.

18

THESIS SUMMARY AND DISCUSSION

PERMUTATIONS — mathematically defined as one-to-one mappings from a finite set to itself — are all around us. In computer science, permutations arise in diverse applications ranging from matching problems to multitarget tracking problems, to the rankings of webpages by a search engine, and even to the analysis of the ordering of events in a news story. Despite their ubiquity, approaches for reasoning with uncertainty (i.e., probability distributions) over permutations have historically been limited to restricted families of distributions due to the fact that there are $n!$ permutations for any n objects.

In this thesis, we have studied tractable representations of uncertainty on the group of permutations and efficient algorithms for performing inference using these representations. We have focused on two general strategies for decomposing large distributions into compactly representable parts based on additive and multiplicative decompositions, respectively. An underlying theme throughout is the idea that both kinds of structural decompositions can be employed in tandem to relax the apparent intractability of probabilistic reasoning over the space of permutations. Using these decompositions, we developed general, efficient techniques applicable to multiple domains. For example, we applied our methods both to solving challenging camera tracking scenarios as well as to reveal voting patterns in Irish elections which had never been identified before. In the following two sections, we summarize the main contributions of this thesis.

18.1 ADDITIVE DECOMPOSITIONS.

Taking inspiration from signal processing, we extensively explored the idea of additively decomposing distributions over permutations into a *weighted sum of low frequency Fourier basis functions*. Unlike the common discrete Fourier transform appearing ubiquitously in digital applications, Fourier transforms on permutations are a modern development based on group theory and until recently, have only been studied theoretically [29]. The work of this thesis, along with that of collaborators such as Leonidas Guibas and Risi Kondor, represent the first successful applications of these modern Fourier analytic methods to machine learning.

The key insight to understanding Fourier transforms on permutations is that simple marginal probabilities, such as the *first order probabilities* (the probability that a *single item*, say, the movie *Braveheart*, maps to rank 1, and is thus the favorite), can be efficiently computed from the "low-frequency" Fourier coefficients of a distribution. Moreover, we discussed how these these low-frequency coefficients can be appropriately generalized to "higher-frequency" coefficients which capture more complicated

marginals involving *multiple items* (like the *second-order probability* that two movies, *Braveheart* and *Forrest Gump*, map to rank 1 and rank 2, respectively). Storing low frequency coefficients is equivalent to storing low-order probabilities involving only small subsets of items, which can be represented with polynomial storage complexity. On the other hand, storing high frequency coefficients leads to higher quality approximations but at the cost of a much higher storage complexity, and therefore our Fourier methods offer a principled approach for trading off between storage complexity and approximation accuracy [59]. The Fourier perspective also offers a new approach for approximate probabilistic reasoning — in particular, we can ignore high-frequency coefficients and run probabilistic reasoning operations by working only with the low frequency coefficients. This line of thought led us to develop a number of highly efficient and provably correct algorithms for performing probabilistic reasoning using these Fourier theoretic representations [59].

APPLICATION TO MULTI-TARGET TRACKING. It may seem surprising, at first blush, to learn that tracking multiple moving persons is much more difficult than tracking a single person The difficulty is rooted in the so-called *data association* (or *identity management*) problem in which one must associate the n measured trajectories with a permutation of the n identities of the persons being tracked. We applied our algorithms for tracking multiple humans moving in an environment equipped with a networked array of twelve cameras. With multiple views, occlusions and incomplete camera coverage, data association problems make tracking extremely challenging (and completely intractable for exact methods), but our Fourier approximations manage to perform data association at a near optimal rate ($\sim 88\%$ the rate of an 'omniscient' tracker which is given the true data associations) [57].

THEORY. Performing probabilistic reasoning algorithms with a truncated Fourier transform can, unsurprisingly, lead to errors. For a number of common reasoning operations, we provided theoretical results illuminating the nature of error propagation in the Fourier domain. As an example, we identified many situations under which the probability estimates produced by our methods can be expected to be *exact*, thus showing that our approximations can sometimes be tight [57, 59]. Finally, a surprising and theoretically significant result is that many of our algorithms can be applied, essentially unchanged, to performing probabilistic reasoning operations over *any finite or compact Lie group* [59]. For example, it may seem that representing distributions over permutations should require substantially different mathematics compared to representing distributions over the rotation matrices or quaternions (which are fundamental to applications in computer vision and robotics). However, our work shows that all of these exotic but useful spaces can really be treated within the same mathematical framework.

18.2 MULTIPLICATIVE DECOMPOSITIONS.

While exactly representing a distribution over permutations of 20-30 objects would require millions of petabytes, our low frequency decompositions might only require a hundred megabytes. To scale to even larger problems, we proposed to additionally exploit probabilistic independence (corresponding to multiplicative decompositions of distributions) [60]. For example, when tracking many persons, it is often unnecessary to jointly reason over all n persons. Intuitively, groups of people who are far apart do not interact (or interact weakly), and thus it would suffice to reason over individual groups independently of each other.

We developed algorithms for detecting independence using Fourier coefficients and have applied these algorithms to track large numbers of objects by adaptively finding independent subgroups of objects and factoring the distribution appropriately. Experimentally we have shown that our algorithm can handle $n = 100$ objects in a challenging ant tracking dataset [72], while the bits required to *exactly* represent such a distribution outnumbers the atoms in the universe.

RIFFLED INDEPENDENCE FOR RANKINGS. Though moving groups of objects can sometimes be approximately tracked independently, not all permutation data is well modeled using naïve independence decompositions. To handle *ranked data*, we introduced ([54]) a novel probabilistic concept which we have called *riffled independence*, that has proven to be a far more appropriate assumption for rankings. Our introduction of riffled independence has led to not only the development of computationally efficient learning algorithms, but also has given us completely new insights into existing ranking datasets.

Experimentally, we showed that while items in real ranking datasets are never independent in the ordinary sense, there are a number of datasets in which items exhibit *riffled independence* [54, 55, 56]. In Irish election datasets, for example, we showed that the many Irish political factions are often near riffle independent of each other, thus revealing a hidden voting pattern that had previously gone unnoticed. Our methodology also applies to preference datasets — we discovered, based on a food survey conducted in Japan [70], that certain subsets of sushi types (such as tuna based sushi and vegetarian options) are also near riffle independent.

Much of the progress and theoretical results that have come from our work on riffled independence has paralleled the theoretical development of graphical models. Algorithmically, we introduced efficient learning algorithms which operate only in the Fourier domain [54], allowing us to simultaneously exploit additive and multiplicative structure. We also introduced polynomial time algorithms which can automatically discover subsets of riffle independent items from low-frequency statistics of the ranking data [55] and provided guarantees that, given polynomially many rankings, our algorithms are successful with high probability. Finally, we established an intimate connection between the concept of riffled indepen-

dence and partial rankings, which has allowed us to develop algorithms exploiting riffled independence structure for efficiently computing probabilistic reasoning queries [61].

18.3 FUTURE RESEARCH DIRECTIONS

My long term research objective is to further our understanding of the role that permutations and related mathematical mappings play in human preference, information presentation and information acquisition. In particular, I am interested in identifying structure in each of these problems and exploiting this structure to develop highly scalable, empirically effective, and provably correct algorithms for reasoning and learning with rich, descriptive probability distributions in these domains. In my dissertation research, I have taken several steps towards this goal. We now understand several fundamental and realistic structural assumptions that one can use for permutation data. Furthermore, we know how to exploit these structural constraints to reduce the computational costs associated with learning and reasoning with permutations by orders of magnitude.

I hope to apply the insights that have resulted from my thesis work to new research directions, which I outline below. The following long-term research directions can thus broadly be partitioned into two categories: (1) new and exciting applications of reasoning with permutations, and (2) going beyond permutations to study probabilistic reasoning on more general mapping spaces with applications to geometric/spatial reasoning.

18.3.1 *New applications*

RANKING RELEVANT INFORMATION IN A SENSOR NETWORK. Modern advances in sensor network technologies and mobile devices have enabled us to have constant access to a wealth of information from a multitude of modalities. One day these technologies will, for example, enable from near-instant notification of disasters such as earthquakes and tsunamis, to personalized health monitoring. In such massive-scale networks, we run into the problem of coping with the deluge of data coming from multiple sources. For any user, we could potentially show them any number of things sensed by any node in the network, but it is infeasible to show the user everything. Instead, we would like to only present relevant and interesting information. For example, while an 8.9 magnitude earthquake may be relevant information for everyone in the world, a farmers market announcement might only be relevant to local residents.

I am interested in *designing efficient probabilistic methods for ranking sensor information with respect to relevance for a particular user*. Unlike ranking problems in ordinary information retrieval problems, there are a number of problems which I believe are unique to the sensor network domain. For example, (1) how do we deal with widely heterogeneous sources of sensor data ranging from not only emails and 'tweets', but also to temperature, images, and sounds? (2) How do we efficiently optimize over the space

of rankings on small, power-constrained devices? (3) How can we deal with ranking aggregated sources of information? (For example, the fact that someone in Denver has the flu may not be important, but the fact that hundreds in Denver have contracted a particular virus may be significant). Solving these problems would allow us to harness much more of the vast potential of large scale sensor networks.

AUTOMATIC CURRICULUM DESIGN. Today, if we were to query Google or Wikipedia for, say, 'trigonometry', we would receive a number of highly relevant websites and articles. What these search engines do not tell us is the optimal order in which we should read through these articles. Intuitively however, the order (or ranking) in which we acquire information is deeply tied to the speed at which we can learn. Today, if we were to query Google or Wikipedia for, say, 'trigonometry', we would receive a number of highly relevant articles. What these search engines do not tell us is the optimal order in which we should read through these articles. Computationally optimizing the ordering over which one reads through a set of articles can be thought of as an automated way for *designing a curriculum*, and by doing so, we may be able to improve the speed at which a student acquires new knowledge, unlocking his or her true potential.

I am excited about the possibilities of optimizing curricula for *personalized education*. Every person has a unique style of learning and comes to the "classroom" with different backgrounds and biases. To succeed as curriculum designers, we should create curricula *personally tailored to each individual*.

A promising starting point will be to exploit the "wisdom of the crowds" phenomenon, in which we collect data about the order in which people read through articles on a certain topic. By modeling such a dataset, we can make predictions and form recommendations. There are a number of possible data sources. For example, with the rising costs of tuition and textbooks, many college students are turning to online distance learning programs [22]. I believe these programs to not only be a valuable source of user interaction data, but a setting in which we could experiment with different curricula and have a real impact in education by eliminating frustration in distance learning and decreasing the number of people who leave these programs.

18.3.2 *New combinatorial and algebraic spaces*

MAPPINGS BETWEEN SHAPES The problem of finding a mapping between two geometric deformable shapes (in order, for example to align, or register the shapes) arises in a number of computational situations such as computer graphics and computer vision. But there has been relatively little work on efficient and principled *probabilistic reasoning* with shape registration problems, which, if solved, would allow us to deal more effectively with noise, missing data, or model misspecification.

In many ways, probabilistic shape mapping is much like any problem involving distributions over permutations, which are themselves a type of mapping (though without geometric features such as curvature or smoothness), and I believe that many of the ideas from my thesis work may generalize to this geometric setting.

Fourier or riffled independence structure may not be sufficient for shape matching, and to develop realistic probabilistic models which allow for efficient inference may require insights from both computational geometry and machine learning. I look forward to collaborating with computational geometers and topologists to formulate probabilistic reasoning problems in geometric spaces.

JOINT MAPPINGS AMONG A COLLECTION OF SPACES. Going beyond mappings between just two shapes, it is useful to consider *mappings between an entire collection of spaces* which allow one to exploit the statistical strength of multiple mappings and compositions thereof to draw better informed inferences. These ideas may have impact in fields beyond geometry. For example, there have been similar ideas in machine translation in which one computes a 'mapping' between sentences of two languages. For some language pairs, such as English-to-Catalan, however, there do not exist many parallel corpora which can be used for training. But, there do exist parallel corpora between Catalan and its neighbors, Spanish and French, and therefore we might hope to obtain superior translations by leveraging the statistical power of multiple translations with intermediary languages, such as English-to-Spanish-to-Catalan and English-to-French-to-Catalan. Little work exists about performing principled probabilistic inference and formulating statistical guarantees for this problem of reasoning over mappings between collections of spaces. How do we algorithmically incorporate information from multiple mappings to inform our inferences/predictions? What kinds of realistic structural assumptions can be used to make probabilistic inference efficient? Can we establish statistical guarantees of accuracy by leveraging multiple mappings? Progress on these problems will lead to advances for both computational geometry as well as machine learning, and allow us to efficiently deal with uncertainty in problems which require spatial reasoning such as robotic manipulation and 3d content creation.

18.4 LAST WORDS

The work in this thesis on permutations is theoretically grounded and has drawn from diverse technical areas such as graphical models, statistics, algorithms, signal processing, group theory, combinatorics and optimization theory. I believe that many of these same fields would likewise be interested in different contributions from this thesis.

I look forward to combining further insights from all these fields and others, as well as collaborating with colleagues from different fields, such as HCI and Education, in order to make progress in my research program.

The steps outlined above represent just the beginnings of my vision of such a program. Advances on these fronts will have implications on education, social sciences, cognitive psychology, and web sciences, and I am excited to see where this progress will lead.

Part V

APPENDIX

A

GROUPS

THIS section is intended as a quick glossary for the basic group theoretic definitions and facts used in this thesis.

A.1 GROUP AXIOMS

Groups are a generalization of many of the spaces that we typically work with, such as the real numbers, integers, vector spaces, and matrices. The definition of a group unifies all of these spaces under a handful of axioms.

Definition 130 (Group)**.** A *group* is a set G together with a binary operation $\cdot : G \times G \to G$ (called the *group operation*) such that the following *group axioms* hold:

1. (Associativity) The group operation is *associative*. That is, for any group elements $g_1, g_2, g_3 \in G$, we have:

$$(g_1 \cdot g_2) \cdot g_3 = g_1 \cdot (g_2 \cdot g_3), \text{ for all } g_1, g_2, g_3 \in G.$$

2. (Identity) There exists an *identity* element (denoted by ϵ) such that $g \cdot \epsilon = \epsilon \cdot g = g$ for any $g \in G$.
3. (Inverses) For every $g \in G$, there exists an *inverse element* g^{-1} such that $g \cdot g^{-1} = g^{-1} \cdot g = \epsilon$.

Definition 131 (Abelian Group)**.** If $g_1 \cdot g_2 = g_2 \cdot g_1$ holds for all $g_1, g_2 \in G$, then G is called an *Abelian* or *commutative* group.

Perhaps the most familiar group is the set of integers, $\mathbb{Z}$, with respect to the addition operation. It is well known that for any integers $a, b, c \in \mathbb{Z}$, $a + (b + c) = (a + b) + c$. The identity element in the integers is zero, and every element has an additive inverse ($a + (-a) = (-a) + a = 0$). Additionally, the integers are an Abelian group since $a + b = b + a$ for any $a, b \in \mathbb{Z}$. Note that the natural numbers $\mathbb{N} = \{0, 1, 2, 3, \dots\}$ do not form a group with respect to addition because inverses do not exist.

The main example of a group in this thesis, of course, is the symmetric group, the set of permutations of $\{1, \dots, n\}$. The group operation on permutations is function composition, which is associative, and we discuss inverses and the identity element in Section 2.

Example 132. *There are many groups besides the integers and the symmetric group. The following are several examples.*

- *The positive real numbers $\mathbb{R}^+$ form a group with respect to multiplication. The identity element of $\mathbb{R}^+$ is the multiplicative identity, 1, and given a real number x, there exists an inverse element $\frac{1}{x}$.*

- *As an example of a finite group, the* integers modulo n, $\mathbb{Z}/n\mathbb{Z} = \{0, 1, \ldots, n-1\}$, *form an Abelian group with respect to addition modulo* n. *For example,* $\mathbb{Z}/6\mathbb{Z}$ *has 6 elements,* $\{0, \ldots, 5\}$, *and* $4 + 5 = 3 \pmod 6$.
- *The invertible* $n \times n$ *matrices over the reals,* $GL_n(\mathbb{R})$, *form a group with respect to matrix multiplication. The* $n \times n$ *identity matrix serves as the identity element in* $GL_n(\mathbb{R})$, *and by assumption, every matrix in* $GL_n(\mathbb{R})$ *is invertible.*

A.2 SUBGROUPS AND COSETS

The group axioms impose strong structural constraints on G, and one of the ways that structure is manifested in groups is in the existence of *subgroups*.

Definition 133 (Subgroup)**.** If G is a group (with group operation $\cdot$), a subset $H \subset G$ is called a *subgroup* if it is itself a group with respect to the same group operation. H is called a *trivial subgroup* if it is either all of G or consists only of a single element.

Example 134. *We have the following examples of subgroups.*

- *The even integers,* $2\mathbb{Z}$, *form a subgroup of the integers since the sum of any two even integers is an even integer, and the inverse (negative) of an even integer is again even. However, the odd integers do* not *form a subgroup since the sum of two odd integers is not odd.*
- *The special orthogonal matrices (orthogonal matrices with determinant* $+1$*) form a subgroup of the group of* $n \times n$ *matrices,* $GL_n(\mathbb{R})$. *This can be seen by using the facts (1), that* $(\det A)(\det B) = \det(AB)$ *and (2), that the inverse of any orthogonal matrix is also orthogonal.*

Given a subgroup, we can partition a group into disjoint translations of that subgroup. These translations are known as cosets.

Definition 135 (Coset)**.** Let H be any subgroup of G. A *left* H*-coset* in G is any subset of the form:

$$gH = \{g \cdot h \ : \text{ such that } h \in H\}, \text{ for some } g \in G.$$

Similarly, a *right* H*-coset* in G is any subset of the form:

$$Hg = \{h \cdot g \ : \text{ such that } h \in H\}, \text{ for some } g \in G.$$

Example 136. *The group* $\mathbb{Z}/6\mathbb{Z}$ *contains a subgroup* H *consisting of elements* $\{0, 3\}$. *The cosets of* H *are:* $0 + H = \{0, 3\}$, $1 + H = \{1, 4\}$ *and* $2 + H = \{2, 5\}$, *which together form a disjoint partition of the entire group* $\mathbb{Z}/6\mathbb{Z}$. *Since* $\mathbb{Z}/6\mathbb{Z}$ *is an Abelian group, left and right cosets are the same, but in general they need not be.*

Because of the way H-cosets can disjointly partition a group, the cardinality of a finite group G is always divisible by the cardinality of any subgroup H. In the above example, $|\mathbb{Z}/6\mathbb{Z}| = 6$ and $|H| = 2$; The number of H-cosets covering G (also known as the *group index*) is therefore $6/2 = 3$.

Proposition 137 (Lagrange's Theorem)**.** $|H|$ *divides* $|G|$ *for any finite group* G *and subgroup* $H \subset G$,

A.3 GROUP HOMOMORPHISMS AND ISOMORPHISMS

We would like to know if two groups G and H are the "same" up to a relabeling of elements, or at least if they share some form of algebraic structure. This notion of 'sameness' is formalized via *homomorphisms* and *isomorphisms*, maps between two groups which preserve the group operation structure.

Definition 138 (Homomorphisms and Isomorphisms)**.** Let $(G, \cdot)$ and $(H, \star)$ be any groups. A function $\phi : G \to H$ is called a *group homomorphism* if for any $g_1, g_2 \in G$, $\phi(g_1 \cdot g_2) = \phi(g_1) \star \phi(g_2)$.

If ϕ is also bijective, then we say that ϕ is a *group isomorphism*.

Example 139.

- ($\phi : \mathbb{Z} \times \mathbb{Z} \to \mathbb{Z}$)*: Define* $\phi((x_1, x_2)) = x_1$*. The map* ϕ*, which projects to the first coordinate, is a homomorphism since* $\phi((x_1, x_2)) + \phi((y_1, y_2)) = x_1 + y_1 = \phi((x_1 + x_1, x_2 + y_2))$.
- ($\phi : \mathbb{Z}/3\mathbb{Z} \to S_3$)*: Define* $\phi(0) = [1\,2\,3]$, $\phi(1) = [2\,3\,1]$*, and* $\phi(2) = [3\,1\,2]$*.* ϕ *can be checked to be a homomorphism. For example,* $\phi(1)\phi(2) = [2\,3\,1][3\,1\,2] = [1\,2\,3] = \phi(0) = \phi(1+2)$.
- (Group representations, $\phi : G \to \mathbb{C}^{d_\rho \times d_\rho}$)*: Notice that the definition of a group representation 20, can be rephrased simply as any homomorphism from a group* G *to* $\mathbb{C}^{d_\rho \times d_\rho}$.

A.4 GROUP ACTIONS

Group theory is often used to study the symmetries of a space or set. These 'symmetries' are often encoded via *group actions*.

Definition 140 (Group action)**.** Let G be any group and X any set. A *group action* of G on S is an operation $\cdot : G \times X \to X$ such that

- For all $g_1, g_2 \in G$ and $x \in X$, $g_1 \cdot (g_2 \cdot x) = (g_1 \cdot g_2) \cdot x$, and
- For the identity element $\epsilon \in G$, $\epsilon \cdot x = x$ for all $x \in X$.

More abstractly (though equivalently), one can view a group action as any group homomorphism $\psi : G \to S_X$ from G to the permutation group of X.

Example 141.

- (Trivial group action)*: The* trivial group action *maps every element of* X *to itself (i.e.,* $g \cdot x = x$ *for all* $x \in X$*).*
- (S_n acting on $\{1, \ldots, n\}$)*: The symmetric group (on* n *items) acts on the set* $\{1, \ldots, n\}$ *via the (obvious) operation* $\sigma \cdot x = \sigma(x)$.
- ($\mathbb{Z}/3\mathbb{Z}$ on $\mathbb{R}^2$)*: The cyclic group of order 3 (*$\mathbb{Z}/3\mathbb{Z}$*) acts on the two dimensional plane via 120°counter-clockwise rotations. In particular, given an element* $(x_1, x_2) \in \mathbb{R}^2$*, we define* $g \cdot (x_1, x_2)$ *via:*

$$\begin{bmatrix} \cos(2\pi/3) & -\sin(2\pi/3) \\ \sin(2\pi/3) & \cos(2\pi/3) \end{bmatrix}^g \begin{bmatrix} x_1 \\ x_2 \end{bmatrix} = \begin{bmatrix} -1/2 & -\sqrt{3}/2 \\ -\sqrt{3}/2 & -1/2 \end{bmatrix}^g \begin{bmatrix} x_1 \\ x_2 \end{bmatrix},$$

for each $g \in \{0, 1, 2\} = \mathbb{Z}/3\mathbb{Z}$.

- (Group representation acting on a vector space)*: Generalizing the previous example, note that any group representation* $\rho : G \to \mathbb{C}^{d_\rho \times d_\rho}$ *acts on the vector space* $\mathbb{C}^{d_\rho}$ *by defining the group action* $g \cdot x$ *via* $g \cdot x = \rho(g)x$.

Finally, the collection of elements in the group G which *fix* an element of X forms a subgroup of G. We call this subgroup a *stabilizer* or *isotropy subgroup*.

Definition 142 (Stabilizer subgroup)**.** Given an element $x \in X$, and a group action of G on X, the set

$$G_x = \{g \in G \,:\, g \cdot x = x\}$$

forms a subgroup of G, which we call the *stabilizer* or *isotropy subgroup* of x.

B

CLEBSCH-GORDAN SERIES

In this appendix, we turn to the *Tensor Product Decomposition* problem, which is that of finding the irreducible components of the typically reducible tensor product representation. If ρ_λ and ρ_μ are irreducible representations of S_n, then there exists an intertwining operator $C_{\lambda\mu}$ such that:

$$C_{\lambda\mu}{}^{-1} \cdot (\rho_\lambda \otimes \rho_\mu(\sigma)) \cdot C_{\lambda\mu} = \bigoplus_{\nu} \bigoplus_{\ell=1}^{z_{\lambda\mu\nu}} \rho_\nu(\sigma). \tag{B.1}$$

We present two methods for computing the Clebsch-Gordan series ($z_{\lambda\mu\nu}$) for a pair of irreducible representations $\rho_\lambda \otimes \rho_\mu$ (we defer the problem of computing the Clebsch-Gordan coefficient matrices for Appendix D). The results of this appendix are specific to the symmetric group.

B.1 DECOMPOSING REPRESENTATIONS VIA CHARACTER THEORY

We begin with a simple, well-known algorithm based on *group characters* for computing the Clebsch-Gordan series that turns out to be computationally intractable, but yields several illuminating theoretical results. See [117] for proofs of the theoretical results cited in this section.

One of the main results of representation theory was the discovery that there exists a relatively compact way of encoding any representation up to equivalence with a vector which we call the *character* of the representation. If ρ is a representation of a group G, then the character of the representation ρ, is defined simply to be the trace of the representation at each element $\sigma \in G$:

$$\chi_\rho(\sigma) = \mathrm{Tr}\left(\rho(\sigma)\right).$$

The reason characters have been so extensively studied is that they uniquely characterize a representation up to equivalence in the sense that two characters χ_{ρ_1} and χ_{ρ_2} are equal if and only if ρ_1 and ρ_2 are equivalent as representations. Even more surprising is that the space of possible group characters is orthogonally spanned by the characters of the irreducible representations. To make this precise, we first define an inner product on functions from G.

Definition 143. Let ϕ, ψ be two real-valued functions on G. The *inner product* of ϕ and ψ is defined to be:

$$\langle \phi, \psi \rangle \equiv \frac{1}{|G|} \sum_{\sigma \in G} \phi(\sigma)\psi(\sigma).$$

With respect to the above inner product, we have the following important result which allows us to test a given representation for irreducibility, and to test two irreducibles for equivalence.

Proposition 144. *Let χ_{ρ_1} and χ_{ρ_2} be characters corresponding to irreducible representations. Then*

$$\langle \chi_{\rho_1}, \chi_{\rho_2} \rangle = \begin{cases} 1 & \text{if } \rho_1 \equiv \rho_2 \\ 0 & \text{otherwise} \end{cases}.$$

Proposition 144 shows that the irreducible characters form an orthonormal set of functions. The next proposition says that the irreducible characters *span* the space of all possible characters.

Proposition 145. *Suppose ρ is any representation of G and which decomposes into irreducibles as:*

$$\rho \equiv \bigoplus_{\lambda} \bigoplus_{\ell=1}^{z_\lambda} \rho_\lambda,$$

where λ indexes over all irreducibles of G. Then:

1. *The character of ρ is a linear combination of irreducible characters ($\chi_\rho = \sum_\lambda z_\lambda \chi_{\rho_\lambda}$),*
2. *and the multiplicity of each irreducible, z_λ, can be recovered using $\langle \chi_\rho, \chi_{\rho_\lambda} \rangle = z_\lambda$.*

A simple way to decompose any group representation ρ, is given by Proposition 145, which says that we can take inner products of χ_ρ against the basis of irreducible characters to obtain the irreducible multiplicities z_λ. To treat the special case of finding the Clebsch-Gordan series, one observes that the character of the tensor product is simply the pointwise product of the characters of each tensor product factor.

Theorem 146. *Let ρ_λ and ρ_μ be irreducible representations with characters χ_λ, χ_μ respectively. Let $z_{\lambda\mu\nu}$ be the number of copies of ρ_ν in $\rho_\lambda \otimes \rho_\mu$ (hence, one term of the Clebsch-Gordan series). Then:*

1. *The character of the tensor product representation is given by:*

$$\chi_{\rho_\lambda \otimes \rho_\mu} = \chi_\lambda \cdot \chi_\mu = \sum_\nu z_{\lambda\mu\nu} \chi_\nu. \tag{B.2}$$

2. *The terms of the Clebsch-Gordan series can be computed using:*

$$z_{\lambda\mu\nu} = \frac{1}{|G|} \sum_{g \in G} \chi_\lambda(g) \cdot \chi_\mu(g) \cdot \chi_\nu(g), \tag{B.3}$$

and satisfy the following symmetry:

$$z_{\lambda\mu\nu} = z_{\lambda\nu\mu} = z_{\mu\lambda\nu} = z_{\mu\nu\lambda} = z_{\nu\lambda\mu} = z_{\nu\mu\lambda}. \tag{B.4}$$

Dot products for characters on the symmetric group can be done in $O(\#(n))$ time where $\#(n)$ is the number of partitions of the number n, instead of the naive $O(n!)$ time. In practice however, $\#(n)$ also grows too quickly for the character method to be tractable.

B.1.1 *Murnaghan's formulas for tensor product decompositions*

To handle larger problems, we turn to a theorem by Murnaghan [102], which gives us a 'bound' on which representations can appear in the tensor product decomposition on S_n.

Theorem 147. *Let ρ_1, ρ_2 be the irreducibles corresponding to the partition $(n-p, \lambda_2, \dots)$ and $(n-q, \mu_2, \dots)$ respectively. Then the product $\rho_1 \otimes \rho_2$ does not contain any irreducibles corresponding to a partition whose first term is less than $n-p-q$.*

In view of the connection between the Clebsch-Gordan series and convolution of Fourier coefficients, Theorem 147 is analogous to the fact that for functions over the reals, the convolution of two compactly supported functions is also compactly supported.

We can use Theorem 147 to show that Kronecker conditioning is exact at certain irreducibles. Below we restate the theorem and provide its proof.

Theorem 148. *If $\lambda^{\text{MIN}} = (n-p, \lambda_2, \dots)$, and the Kronecker conditioning algorithm is called with a likelihood function whose Fourier coefficients are nonzero only at ρ_μ when $\mu \trianglerighteq (n-q, \mu_2, \dots)$, then the approximate Fourier coefficients of the posterior distribution are exact at the set of irreducibles:*

$$\Lambda_{\text{EXACT}} = \{\rho_\lambda \ : \ \lambda \trianglerighteq (n - |p-q|, \dots)\}.$$

Proof. (of Theorem 60) Let Λ denote the set of irreducibles at which our algorithm maintains Fourier coefficients. Since the errors in the prior come from setting coefficients outside of Λ to be zero, we see that Kronecker conditioning returns an approximate posterior which is exact at the irreducibles in

$$\Lambda_{\text{EXACT}} = \{\rho_\nu \ : \ z_{\lambda\mu\nu} = 0, \text{ where } \lambda \notin \Lambda \text{ and } \mu \trianglerighteq (n-q, \mu_2, \dots)\}.$$

Combining Theorem 147 with Equation B.4: if $z_{\lambda\mu\nu} > 0$, with $\lambda = (n-p, \lambda_2, \lambda_3, \dots)$, $\mu = (n-q, \mu_2, \mu_3, \dots)$ and $\nu = (n-r, \nu_2, \nu_3, \dots)$, then we have that: $r \leqslant p+q, p \leqslant q+r$, and $q \leqslant p+r$. In particular, it implies that $r \geqslant p-q$ and $r \geqslant q-p$, or more succinctly, $r \geqslant |p-q|$. Hence, if $\nu = (n-r, \nu_2, \dots)$, then $\rho_\nu \in \Lambda_{\text{EXACT}}$ whenever $r \leqslant |p-q|$, which proves the desired result. □

The same paper [102] derives several general Clebsch-Gordan series formulas for pairs of low-order irreducibles in terms of n, and in particular, derives the Clebsch-Gordan series for many of the Kronecker product pairs that one would likely encounter in practice. For example, we have:

$$\begin{aligned}
\rho_{(n-1,1)} \otimes \rho_{(n-1,1)} &\equiv \rho_{(n)} \oplus \rho_{(n-1,1)} \oplus \rho_{(n-2,2)} \oplus \rho_{(n-2,1,1)}, \\
\rho_{(n-1,1)} \otimes \rho_{(n-2,2)} &\equiv \rho_{(n-1,1)} \oplus \rho_{(n-2,2)} \oplus \rho_{(n-2,1,1)} \\
&\qquad \oplus \rho_{(n-3,3)} \oplus \rho_{(n-3,2,1)}, \\
\rho_{(n-1,1)} \otimes \rho_{(n-2,1,1)} &\equiv \rho_{(n-1,1)} \oplus \rho_{(n-2,2)} \oplus \rho_{(n-2,1,1)} \\
&\qquad \oplus \rho_{(n-3,2,1)} \oplus \rho_{(n-3,1,1,1)},
\end{aligned}$$

$$\rho_{(n-1,1)} \otimes \rho_{(n-3,3)} \equiv \rho_{(n-2,2)} \oplus \rho_{(n-3,3)} \oplus \rho_{(n-3,2,1)} \\ \oplus \rho_{(n-4,4)} \oplus \rho_{(n-4,3,1)}.$$

C

LITTLEWOOD-RICHARDSON COEFFICIENTS

In this appendix, we present the Littlewood-Richardson rule for computing the $c^{\lambda}_{\mu,\nu}$ coefficients, as well as proofs for Chapter 12.

C.1 DEFINITIONS

Let λ be a partition of n and let p and q be positive integers such that $p+q=n$. If ρ_λ is any irreducible representation of S_n, then, restricted to permutations which lie in the subgroup $S_p \times S_q \subset S_n$, ρ_λ splits according to Equation 5.4 as a direct sum of irreducibles of $S_p \times S_q$ which take the form $\rho_\mu \otimes \rho_\nu$ (where μ and ν are partitions of p and q respectively):

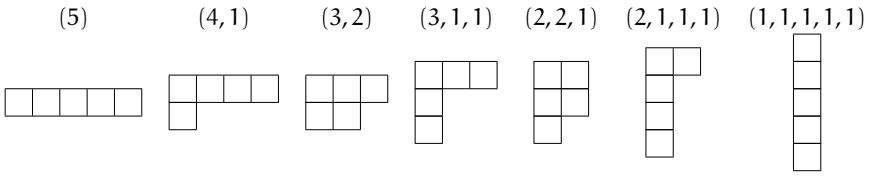
$$L^{\lambda}_{\mu\nu} \cdot \rho_\lambda(\sigma) \cdot {L^{\lambda}_{\mu\nu}}^{\mathsf{T}} \equiv \bigoplus_{\mu,\nu} \bigoplus_{\ell=1}^{c^{\lambda}_{\mu,\nu}} \rho_\mu(\sigma_p) \otimes \rho_\nu(\sigma_q), \text{ for all } \sigma = [\sigma_p, \sigma_q] \in S_p \times S_q. \tag{C.1}$$

As a shorthand, we will also write the decomposition as: $\rho_\lambda \downarrow_{S_p \times S_q} \equiv \bigoplus_{\mu,\nu} \bigoplus_{\ell=1}^{c^{\lambda}_{\mu,\nu}} \rho_\mu \otimes \rho_\nu$, where we have suppressed the coupling matrices $L^{\lambda}_{\mu\nu}$, and the $\downarrow_{S_p \times S_q}$ symbol (called the *restriction operator*) means that the decomposition holds only for elements of $S_p \times S_q$ even if the representation ρ_λ is defined over all of S_n. The multiplicities in the decomposition (Equation C.1) are famously known as the *Littlewood-Richardson coefficients*.[1] In this appendix, we describe a result known as the *Littlewood-Richardson (LR) rule* which will allow us to compute the Littlewood-Richardson coefficients tractably (at least for low-order terms). There are several methods for computing these numbers (see [73, 130], for example) but it is known [104] that, in general, the problem of computing the Littlewood-Richardson coefficients is #P-hard.

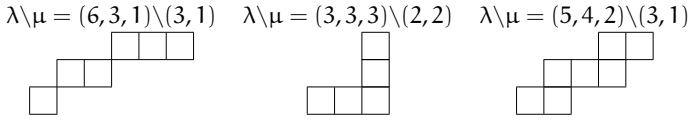
The statement of the LR rule requires us to define a class of (rather complex) combinatorial objects known as the *Littlewood-Richardson tableaux*, which will be used to count the LR coefficients. We proceed by defining the LR tableaux in several stages.

- *(Ferrers diagrams)* We can visualize a partition λ, of n, using a *Ferrers diagram* which is an arrow of boxes with λ_i boxes in the i^{th} row of the associated Ferrers diagram. For example, we have the following partitions of $n=5$ and their respective Ferrers diagrams.

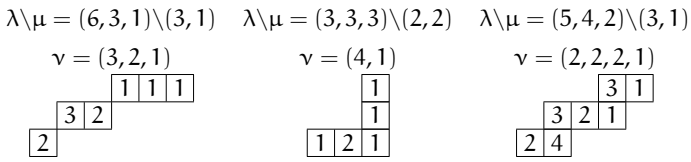
1 In most texts, the Littlewood-Richardson coefficients are defined in a slightly different way using induced representations (see [114]).



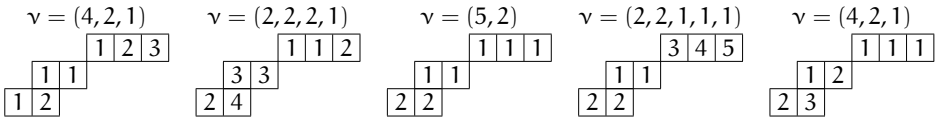
- *(Skew tableaux)* Let λ be a partition of n and let μ be a partition of some $p \leqslant n$ such that $\mu_i \leqslant \lambda_i$ for each i. A *skew tableau* with shape $\lambda\backslash\mu$ is the diagram obtained by removing all boxes of the Ferrers diagram of λ which also belong to the Ferrers diagram of μ. The following are a few examples of skew tableaux and their corresponding shapes.



- *(Content)* As before, we will consider λ to be a partition of n and μ to be a partition of some $p \leqslant n$. Additionally, let ν be a partition of $q = n - p$. We say that a skew tableaux of shape $\lambda\backslash\mu$ has *content* $\nu = (\nu_1, \nu_2, \nu_3, \dots)$ if its boxes are filled in with ν_1 ones, ν_2 twos, ν_3 threes, and so on. To extend the previous example, we have:



- *(Semistandard tableaux)* We say that a skew tableau with shape $\lambda\backslash\mu$ and content ν is *semistandard* if its rows are *weakly increasing* (reading from left to right) and its columns are *strictly increasing* (reading from top to bottom). For example, the following are semistandard tableaux with shape $(6, 3, 2)\backslash(3, 1)$:

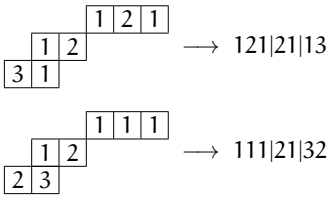


While the following are *invalid* as semistandard tableaux:

(row 1 not weakly increasing) (column 2 not strictly increasing)

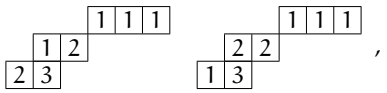
- *(Reverse lattice word constraint)* A word $w_1 \dots w_r$ is said to be a *lattice word* if, for each $s \leqslant r$, the subsequence $w_1 \dots w_s$ contains at least as many ones as twos, at least as many twos as threes, and so on. For example, 11123211 and 12312111 are lattice words while 1114 and 12321111 are not (in the first case because there are more fours than threes in 1114, and in the second case because there are more twos than ones in the subsequence 1232).

  A skew tableau is said to satisfy the *reverse lattice word constraint* if a lattice word is obtained by reading its entries from top to bottom and from right to left (as in Hebrew). The following are two examples for skew tableaux satisfying the reverse lattice word constraint.

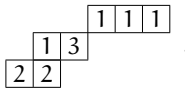


Definition 149. A skew tableaux with shape $\lambda\backslash\mu$ and content ν which is semistandard and satisfies the reverse lattice word constraint is called a *Littlewood-Richardson tableau*.

As an example, the following are the two valid Littlewood-Richardson tableaux with shape $\lambda\backslash\mu = (6,3,2)\backslash(3,1)$ and content $\nu = (4,2,1)$:



while the following tableau is invalid as a Littlewood-Richardson tableau since it does not satisfy the reverse lattice word constraint:



We conclude that $c^{\lambda}_{\mu\nu} = 2$.

C.2 THE LITTLEWOOD RICHARDSON RULE

Theorem 150 (Littlewood-Richardson rule)**.** *The Littlewood-Richardson coefficient, $c^{\lambda}_{\mu,\nu}$, is equal to the number of Littlewood-Richardson tableaux with shape $\lambda\backslash\mu$ and content ν.*

Proof. See [114, 65], for example. □

Proposition 151. *Consider $n \geqslant 2$ and any positive integers p, q such that $p + q = n$. Then the following decomposition holds:*

$$\rho_{(n-1,1)} \downarrow_{S_p \times S_q} \equiv (\rho_{(p)} \otimes \rho_{(q)}) \oplus (\rho_{(p-1,1)} \otimes \rho_{(q)}) \oplus (\rho_{(p)} \otimes \rho_{(q-1,1)}).$$

Proof. We first claim that the topmost row of any Littlewood-Richardson tableau is filled only with ones. To see why, notice that the reverse lattice word constraint forces the rightmost label of the topmost row to be labeled with a one. The semistandard constraint (that ensures that rows are weakly increasing) then forces all remaining entries to also be labeled one.

We now enumerate the possible Littlewood-Richardson tableaux for $\lambda = (n-1,1)$. We consider two cases: when $\mu = (p-1,1)$ and when $\mu = (p)$.

- If $\mu = (p-1,1)$, then $\lambda\backslash\mu$ consists of a single row. By our claim above, there must be only one Littlewood-Richardson tableau consistent with $\lambda\backslash\mu$ — the one labeled with all ones (i.e., with content $\nu = (q)$). Therefore we have:

$$c^{\lambda}_{\mu,\nu} = \begin{cases} 1 & \text{if } \nu = (q) \\ 0 & \text{if } \nu = (q-1,1) \\ 0 & \text{otherwise} \end{cases} .$$

- If $\mu = (p)$, then $\lambda\backslash\mu$ consists of two rows with no shared columns since we assumed $n > 2$. As before, the first row of $\lambda\backslash\mu$ must be filled with ones. The second row (consisting of just one box) can be filled with either a one or a two, yielding two possible content vectors: $\nu = (q)$ or $\nu = (q-1,1)$. Therefore we have:

$$c^{\lambda}_{\mu,\nu} = \begin{cases} 1 & \text{if } \nu = (q) \text{ or } (q-1,1) \end{cases} .$$

Collecting the triplets for which $c^{\lambda}_{\mu,\nu} = 1$ and using Theorem 150 shows that

$$\rho_{(n-1,1)} \downarrow_{S_p \times S_q} \equiv (\rho_{(p)} \otimes \rho_{(q)}) \oplus (\rho_{(p-1,1)} \otimes \rho_{(q)}) \oplus (\rho_{(p)} \otimes \rho_{(q-1,1)}).$$

□

A similar (but more involved) analysis can be applied to decompose the second order irreducibles, which we summarize below (proofs omitted).

Proposition 152. *Let $n \geqslant 2$ and p, q be any positive integers such that $p + q = n$. Then the following decomposition holds:*

$$\begin{aligned} \rho_{(n-2,2)} \downarrow_{S_p \times S_q} \equiv & (\rho_{(p)} \otimes \rho_{(q)}) \oplus (\rho_{(p)} \otimes \rho_{(q-1,1)}) \oplus (\rho_{(p)} \otimes \rho_{(q-2,2)}) \oplus \\ & (\rho_{(p-1,1)} \otimes \rho_{(q)}) \oplus (\rho_{(p-1,1)} \otimes \rho_{(q-1,1)}) \oplus \\ & (\rho_{(p-2,2)} \otimes \rho_{(q)}), \end{aligned}$$

except in the following exceptional boundary cases for p (the same rules apply for q):

- *($p = 3$) If $p = 3$, remove the term $\rho_{(p-2,2)} \otimes \rho_{(q)}$.*
- *($p = 2$) If $p = 2$, remove the terms $\rho_{(p-1,1)} \otimes \rho_{(q)}$ and $\rho_{(p-2,2)} \otimes \rho_{(q)}$.*
- *($p = 1$) If $p = 1$, we recover the* branching rule *[114] and remove all terms except $\rho_{(p)} \otimes \rho_{(q-1,1)}$ and $\rho_{(p)} \otimes \rho_{(q-2,2)}$.*

Proposition 153. *Let $n \geqslant 2$ and p, q be any positive integers such that $p + q = n$. Then the following decomposition holds:*

$$\begin{aligned}\rho_{(n-2,1,1)} \downarrow_{S_p \times S_q} \equiv \quad & (\rho_{(p)} \otimes \rho_{(q-1,1)}) \oplus (\rho_{(p)} \otimes \rho_{(q-2,1,1)}) \oplus \\ & (\rho_{(p-1,1)} \otimes \rho_{(q)}) \oplus (\rho_{(p-1,1)} \otimes \rho_{(q-1,1)}) \oplus \\ & (\rho_{(p-2,1,1)} \otimes \rho_{(q)}),\end{aligned}$$

except in the following exceptional boundary cases for p (the same rules apply for q):

- *($p = 2$) If $p = 2$, remove the term $\rho_{(p-2,1,1)} \otimes \rho_{(q)}$.*
- *($p = 1$) If $p = 1$, we recover the* branching rule *and remove all terms except $\rho_{(p)} \otimes \rho_{(q-1,1)}$ and $\rho_{(p)} \otimes \rho_{(q-2,2)}$.*

C.3 MARGINAL PRESERVATION GUARANTEES

We now state a few properties of partitions and Littlewood-Richardson coefficients in order to prove the Join and Split guarantees.

Definition 154. Let $\lambda = (\lambda_1, \ldots, \lambda_\ell)$ be a partition of n. The *height* of λ is defined to be ℓ.

Definition 155. Let λ be a partition of n and μ a partition of $p \leqslant n$. We say that μ is a *subpartition* of λ if for every i, $\mu_i \leqslant \lambda_i$.

Lemma 156. *Define the partition:*

$$\lambda_s^{\text{MIN}} = (n - s, \underbrace{1, \ldots, 1}_{s \text{ times}}),$$

for some $0 \leqslant s < n$, and the set $\Lambda_s = \{\mu \ : \ \mu = (n - r, \ldots) \text{ for some } r \leqslant s.\}$. The following three statements are equivalent.

1. $\mu \trianglerighteq \lambda_s^{\text{MIN}}$.
2. $\mu \in \Lambda_s$.
3. $\text{height}(\mu) \leqslant \text{height}(\lambda_s^{\text{MIN}}) = s + 1$.

Lemma 157. *Define the partitions:*

$$\lambda^{\text{MIN}} = (n - s, \underbrace{1, \ldots, 1}_{s \text{ times}}), \qquad \mu^{\text{MIN}} = (p - k, \underbrace{1, \ldots, 1}_{k \text{ times}}), \tag{C.2}$$

where $k = \min(s, p - 1)$. If λ is any partition of n such that $\lambda \trianglerighteq \lambda^{\text{MIN}}$, then for any partition μ of p which is also a subpartition of λ, we have $\mu \trianglerighteq \mu^{\text{MIN}}$.

Proof. By Lemma 156, $\text{height}(\lambda) \leqslant \text{height}(\lambda^{\text{MIN}}) = s + 1$. But since μ is a subpartition of λ, we also have that $\text{height}(\mu) \leqslant \text{height}(\lambda)$, and since μ is a partition of p, $\text{height}(\mu) \leqslant p$. Putting these inequalities together, we see that $\text{height}(\mu) \leqslant \min(p, s + 1)$. Finally,

$$\text{height}(\mu^{\text{MIN}}) = k + 1 = \min(s, p - 1) + 1 = \min(p, s + 1),$$

showing that $\text{height}(\mu) \leqslant \text{height}(\mu^{\text{MIN}})$. By Lemma 156 again, we conclude that $\mu \trianglerighteq \mu^{\text{MIN}}$. □

Lemma 158. *The LR coefficient* $c^{\lambda}_{\mu\nu} = 0$ *unless both* μ *and* ν *are subpartitions of* λ.

Proof. This fact follows directly from the Littlewood-Richardson rule [114]. □

Corollary 159. *The set of Fourier coefficients:*

$$\{\widehat{f}_\mu, \widehat{g}_\nu \ : \ \mu \text{ is a partition of } p, \nu \text{ is a partition of } q, \text{and } \mu, \nu \text{ are both subpartitions of } \lambda\},$$

is sufficient for constructing $\widehat{h}_\lambda$ *for any partition* λ, *of* $n = p + q$.

Proof of Theorem 68. We need to be able to construct $\widehat{h}_\lambda$ at all partitions λ such that $\lambda \trianglerighteq \lambda^{MIN}$. By Corollary 159, we need subpartitions μ and ν of λ at all $\lambda \trianglerighteq \lambda^{MIN}$, but by Lemma 157, all such subpartitions are above μ^{MIN} and ν^{MIN} with respect to the dominance ordering, respectively. ■

Proof of Theorem 69. Let $\mu = (\mu_1, \mu_2, \ldots, \mu_\ell)$ be any partition of p such that $\mu \trianglerighteq \mu^{MIN}$. By Lemma 156, we have that $\text{height}(\mu) \leqslant \text{height}(\mu^{MIN}) = \min(p, s+1)$. Define the partition $\tilde{\mu} = (\mu_1 + n - p, \mu_2, \ldots, \mu_\ell)$. Two things are immediate: first, μ is a subpartition of $\tilde{\mu}$, and second, since μ is a partition of p, $\tilde{\mu}$ is a partition of n. We also have, by Lemma 156 again, that

$$\text{height}(\tilde{\mu}) = \text{height}(\mu) \leqslant \min(p, s+1) \leqslant s+1 = \text{height}(\lambda^{MIN}),$$

and therefore it must be the case that $\tilde{\mu} \trianglerighteq \lambda^{MIN}$. Finally, we have that $c^{\tilde{\mu}}_{\mu,(q)} = 1$ exactly. ■

C.3.1 *Lexicographical order preservation guarantees*

Recall that lexicographical comparisons induce a total ordering on the set of partitions of n, and we will denote the index of a particular partition λ in the lex ordering by $\text{LEX}_n(\lambda)$. For example, the lex ordering for partitions of $n = 4$ is given by:

$\text{LEX}_n(\lambda)$	1	2	3	4	5
λ	(4)	(3,1)	(2,2)	(2,1,1)	(1,1,1,1)

It is known that the lexicographical ordering is a refinement of the dominance ordering.

Proposition 160. *Consider partitions* μ, ν *and* λ *of* p, q, *and* n, *respectively (where* $p + q = n$*). If* $\text{LEX}_n(\lambda) < \text{LEX}_p(\mu)$ *(or similarly, if* $\text{LEX}_n(\lambda) < \text{LEX}_q(\nu)$*), then* $c^{\lambda}_{\mu\nu} = 0$.

Before proving the proposition, we define the following operation which will allow us to compare partitions of p with partitions of n.

Definition 161. Given a partition $\mu = (\mu_1, \mu_2, \ldots)$ of p, the partition $\mu \uparrow^n_p$ is a partition of n given by $\mu \uparrow^n_p = (\mu_1 + n - p, \mu_2, \mu_3, \ldots)$. Thus μ and $\mu \uparrow^n_p$ agree on all parts except the first.

Lemma 162. *Let μ be a partition of p. The following inequality holds:* $\textsc{lex}_p(\mu) \leqslant \textsc{lex}_n(\mu \uparrow_p^n)$.

Proof. We proceed by induction on $\textsc{lex}_p(\mu)$. First, the lemma is obvious when $\textsc{lex}_p(\mu) = 1$ (when $\mu = (p)$).

Now we consider the case where $\textsc{lex}_p(\mu) > 1$. Let μ' be any partition of p with $\mu' \prec \mu$. First, we remark that $(\mu') \uparrow_p^n \prec_n \mu \uparrow_p^n$ since we are adding $n-p$ to both μ'_1 and μ_1.

By the inductive hypothesis, we have that

$$\begin{aligned}\textsc{lex}_p(\mu') &\leqslant \textsc{lex}_n\left((\mu') \uparrow_p^n\right) \\ &< \textsc{lex}_n\left(\mu \uparrow_p^n\right).\end{aligned}$$

Since we have shown that $\textsc{lex}_p(\mu') < \textsc{lex}_n(\mu \uparrow_p^n)$ for every μ' such that $\mu' \prec_p \mu$, we conclude that $\textsc{lex}_p(\mu) \leqslant \textsc{lex}_n(\mu \uparrow_p^n)$. □

Proof of Proposition 160. The assumption is that $\textsc{lex}_n(\lambda) < \textsc{lex}_p(\mu)$. By Lemma 162, it must also be the case that:

$$\textsc{lex}_n(\lambda) < \textsc{lex}_n(\mu \uparrow_p^n). \tag{C.3}$$

By definition of the lex ordering, Equation C.3 means that there exists some i for which $\left(\mu \uparrow_p^n\right)_i < \lambda_i$ *and* $\left(\mu \uparrow_p^n\right)_j = \lambda_j$ for all $j < i$.

We now argue that μ cannot possibly be a subpartition of λ, which will imply that $c_{\mu\nu}^{\lambda} = 0$ by the Littlewood-Richardson rule. We define the following partitions:

$$\lambda_{chop} = (\lambda_i, \lambda_{i+1}, \dots), \quad \mu_{chop} = (\mu_i, \mu_{i+1}, \dots).$$

Clearly, λ_{chop} is a partition of $a = n - \sum_{j=1}^{i-1} \lambda_j$ and μ_{chop} is a partition of $b = p - \sum_{j=1}^{i-1} \mu_j$. We will prove that μ_{chop} is not a subpartition of λ_{chop}, which will then imply that μ is not a subpartition of λ.

Now we know that $\left(\mu \uparrow_p^n\right)_1 \leqslant \lambda_1$ (if it were greater, then the assumption in Equation C.3 would be false). Thus, $\mu_1 + n - p \leqslant \lambda_1$. Or rearranging,

$$\mu_1 \leqslant \lambda_1 - n + p. \tag{C.4}$$

Thus, we have:

$$\begin{aligned} b = p - \sum_{j=1}^{i-1} \mu_j &= p - \mu_1 - \sum_{j=2}^{i-1} \mu_j, \\ &\geqslant p - (\lambda_1 - n + p) - \sum_{j=2}^{i-1} \lambda_j, \\ &\qquad \left(\text{by Equation C.4 and since } \mu_j = \lambda_j \right. \\ &\qquad\qquad \left. \text{for } j = 2, \dots, i-1 \text{ by our definition of } i\right) \\ &\geqslant n - \sum_{j=1}^{i-1} \lambda_j, \\ &\geqslant a. \end{aligned}$$

To summarize, we have shown that λ_{chop} and μ_{chop} are partitions of a and b, respectively, where $b \geqslant a$, but $\mu_i < \lambda_i$. Under these conditions, it is impossible for μ_{chop} to be a subpartition of λ_{chop}. We therefore must conclude that μ is not a subpartition of λ and hence, that $c^{\lambda}_{\mu\nu} = 0$. ■

C.4 PROOF THAT SPLIT RETURNS EXACT MARGINALS

The proof (of Theorem 72) that the splitting procedure gives exact marginals when the first order condition is satisfied comes from a simple modification to the existing proof of Proposition 67 which we now give below.

Consider the marginalization problem in which we compute the marginal distribution of $\sigma_p = [\sigma(1), \ldots, \sigma(p)]$ using:

$$f(\sigma_p) = \sum_{\sigma_q \in S_q} h([\sigma_p, \sigma_q]).$$

The Fourier transform of the marginal distribution at partition μ of p is given by:

$$\widehat{f}_\mu = \sum_{\sigma_p \in S_p} f(\sigma_p)\rho_\mu(\sigma_p), \tag{C.5}$$

$$\text{(Definition 24)} \tag{C.6}$$

$$= \sum_{\sigma_p \in S_p} \left(\sum_{\sigma_q \in S_q} h(\sigma_p, \sigma_q) \right) \rho_\mu(\sigma_p), \tag{C.7}$$

$$\text{(Expand } f(\sigma_p)\text{)} \tag{C.8}$$

$$= \sum_{\sigma \in S_p \times S_q} h(\sigma) \left(\rho_\mu \otimes \rho_{(q)}(\sigma) \right), \tag{C.9}$$

$$(\rho_{(q)} = 1 \text{ and rearrange}) \tag{C.10}$$

where $\rho_{(q)}$ is the trivial representation on S_q (recall that $\rho_{(q)}(\sigma_q) = 1$ for all $\sigma_q \in S_q$). We recognize the expression in line C.9 as being equal to one of the blocks from line 12.6 in the proof of Proposition 67 for appropriately chosen λ (i.e., is the Fourier coefficient matrix returned by the SPLIT algorithm), and we therefore conclude that the splitting operation will return the correct marginal distribution of h even if probabilistic independence fails to hold.

We make two additional remarks about the above argument:

- In line 12.2 of the proof of Proposition 67, we make the assumption that the distribution assigns zero probability mass to any permutation outside of $S_p \times S_q$.
- Note that independence assumption remains necessary for the join operation, and that while SPLIT(JOIN($\widehat{f}, \widehat{g}$)) should return $\widehat{f}$ and $\widehat{g}$ exactly, JOIN(SPLIT($\widehat{h}$)) will not exactly return $\widehat{h}$ unless h happens to already factor into independent f and g.

D

COMPUTING COUPLING MATRICES

In this appendix, we consider the general problem of finding an orthogonal operator which decomposes an arbitrary complex representation, $X(\sigma)$, of a finite group G [1]. This coupling matrix problem arises in our formulation of Bayesian conditioning (Chapter 8) as well as the Fourier domain probabilistic independence algorithms of Chapter 12. Unlike the Clebsch-Gordan series or the Littlewood-Richardson coefficients which are basis-independent numbers, intertwining operators must be recomputed if we change the underlying basis by which the irreducible representation matrices are constructed. However, for a fixed basis, we remind the reader that these intertwining operators need only be computed once and for all and can be stored in a table for future reference. Let X be any degree d group representation of G, and let Y be an equivalent direct sum of irreducibles, e.g.,

$$Y(\sigma) = \bigoplus_{\nu} \bigoplus_{\ell=1}^{z_\nu} \rho_\nu(\sigma), \tag{D.1}$$

where each irreducible ρ_ν has degree d_ν. We would like to compute an invertible (and orthogonal) operator C, such that $C \cdot X(\sigma) = Y(\sigma) \cdot C$, for all $\sigma \in G$. Throughout this section, we will assume that the multiplicities z_ν are known.

For concreteness, we focus on the problem of computing Clebsch-Gordan coefficients, where we would set $X = \rho_\lambda \otimes \rho_\mu$, and the multiplicities would be given by the Clebsch-Gordan series (Equation B.1). However, the same ideas hold for computing the coupling matrices correspond to the Littlewood-Richardson decomposition and for finding the matrix which relates marginal probabilities to irreducible coefficients (where we would set $X = \tau_\lambda$, and the multiplicities would be given by the Kostka numbers (Equation 5.10)).

D.1 AN ALGORITHM FOR COMPUTING COUPLING MATRICES

We will begin by describing an algorithm for computing a basis for the space of all possible intertwining operators which we denote by:

$$\mathrm{Int}_{[X;Y]} = \{C \in \mathbb{R}^{d \times d} \ : \ C \cdot X(\sigma) = Y(\sigma) \cdot C, \ \ \forall \sigma \in G\}.$$

We will then discuss some of the theoretical properties of $\mathrm{Int}_{[X;Y]}$ and show how to efficiently select an *orthogonal* element of $\mathrm{Int}_{[X;Y]}$.

1 Though the fundamental ideas in this section hold for a general finite group, we will continue to index irreducible by partitions and think of representations as being real-valued. To generalize the results, one can simply replace all transposes in this section by adjoints and think of η as indexing over the irreducibles of G rather than partitions.

Our approach is to naively[2] view the task of finding elements of $\mathrm{Int}_{[X,Y]}$ as a similarity matrix recovery problem, with the twist that the similarity matrix must be consistent over all group elements. To the best of our knowledge, the technique presented in this section is original. We first cast the problem of recovering a similarity matrix as a nullspace computation.

Proposition 163. *Let* A, B, C *be matrices and let* $K_{AB} = I \otimes A - B^T \otimes I$. *Then* $AC = CB$ *if and only if vec*$(C) \in$ *Nullspace*(K_{AB}).

Proof. A well known matrix identity [132] states that if A, B, C are matrices, then $\mathrm{vec}(ABC) = \left(C^T \otimes A\right) \mathrm{vec}(B)$. Applying the identity to $AC = CB$, we have:

$$\mathrm{vec}(ACI) = \mathrm{vec}(ICB),$$

and after some manipulation:

$$\left(I \otimes A - B^T \otimes I\right) \mathrm{vec}(C) = 0,$$

showing that $\mathrm{vec}(C) \in \mathrm{Nullspace}(K_{AB})$. □

For each $\sigma \in G$, the nullspace of the matrix $K(\sigma)$ constructed using the above proposition as:

$$K(\sigma) = I \otimes Y(\sigma) - X(\sigma) \otimes I, \tag{D.2}$$

where I is a $d \times d$ identity matrix, corresponds to the space of matrices C_σ such that

$$C_\sigma \cdot X(\sigma) = Y(\sigma) \cdot C, \qquad \text{for all } \sigma \in G.$$

To find the space of intertwining operators which are consistent across all group elements, we need to find the intersection:

$$\bigcap_{\sigma \in G} \mathrm{Nullspace}(K(\sigma)). \tag{D.3}$$

At first glance, it may seem that computing the intersection might require examining $n!$ nullspaces if $G = S_n$, but as luck would have it, most of the nullspaces in the intersection are extraneous. We use the following proposition which says that it suffices to find a similarity matrix consistent on any set of generators of the group.

Proposition 164. *Let* X *and* Y *be representations of finite group* G *and suppose that* G *is generated by the elements* $\sigma_1, \ldots, \sigma_m$. *If there exists an invertible linear operator* C *such that* $C \cdot X(\sigma_i) = Y(\sigma_i) \cdot C$ *for each* $i \in \{1, \ldots, m\}$, *then* X *and* Y *are equivalent as representations with* C *as the intertwining operator.*

2 In implementation, we use a more efficient algorithm for computing intertwining operators known as the *Eigenfunction Method* (EFM) [18]. Unfortunately, the EFM is too complicated for us to describe in this thesis. The method which we describe in this appendix is conceptually simpler than the EFM and generalizes easily to groups besides S_n.

Proof. We just need to show that C is a similarity transform for any other element of G as well. Let π be any element of G and suppose π can be written as the following product of generators: $\pi = \prod_{i=1}^{n} \sigma_i$. It follows that:

$$\begin{aligned} C^{-1} \cdot Y(\pi) \cdot C &= C^{-1} \cdot Y\left(\prod_i \sigma_i\right) \cdot C = C^{-1} \cdot \left(\prod_i Y(\sigma_i)\right) \cdot C \\ &= (C^{-1} \cdot Y(\sigma_1) \cdot C)(C^{-1} \cdot Y(\sigma_2) \cdot C) \cdots (C^{-1} \cdot Y(\sigma_m) \cdot C) \\ &= \prod_i \left(C^{-1} \cdot Y(\sigma_i) \cdot C\right) = \prod_i X(\sigma_i) = X\left(\prod_i \sigma_i\right) = X(\pi) \end{aligned}$$

Since this holds for every $\pi \in G$, we have shown C to be an intertwining operator between the representations X and Y. □

The good news is that despite having $n!$ elements, S_n can be generated by just two elements, namely, $(1,2)$ and $(1,2,\ldots,n)$ (see Chapter 2 for a proof), and so the problem reduces to solving for the intersection of two nullspaces, $(K(1,2) \cap K(1,2,\ldots,n))$, which can be done using standard numerical methods. Typically, the nullspace is multidimensional, showing that, for example, the Clebsch-Gordan coefficients for $\rho_\lambda \otimes \rho_\mu$ are not unique even up to scale.

Because $\text{Int}_{[X;Y]}$ contains singular operators (the zero matrix is a member of $\text{Int}_{[X;Y]}$, for example), not every element of $\text{Int}_{[X;Y]}$ is actually a legitimate intertwining operator as we require invertibility. In practice, however, since the singular elements correspond to a measure zero subset of $\text{Int}_{[X;Y]}$, one method for reliably selecting an operator from $\text{Int}_{[X;Y]}$ that "works" is to simply select a random element from the nullspace to be C. It may, however, be desirable to have an *orthogonal* matrix C which works as an intertwining operator. In the following, we discuss an object called the *Commutant Algebra* which will lead to several insights about the space $\text{Int}_{[X;Y]}$, and in particular, will lead to an algorithm for 'modifying' any invertible intertwining operator C to be an *orthogonal* matrix.

Definition 165. The *Commutant Algebra* of a representation Y is defined to be the space of operators which commute with Y:[3]

$$\text{Com}_Y = \{S \in \mathbb{R}^{d \times d} \ : \ S \cdot Y(\sigma) = Y(\sigma) \cdot S, \ \ \forall \sigma \in G\}.$$

The elements of the Commutant Algebra of Y can be shown to always take on a particular constrained form (shown using Schur's Lemma in [114]). In particular, every element of Com_Y takes the form

$$S = \bigoplus_\nu (M_{z_\nu} \otimes I_{d_\nu}), \tag{D.4}$$

where M_{z_ν} is some $z_\nu \times z_\nu$ matrix of coefficients and I_{d_ν} is the $d_\nu \times d_\nu$ identity (recall that the z_ν are the multiplicities from Equation D.1). Moreover, it can be shown that every matrix of this form must necessarily be an element of the Commutant Algebra.

The link between Com_Y and our problem is that the space of intertwining operators can be thought of as a 'translate' of the Commutant Algebra.

3 Notice that the definition of the Commutant Algebra does not involve the representation X.

Lemma 166. *There exists a vector space isomorphism between $Int_{[X;Y]}$ and Com_Y.*

Proof. Let R be any invertible element of $\mathrm{Int}_{[X;Y]}$ and define the linear map $f : \mathrm{Com}_Y \to \mathbb{R}^{d\times d}$ by: $f : S \mapsto (S \cdot R)$. We will show that the image of f is exactly the space of intertwining operators. Consider any element $\sigma \in G$:

$$\begin{aligned}(S \cdot R) \cdot X(\sigma) \cdot (S \cdot R)^{-1} &= S \cdot R \cdot X(\sigma) \cdot R^{-1} \cdot S^{-1},\\ &= S \cdot Y(\sigma) \cdot S^{-1} \quad (\text{since } R \in \mathrm{Int}_{[X;Y]}),\\ &= Y(\sigma) \quad (\text{since } S \in \mathrm{Com}_Y).\end{aligned}$$

We have shown that $S \cdot R \in \mathrm{Int}_{[X;Y]}$, and since f is linear and invertible, we have that $\mathrm{Int}_{[X;Y]}$ and Com_Y are isomorphic as vector spaces. □

Using the lemma, we can see that the dimension of $\mathrm{Int}_{[X;Y]}$ must be the same as the dimension of Com_Y, and therefore we have the following expression for the dimension of $\mathrm{Int}_{[X;Y]}$.

Proposition 167.

$$\dim Int_{[X;Y]} = \sum_{\nu} z_\nu^2.$$

Proof. To compute the dimension of $\mathrm{Int}_{[X;Y]}$, we need to compute the dimension of Com_Y, which can be accomplished simply by computing the number of free parameters in Equation D.4. Each matrix M_{z_ν} is free and yields z_ν^2 parameters, and summing across all irreducibles ν yields the desired dimension. □

To select an orthogonal intertwining operator, we will assume that we are given some invertible $R \in \mathrm{Int}_{[X;Y]}$ which is not necessarily orthogonal (such as a random element of the nullspace of K (Equation D.2)). To find an orthogonal element, we will 'modify' R to be an orthogonal matrix by applying an appropriate rotation, such that $R \cdot R^T = I$. We begin with a simple observation about $R \cdot R^T$.

Lemma 168. *If both X and Y are orthogonal representations and R is an invertible member of $Int_{[X;Y]}$, then the matrix $R \cdot R^T$ is an element of Com_Y.*

Proof. Consider a fixed $\sigma \in G$. Since $R \in \mathrm{Int}_{[X;Y]}$, we have that:

$$X(\sigma) = R^{-1} \cdot Y(\sigma) \cdot R.$$

It is also true that:

$$X(\sigma^{-1}) = R^{-1} \cdot Y(\sigma^{-1}) \cdot R. \tag{D.5}$$

Since $X(\sigma)$ and $Y(\sigma)$ are orthogonal matrices by assumption, Equation D.5 becomes:

$$X^T(\sigma) = R^{-1} \cdot Y^T(\sigma) \cdot R.$$

Algorithm D.1: Pseudocode for computing an orthogonal intertwining operators. The input is a degree d orthogonal matrix representation X evaluated at permutations $(1,2)$ and $(1,\ldots,n)$. The output of ComputeCG is a matrix C_ν with orthogonal rows such that $C_\nu^T \cdot \oplus^{z_\nu} \rho_\nu \cdot C_\nu = X$.

ComputeCG($X((1,2)), X((1,2,\ldots,n))$):

$K_1 \leftarrow I_{d\times d} \otimes (\oplus^{z_\nu} \rho_\nu(1,2)) - X(1,2) \otimes I_{d\times d}$;
$K_2 \leftarrow I_{d\times d} \otimes (\oplus^{z_\nu} \rho_\nu(1,\ldots,n)) X(1,\ldots,n) \otimes I_{d\times d}$;
$K \leftarrow [K_1; K_2]$; //*Stack* K_1 *and* K_2
$v \leftarrow \text{SparseNullspace}(K, z_\nu^2)$; //*Find the* d_ν^2*-dimensional nullspace*
$R \leftarrow \text{Reshape}(v; z_\nu d_\nu, d)$; //*Reshape* v *into a* $(z_\nu d_\nu) \times d$ *matrix*
$M \leftarrow \text{KroneckerFactors}(R \cdot R^T)$; //*Find* M *such that* $R \cdot R^T = M \otimes I_{d_\nu}$
$S_\nu \leftarrow \text{Eigenvectors}(M)$;
$C_\nu \leftarrow S_\nu^T \cdot R$;
NormalizeRows(C_ν);
return (C_ν);

Taking transposes,

$$X(\sigma) = R^T \cdot Y(\sigma) \cdot (R^{-1})^T.$$

We now multiply both sides on the left by R, and on the right by R^T,

$$\begin{aligned} R \cdot X(\sigma) \cdot R^T &= R \cdot R^T \cdot Y(\sigma) \cdot (R^{-1})^T \cdot R^T \\ &= R \cdot R^T \cdot Y(\sigma). \end{aligned}$$

Since $R \in \text{Int}_{[X;Y]}$,

$$Y(\sigma) \cdot R \cdot R^T = R \cdot R^T \cdot Y(\sigma),$$

which shows that $R \cdot R^T \in \text{Com}_Y$. □

We can now state and prove our orthogonalization procedure, which works by diagonalizing the matrix $R \cdot R^T$. Due to its highly constrained form, the procedure is quite efficient.

Theorem 169. *Let X be any orthogonal group representation of G and Y an equivalent orthogonal irreducible decomposition (As in Equation D.1). Then for any invertible element $R \in Int_{[X;Y]}$, there exists an (efficiently computable) orthogonal matrix T such that the matrix $T \cdot R$ is an element of $Int_{[X;Y]}$ and is* orthogonal.

Proof. Lemma 168 and Equation D.4 together imply that the matrix $R \cdot R^T$ can always be written in the form

$$R \cdot R^T = \oplus_\nu (M_{z_\nu} \otimes I_{d_\nu})$$

Since $R \cdot R^T$ is symmetric, each of the matrices M_{z_ν} is also symmetric and must therefore possess an orthogonal basis of eigenvectors. Define the matrix S_{z_ν} to be the matrix whose columns are the eigenvectors of M_{z_ν}.

The matrix $S = \oplus_\nu (S_{z_\nu} \otimes I_{d_\nu})$ has the following two properties:

1. $(S^T \cdot R)(S^T \cdot R)^T$ is a diagonal matrix:

   Each column of S is an eigenvector of $R \cdot R^T$ by standard properties of the direct sum and Kronecker product. Since each of the matrices, S_{z_ν}, is orthogonal, the matrix S is also orthogonal. We have:

   $$\begin{aligned}(S^T \cdot R)(S^T \cdot R)^T &= S^T \cdot R \cdot R^T \cdot S, \\ &= S^{-1} \cdot R \cdot R^T \cdot S, \\ &= D,\end{aligned}$$

   where D is a diagonal matrix of eigenvalues of $R \cdot R^T$.

2. $S^T \cdot R \in \mathrm{Int}_{[X;Y]}$:

   By Equation D.4, a matrix is an element of Com_Y if and only if it takes the form $\oplus_\nu(S_{z_\nu} \otimes I_{d_\nu})$. Since S can be written in the required form, so can S^T. We see that $S^T \in \mathrm{Com}_Y$, and by the proof of Lemma 166, we see that $S^T \cdot R \in \mathrm{Int}_{[X;Y]}$.

Finally, setting $T = D^{1/2} \cdot S^T$ makes the matrix $T \cdot R$ orthogonal (and does not change the fact that $T \cdot R \in \mathrm{Int}_{[X;Y]}$). □

We see that the complexity of computing T is dominated by the eigenspace decomposition of M_{z_ν}, which is $O\left(z_\nu^3\right)$. Pseudocode for computing orthogonal intertwining operators is given Algorithm D.1.

D.2 CONCLUSION

In this chapter we have introduced a simple method for computing coupling matrices between generic (finite) group representations and their irreducible decompositions. Surprisingly, there is little related work for this specific problem on the symmetric group. Most authors have focused on the basis independent problem of determining multiplicities within a decomposition (such as the Clebsch-Gordan series, Littlewood-Richardson coefficients and Kostka numbers) and have ignored the more numerical problem of determining specific intertwining matrices. There have been exceptions however, such as the work of Chen et al. (see [18] for example), who pioneered the use of the EFM (Eigenfunction method) for a number of numerical problems in group representation theory.

Our method has the advantage of being simple to understand and implement and requires little specific knowledge about the underlying group (as long as one can provide representation matrices evaluated at a finite set of group generators). For the symmetric group, we showed that only two representation matrices are needed as input and that our method applies to at least three problems: computing Clebsch-Gordan coupling matrices, Littlewood-Richardson coupling matrices, as well as matrices which convert between a matrix of marginals and its irreducible decomposition.

On the other hand, there is room for improvement. For example, our method currently does not exploit problem-specific domain knowledge that might otherwise lead to faster running times. For example, there are

many known symmetries [18] in the Clebsch-Gordan coupling matrices that we do not explicitly utilize to improve performance. Finding ways to leverage known sparsity and symmetry properties may lead to much faster results in practice. Finally, we note that a C++/Python implementation of the methods in this appendix is publicly available as part of our *PROPS Toolbox* [88].

E

STRUCTURE LEARNING APPENDIX

THIS section includes the proofs for Chapter 14.

E.1 SAMPLE COMPLEXITY PROOFS

Lemma 170 (adapted from [51]). *The entropy of a discrete random variable with arity R can be estimated to within accuracy Δ with probability $1-\beta$ using $O\left(\frac{R^2}{\Delta^2}\log^2\frac{R}{\Delta}\log\frac{R}{\beta}\right)$ i.i.d samples and the same time.*

Lemma 171. *The collection of mutual informations $I_{i;j,k}$ can be estimated to within accuracy Δ for all triplets (i,j,k) with probability at least $1-\gamma$ using $S(\Delta,\gamma) \equiv O\left(\frac{n^2}{\Delta^2}\log^2\frac{n}{\Delta}\log\frac{n^4}{\gamma}\right)$ i.i.d. samples and the same amount of time.*

Proof. Fix a $0 < \gamma \leqslant 1$ and Δ. For any fixed triplet (i,j,k), Hoffgen's result (Lemma 170) implies that $H(\sigma_i; \sigma_j < \sigma_k)$ can be estimated with accuracy Δ with probability at least $1-\gamma/n^3$ using $O\left(\frac{n^2}{\Delta^2}\log^2\frac{n}{\Delta}\log\frac{n^4}{\gamma}\right)$ i.i.d. samples since the variable $(\sigma_i, \sigma_j < \sigma_k)$ has arity $2n$ and setting $\beta \equiv \frac{\gamma}{n^3}$.

Estimating the mutual information for the same triplet therefore requires the same sample complexity by the expansion: $I_{i;j,k} = H(\sigma_i) + H(\sigma_j < \sigma_k) - H(\sigma_i; \sigma_j < \sigma_k)$. Now we use a simple union bound to bound the probability that the collection of mutual informations over all triplets is estimated to within Δ accuracy. Define $\Delta_{i,j,k} \equiv I_{i;j,k} - \hat{I}_{i;j,k}$.

$$P(|\Delta_{i,j,k}| < \Delta,\ \forall(i,j,k)) \geqslant 1 - \sum_{i,j,k} P(|\Delta_{i,j,k}| \geqslant \Delta) \geqslant 1 - n^3 \cdot \frac{\gamma}{n^3} \geqslant 1-\gamma.$$

□

Lemma 172. *Fix $k \leqslant n/2$. and let A be a k-subset of $\{1,\ldots,n\}$ with A riffle independent of its complement B. Let A' be a k-subset with $A' \neq A$ or B. If A and B are each ϵ-third order strongly connected, we have $\tilde{\mathcal{F}}(A') = \tilde{\mathcal{F}}(B') > \psi(n,k)\cdot\epsilon$, where $\psi(n,k) \equiv (n-k)(n-2k)$.*

Proof. Let us first establish some notation. Given a subset $X \subset \{1,\ldots,n\}$, define

$$\Omega_X^{int} \equiv \{(x;y,z) : x,y,z \in X\}.$$

Thus Ω_A^{int} and Ω_B^{int} are the sets of triplets whose indices are all internal to A or internal to B respectively. We define $\Omega_{A',B'}^{cross}$ to be the set of triplets which "cross" between the sets A and B:

$$\Omega_{A',B'}^{cross} \equiv \{(x;y,z) : x \in A, y,z \in B, \text{ or } x \in B, y,z \in A\}.$$

The goal of this proof is to use the strong connectivity assumptions to lower bound $\tilde{\mathcal{F}}(A')$. In particular, due to strong connectivity, each triplet inside $\Omega^{cross}_{A',B'}$ that also lies in either Ω^{int}_A or Ω^{int}_B must contribute at least ϵ to the objective function $\tilde{\mathcal{F}}(A')$. It therefore suffices to lower bound the number of triplets which cross between A' and B', but are internal to either A or B (i.e., $|\Omega^{cross}_{A',B'} \cap (\Omega^{int}_A \cup \Omega^{int}_B)|$). Define $\ell \equiv |A \cap A'|$ and note that $0 \leqslant \ell < k$. It is straightforward to check that: $|A \cap B'| = k - \ell$, $|B \cap A'| = k - \ell$, and $|B \cap B'| = (n-k) - (k-\ell) = n + \ell - 2k$.

$$\begin{aligned}
|\Omega^{cross}_{A',B'} \cap (\Omega^{int}_A \cup \Omega^{int}_B)| &= |\Omega^{cross}_{A',B'} \cap \Omega^{int}_A| + |\Omega^{cross}_{A',B'} \cap \Omega^{int}_B|, \\
&\geqslant \ell(k-\ell)^2 + \ell^2(k-\ell) + (k-\ell)(n+\ell-2k)^2 + (n+\ell-2k)(k-\ell)^2, \\
&\geqslant (k-\ell)\left((n-k)(n-2k) + \ell n\right), \\
&\geqslant k\left((n-k)(n-2k) + kn\right).
\end{aligned}$$

We do want the bound above to depend on ℓ. Intuitively, for a fixed k and n, the above expression is minimized when either $\ell = 0$ or $k-1$ (a more formal argument is shown below in the proof of Lemma 173). Plugging $\ell = 0$ and $k-1$ and bounding from below yields:

$$\begin{aligned}
|\Omega^{cross}_{A',B'} \cap (\Omega^{int}_A \cup \Omega^{int}_B)| &\geqslant \min\left(k(n-k)(n-2k), (n-k)(n-2k) + n(k-1)]\right), \\
&\geqslant (n-k)(n-2k).
\end{aligned}$$

Finally due to strong connectivity, we know that for each triplet in $\Omega^{int}_A \cup \Omega^{int}_B$, we have $I_{x;y,z} > \epsilon$, thus each edge in $\Omega^{cross}_{A',B'} \cap (\Omega^{int}_A \cup \Omega^{int}_B)$ contributes at least ϵ to $\tilde{\mathcal{F}}(A')$, establishing the desired result. □

Lemma 173. *Under the same assumptions as Lemma 172,* $p(n,k,\ell) = (k-\ell)\left((n-k)(n-2k) + \ell n\right)$ *is minimized at either* $\ell = 0$ *or* $k-1$.

Proof. Let $\alpha = (n-k)(n-2k)$. We know that $\alpha \geqslant 0$ since $k \leqslant n/2$ by assumption (and equals zero only when $k = n/2$). We want to find the $\ell \in \{0, \ldots, k-1\}$ which minimizes the concave quadratic function $p(\ell) = (k-\ell)(\alpha + \ell n)$, the roots of which are $\ell = k$ and $\ell = -\alpha/n$ (note that $-\alpha/n \leqslant 0$. The minimizer is thus the element of $\{0, \ldots, k-1\}$ which is closest to either of the roots. □

Theorem 174. *Let* A *be a* k*-subset of* $\{1, \ldots, n\}$ *with* A *riffle independent of its complement* B. *If* A *and* B *are each* ϵ*-third order strongly connected, then given* $S(\Delta, \epsilon) \equiv O\left(\frac{n^4}{\epsilon^2} \log^2 \frac{n^2}{\epsilon} \log \frac{n^4}{\gamma}\right)$ *i.i.d. samples, the minimum of* $\hat{\mathcal{F}}$ *(evaluated over all* k*-subsets of* $\{1, \ldots, n\}$*) is achieved at exactly the subsets* A *and* B *with probability at least* $1 - \gamma$.

Proof. Let A' be a k-subset with $A' \neq A$ or B. Our goal is to show that $\hat{\mathcal{F}}(A') > \hat{\mathcal{F}}(A)$.

Denote the error between estimated mutual information and true mutual information by $\Delta_{i;j,k} \equiv \hat{I}_{i;j,k} - I_{i;j,k}$. We have:

$$\begin{aligned}
\hat{\mathcal{F}}(A') - \hat{\mathcal{F}}(A) &= \left(\sum_{(i,j,k) \in \Omega^{cross}_{A',B'}} \hat{I}_{i;j,k} \right) - \left(\sum_{(i,j,k) \in \Omega^{cross}_{A,B}} \hat{I}_{i;j,k} \right) \\
&= \tilde{\mathcal{F}}(A') - \tilde{\mathcal{F}}(A) + \sum_{(i,j,k) \in \Omega^{cross}_{A',B'}} \Delta_{i;j,k} - \sum_{(i,j,k) \in \Omega^{cross}_{A,B}} \Delta_{i;j,k} \\
&\geqslant \psi(n,k) \cdot \epsilon + \sum_{(i,j,k) \in \Omega^{cross}_{A',B'}} \Delta_{i;j,k} - \sum_{(i,j,k) \in \Omega^{cross}_{A,B}} \Delta_{i;j,k} \\
&\quad \text{(by Lemma 172 and } \tilde{\mathcal{F}}(A) = 0\text{)}
\end{aligned}$$

Now assume that all of the estimation errors Δ are uniformly bounded as:

$$|\Delta_{i;j,k}| \leqslant \frac{\epsilon}{4} \left(\frac{\psi(n,k)}{n^2 k - k^2 n} \right). \tag{E.1}$$

And note that $|\Omega^{cross}_{A',B'}| = |\Omega^{cross}_{A,B}| = k^2(n-k) + k(n-k)^2 = n^2 k - k^2 n$. We have:

$$\begin{aligned}
\sum_{(i,j,k) \in \Omega^{cross}_{A',B'}} |\Delta_{i;j,k}| - \sum_{(i,j,k) \in \Omega^{cross}_{A,B}} |\Delta_{i;j,k}| &\leqslant 2 \cdot (n^2 k - k^2 n) \cdot \frac{\epsilon}{4} \left(\frac{\psi(n,k)}{n^2 k - k^2 n} \right) \\
&\leqslant \frac{\epsilon \psi(n,k)}{2} \\
&\leqslant \epsilon \cdot \psi(n,k)
\end{aligned}$$

Combining this bound on the estimation errors with the bound on $\hat{\mathcal{F}}(A') - \hat{\mathcal{F}}(A)$ yields:

$$\begin{aligned}
\hat{\mathcal{F}}(A') - \hat{\mathcal{F}}(A) &\geqslant \epsilon \psi(n,k) - \left(\sum_{(i,j,k) \in \Omega^{cross}_{A',B'}} |\Delta_{i;j,k}| - \sum_{(i,j,k) \in \Omega^{cross}_{A,B}} |\Delta_{i;j,k}| \right) \\
&\geqslant \frac{\epsilon \psi(n,k)}{2} \\
&> 0,
\end{aligned}$$

which is almost what we want to show. How many samples do we require to achieve the bound assumed in Equation E.1 with high probability? Observe that the bound simplifies as,

$$\frac{\epsilon}{4} \left(\frac{\psi(n,k)}{n^2 k - k^2 n} \right) = \frac{\epsilon}{4} \left(\frac{(n-k)(n-2k)}{nk(n-k)} \right) = \frac{\epsilon}{4} \left(\frac{n-2k}{nk} \right),$$

which behaves like $O(\epsilon)$ when k is $O(1)$, but like $O\left(\frac{\epsilon}{n}\right)$ when k is $O(n)$. Applying the sample complexity result of Lemma 171 with $\Delta = O(\epsilon/n)$, we see that given $O\left(\frac{n^4}{\epsilon^2} \log^2 \frac{n^2}{\epsilon} \log \frac{n^4}{\gamma} \right)$ i.i.d. samples, the bound in Equation E.1 holds with probability $1 - \gamma$, concluding the proof. □

E.2 LOG-LIKELIHOOD INTERPRETATIONS

If we examine the KL divergence objective introduced in Section 14.2, it is a standard fact that minimizing Equation 14.2 is equivalent to find the structure which maximizes the log-likelihood of the training data.

$$\begin{aligned}
\mathcal{F}[A,B] &= D_{KL}(\hat{h}(\sigma) \,||\, m(\tau_{A,B}(\sigma))f(\phi_A(\sigma))g(\phi_B(\sigma))), \\
&= \sum_{\sigma \in S_n} \hat{h}(\sigma) \log \left(\frac{\hat{h}(\sigma)}{m(\tau_{A,B}(\sigma))f(\phi_A(\sigma))g(\phi_B(\sigma))} \right), \\
&= \text{const.} - \sum_{\sigma \in S_n} \hat{h}(\sigma) \log \left(m(\tau_{A,B}(\sigma))f(\phi_A(\sigma))g(\phi_B(\sigma)) \right), \\
&= \text{const.} - \log \left(\prod_{i=1}^{m} m(\tau_{A,B}(\sigma^{(i)}))f(\phi_A(\sigma^{(i)}))g(\phi_B(\sigma^{(i)})) \right).
\end{aligned}$$

In the above (with some abuse of notation), m, f and g are estimated using counts from the training data. The equivalence is significant because it justifies structure learning for data which is not necessarily generated from a distribution which factors into riffle independent components. Using a similar manipulation, we can rewrite our objective function (Equation 14.3) as:

$$\begin{aligned}
\mathcal{F}[A,B] &= I(\sigma(A)\,;\,\phi_B(\sigma)) + I(\sigma(B)\,;\,\phi_A(\sigma)), \\
&= \text{const.} - \log \left(\prod_{i=1}^{m} \psi_A(\phi_A(\sigma^{(i)}), \tau_{A,B}(\sigma^{(i)}))g(\phi_B(\sigma^{(i)})) \right) \\
&\quad - \log \left(\prod_{i=1}^{m} f(\phi_A(\sigma^{(i)}))\psi_B(\phi_B(\sigma^{(i)}), \tau_{A,B}(\sigma^{(i)})) \right),
\end{aligned}$$

which we can see to be a "composite" of two likelihood functions. We are evaluating our data log-likelihood first under a model in which the absolute ranks of items in A are independent of relative ranks of items in B, and secondly under a model in which the absolute ranks of items in B are independent of relative ranks of items in A. Here again, ψ_A and ψ_B are estimated using counts of the training data. We see that if these distributions, ψ_A, ψ_B factor along their inputs, then optimizing the objective function is equivalent to optimizing the likelihood under the riffle independent model. Thus, if the data is already riffle independent (or nearly riffle independent), then the structure learning objective can indeed to be interpreted as maximizing the log-likelihood of the data, but otherwise there does not seem to be a clear equivalence between the two objective functions.

E.3 WHY TESTING FOR INDEPENDENCE OF RELATIVE RANKS IS INSUFFICIENT

Why can we not just check to see that the relative ranks of A are independent of the relative ranks of B? Another natural objective function for detecting riffle independent subsets is:

$$\mathcal{F}[A,B] \equiv I(\phi_A(\sigma)\,;\,\phi_B(\sigma)). \tag{E.2}$$

Equation E.2 is certainly a necessary condition for subsets A and B to be riffle independent but why would it not be sufficient? It is easy to construct a counterexample — simply find a distribution in which the interleaving depends on either of the relative rankings.

Example 175. *In this example, we will consider a distribution on* S_4. *Let* $A = \{1,2\}$ *and* $B = \{3,4\}$. *To generate rankings* $\sigma \in S_4$, *we will draw independent relative rankings,* σ_A *and* σ_B, *with uniform probability for each of* A *and* B. *Then set the interleaving as follows:*

$$\tau = \begin{cases} [\![AABB]\!] & \text{if } \sigma_A = (1,2) \\ [\![BBAA]\!] & \text{otherwise} \end{cases}.$$

Finally set $\sigma = \tau \cdot [\sigma_A, \sigma_B]$.

Since the relative rankings are independent, $\mathcal{F}[A,B] = 0$. *But since the interleaving depends on the relative ranking of items in* A, *we see that* A *and* B *are not riffle independent in this example.*

F

PARTIAL RANKING PROOFS

In this appendix, we provide supplementary proofs of some of the theoretical results from Chapter 15.

F.1 THE PSPAN OF A SET IS ALWAYS A PARTIAL RANKING

To reason about the PSPAN of a set of rankings, we first introduce some basic concepts regarding the combinatorics of partial rankings. The collection of partial rankings over Ω forms a *partially ordered set (poset)* where $S_{\gamma'}\pi' \prec S_{\gamma}\pi$ if $S_{\gamma}\pi$ can be obtained from $S_{\gamma'}\pi'$ by dropping vertical lines. For example, on S_3, we have that $1|2|3 \prec 12|3$. The *Hasse diagram* is the graph in which each node corresponds to a partial ranking and a node x is connected to node y via an edge if $x \prec y$ and there exists no partial ranking z such that $x \prec z \prec y$ (see [86]). At the top of the Hasse diagram is the partial ranking $1, 2, \ldots, n$ (i.e., all of S_Ω) and at the bottom of the Hasse diagram lie the full rankings. See Figure 62 for an example of the partial ranking lattice on S_3.

Lemma 176. *[Lebanon and Mao (2008) [86]] Given any two partial rankings $S_{\gamma}\pi$, $S_{\gamma'}\pi'$, there exists a unique supremum of $S_{\gamma}\pi$ and $S_{\gamma'}\pi'$ (a node $S_{\gamma_{sup}}\pi_{sup}$ such that $S_{\gamma}\pi \prec S_{\gamma_{sup}}\pi_{sup}$ and $S_{\gamma'}\pi' \prec S_{\gamma_{sup}}\pi_{sup}$, and any other such node is greater than $S_{\gamma_{sup}}\pi_{sup}$). Similarly, there exists a unique infimum of $S_{\gamma}\pi$ and $S_{\gamma'}\pi'$.*

Lemma 177. *Given two partial rankings $S_{\gamma}\pi$, $S_{\gamma'}\pi'$, the relation $S_{\gamma'}\pi' \subset S_{\gamma}\pi$ holds if and only $S_{\gamma}\pi$ lies above $S_{\gamma'}\pi'$ in the Hasse diagram.*

Proof. If $S_{\gamma}\pi$ lies above $S_{\gamma'}\pi'$ in the Hasse diagram, then $S_{\gamma'}\pi' \subset S_{\gamma}\pi$ is trivial since $S_{\gamma}\pi$ can be obtained by dropping vertical bars of $S_{\gamma'}\pi'$. Now given that $S_{\gamma}\pi$ *does not* lie above $S_{\gamma'}\pi'$, we would like to show that $S_{\gamma'}\pi' \not\subset S_{\gamma}\pi$. Let $S_{\gamma_{inf}}\pi_{inf}$ be the unique infimum of $S_{\gamma}\pi$ and $S_{\gamma'}\pi'$ as guaranteed by Lemma 176. By the definition of the Hasse diagram, both $S_{\gamma}\pi$ and $S_{\gamma}\pi$ can be obtained by 'dropping' verticals from the vertical bar representation of $S_{\gamma_{inf}}\pi_{inf}$. Since $S_{\gamma}\pi$ does not lie above $S_{\gamma'}\pi'$, there must be a vertical bar that was dropped by $S_{\gamma'}\pi'$ which was not dropped by $S_{\gamma}\pi$ (if there does not exist such a bar, then $S_{\gamma'}\pi' \subset S_{\gamma}\pi$), and hence there must exist a pair of items i, j separated by a single vertical bar in $S_{\gamma}\pi$ but unseparated in $S_{\gamma'}\pi'$. Therefore there exists $\sigma \in S_{\gamma'}\pi'$ such that $\sigma(j) < \sigma(i)$ even though there exists no such $\sigma \in S_{\gamma}\pi$. We conclude that $S_{\gamma'}\pi' \not\subset S_{\gamma}\pi$. □

Lemma 178. *For any $X \subset S_n$, PSPAN(X) is a partial ranking.*

Proof. Consider any subset $X \subset S_n$. A partial ranking containing every element in X must be an upper bound of every element of X in the Hasse

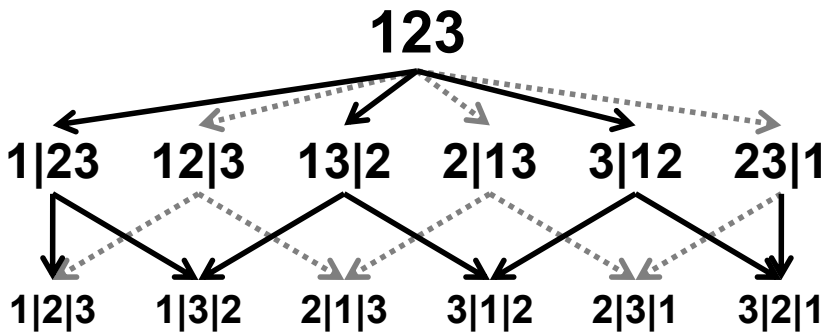


Figure 62: The Hasse diagram for the lattice of partial rankings on S_3.

diagram by Lemma 177. By Lemma 176, there must exist a unique least upper bound (supremum) of X, $S_{\gamma_{sup}}\pi_{sup}$, such that for any common upper bound $S_\gamma\pi$ of X, $S_\gamma\pi$ must also be an ancestor of $S_{\gamma_{sup}}\pi_{sup}$ and hence $S_{\gamma_{sup}}\pi_{sup} \subset S_\gamma\pi$. We therefore see that any partial ranking containing X must be a superset of $S_{\gamma_{sup}}\pi_{sup}$. On the other hand, $S_{\gamma_{sup}}\pi_{sup}$ is itself a partial ranking containing X. Since $\textsc{pspan}(X)$ is the intersection of partial rankings containing X, we have $\textsc{pspan}(X) = S_{\gamma_{sup}}\pi_{sup}$ and therefore that $\textsc{pspan}(X)$ must be a partial ranking. □

F.2 SUPPLEMENTARY PROOFS FOR THE CLAIM THAT $\textsc{rspan}(X) = \textsc{pspan}(X)$

To simplify the notation in some of the remaining proofs, we introduce the following definition.

Definition 179 (Ties)**.** Given a partial ranking $S_\gamma\pi = \Omega_1|\ldots|\Omega_k$, we say that items a_1 and a_2 are *tied* (written $a_1 \sim a_2$) with respect to $S_\gamma\sigma$ if $a_1, a_2 \in \Omega_i$ for some i.

The following basic properties of the tie relation are straightforward.

Proposition 180.

I. With respect to a fixed partial ranking $S_\gamma\pi$, the tie relation, $\sim$, is an equivalence relation on the item set (i.e., is reflexive, symmetric and transitive).

II. If there exist $\sigma, \sigma' \in S_\gamma\pi$ which disagree on the relative ranking of items a_1 and a_2, then $a_1 \sim a_2$ with respect to $S_\gamma\pi$.

III. If $S_\gamma\pi \prec S_{\gamma'}\pi'$, and $a_1 \sim a_2$ with respect to $S_\gamma\pi$, then $a_1 \sim a_2$ with respect to $S_{\gamma'}\pi'$.

IV. If $a_1 \sim a_2$ with respect to $S_\gamma\pi$, and $\sigma(a_1) < \sigma(a_3) < \sigma(a_2)$ for some item $a_3 \in \Omega$ and some $\sigma \in S_\gamma\pi$, then $a_1 \sim a_2 \sim a_3$.

Proposition 181. *Given a set of rankings X as input, Algorithm 15.2 outputs* $\textsc{pspan}(X)$.

Proof. We prove three things, which together prove the proposition: (1) that the algorithm terminates, (2) that at each stage the elements of X are contained in $\textsc{pspan}(X)$, and (3) that upon termination, $\textsc{pspan}(X)$ is contained in each element of X.

1. First we note that the algorithm must terminate in finitely many iterations of the while loop since at each stage at least one vertical bar is removed from a partial ranking, and when all of the vertical bars have been removed from the elements of X, there are no disagreements on relative ordering.
2. We now show that at any stage in the algorithm, every element of X_t is a subset of the PSPAN(X). Consider $S_\gamma\pi \in X_t$ such that $S_\gamma\pi \subset$ PSPAN(X). If $S_\gamma\pi$ is replaced by $S_{\gamma'}\pi'$ in X_{t+1}, then we want to show that $S_{\gamma'}\pi' \subset$ PSPAN(X) as well. From Algorithm 15.2, for some i, if $S_\gamma\pi = \Omega_1|\ldots|\Omega_j|\Omega_{j+1}|\ldots|\Omega_k$, $S_{\gamma'}\pi'$ can be written as $\Omega_1|\ldots|\Omega_j \cup \Omega_{j+1}|\ldots|\Omega_k$, where the vertical bar between Ω_j and Ω_{j+1} are deleted due to the existence of partial rankings in X_t which disagree on the relative ordering of items a_1, a_2 on opposite sides of the bar, then by Proposition 126 (II), we know that $a_1 \sim a_2$ (with respect to $S_\gamma\pi$). By transitivity (I) and (II), if $a_1 \in \Omega_i$ and $a_2 \in \Omega_{i'}$, then any two elements of Ω_i and $\Omega_{i'}$ are also tied. By (IV), all the items lying in $\Omega_i, \Omega_{i+1}, \ldots, \Omega_{i'}$ are thus tied with respect to PSPAN(X) and therefore removing any bar between items a_1 and a_2 (producing, for example, $S_{\gamma'}\pi'$) results in a partial ranking which is a subset of PSPAN(X).
3. Finally, upon termination, if some ranking $\sigma \in X$ is not contained in some element $S_\gamma\pi \in X_t$, then there would exist two items a_1, a_2 whose relative ranking σ and $S_\gamma\pi$ disagree upon, which is a contradiction. Therefore, every element $S_\gamma\pi \in X_t$ contains every element of X and thus PSPAN$(X) \subset S_\gamma\pi$ for every $S_\gamma\pi \in X_t$.

□

Lemma 182. *Let $S_\gamma\pi = \Omega_1|\ldots|\Omega_i|\Omega_{i+1}|\ldots|\Omega_k$ be a partial ranking on item set Ω, and $S_{\gamma'}\pi' = \Omega_1|\ldots|\Omega_i \cup \Omega_{i+1}|\ldots|\Omega_k$, the partial ranking in which the sets Ω_i and Ω_{i+1} are merged. Let $a_1 \in \cup_{j=1}^{i}\Omega_j$ and $a_2 \in \cup_{j=i+1}^{k}\Omega_j$. If $\mathcal{O}$ is any element of $\mathcal{CRI}$ such that $S_\gamma\pi \subset \mathcal{O}$ and there additionally exists a ranking $\tilde{\pi} \in \mathcal{O}$ which disagrees with $S_\gamma\pi$ on the relative ordering of a_1, a_2, then $S_{\gamma'}\pi' \subset \mathcal{O}$.*

Proof. We will fix a completely decomposable $\mathcal{O}$ and again work with h, the indicator distribution corresponding to $\mathcal{O}$. Let $\sigma \in S_{\gamma'}\pi'$. To prove the lemma, we need to establish that $h(\sigma) > 0$. Let σ^0 be any element of $S_\gamma\pi$ such that $\sigma^0(k) = \sigma(k)$ for all $k \in \Omega\backslash(\Omega_i \cup \Omega_{i+1})$. Since $S_\gamma\pi \subset \text{supp}(h)$ by assumption, we have that $h(\sigma^0) > 0$.

Since σ^0 and σ match on all items except for those in $\Omega_i \cup \Omega_{i+1}$, there exists a sequence of rankings $\sigma^0, \sigma^1, \sigma^2, \ldots, \sigma^m = \sigma$ such that adjacent rankings in this sequence differ only by a pairwise exchange of items$b_1, b_2 \in \Omega_i \cup \Omega_{i+1}$. We will now show that at each step along this sequence, $h(\sigma^t) > 0$ implies that $h(\sigma^{t+1}) > 0$, which will prove that $h(\sigma) > 0$. Suppose now that $h(\sigma^t) > 0$ and that σ^t and σ^{t+1} differ only by the relative ranking of items $b_1, b_2 \in \Omega_i \cup \Omega_{i+1}$ (without loss of generality, we will assume that $\sigma^t(b_2) < \sigma^t(b_1)$ and $\sigma^{t+1}(b_1) < \sigma^{t+1}(b_2)$).

The idea of the following paragraph is to use the previous lemma (Lemma 127) to prove that σ^{t+1} has positive probability and to do so,

it will be necessary to argue that there exists some ranking σ' such that $h(\sigma') > 0$ and $\sigma'(b_1) < \sigma'(b_2)$ (i.e., σ' disagrees with σ^t on the relative ranking of b_1, b_2). Let ω be any element of $S_\gamma\pi$. If $a_1 \in \Omega_i$, rearrange ω such that a_1 is ranked first among elements of Ω_i. If $a_2 \in \Omega_{i+1}$, further rearrange ω such that a_2 is ranked last among elements of Ω_{i+1}. Note that ω is still an element of $S_\gamma\pi$ after the possible rearrangements and therefore $h(\omega) > 0$. We can assume that $\omega(b_2) < \omega(b_1)$ since otherwise we will have shown what we wanted to show. Thus the relative ordering of a_1, a_2, b_1, b_2 within ω is $a_1|b_2|b_1|a_2$. Note that we treat the case where the items a_1, a_2, b_1, b_2 are distinct, but the same argument follows in the cases when $a_1 = b_2$ or $a_2 = b_1$.

Now since $\tilde{\pi}$ disagrees with $S_\gamma\pi$ on the relative ordering of a_1, a_2 by assumption (and hence disagrees with ω), we apply Lemma 127 to conclude that swapping the relative ordering of a_1, a_2 within ω (obtaining $a_2|b_2|b_1|a_1$) results in a ranking, ω', such that $h(\omega') > 0$. Finally, observe that ω and ω' must now disagree on the relative ranking of a_2, b_2, and invoking Lemma 127 again shows that we can swap the relative ordering of a_2, b_2 within ω (obtaining $a_1|a_2|b_1|b_2$) to result in a ranking σ' such that $h(\sigma') > 0$. This element σ' ranks b_1 before b_2, which is what we wanted to show.

We have shown that there exist rankings which disagree on the relative ordering of b_1 and b_2 with positive probability under h. Again applying Lemma 127 shows that we can swap the relative ordering of items b_1, b_2 within σ^t to obtain σ^{t+1} such that $h(\sigma^{t+1}) > 0$, which concludes the proof. □

[1] Nir Ailon. Aggregation of partial rankings, p-ratings and top-m lists. In *Proceedings of the eighteenth annual ACM-SIAM symposium on Discrete algorithms*, SODA '07, New Orleans, Louisiana, 2007.

[2] Adi Akavia. Finding significant fourier transform coefficients deterministically and locally. Technical Report TR-08-102, Electronic Colloquium on Computational Complexity, November 2008.

[3] Gorjan Alagic and Alexander Russell. Uncertainty principles for compact groups. *Illinois Journal of Mathematics*, 52(2):1315–1324, 2008.

[4] David Aldous and Persi Diaconis. Shuffling cards and stopping times. *American Mathematical Monthly*, 93(5):333–348, 1986.

[5] A. Arnaud Doucet, Nando de Freitas, and Neil J. Gordon, editors. *Sequential Monte Carlo Methods in Practice*. Springer Verlag, New York, 2001.

[6] Kenneth Arrow. *Social Choice and Individual Values*. Yale University Press, 1963.

[7] Francis R. Bach and Michael I. Jordan. Thin junction trees. In Thomas G. Dietterich, Suzanna Becker, and Zoubin Ghahramani, editors, *Advances in Neural Information Processing Systems 14*, NIPS '01, pages 569–576, Cambridge, Massachusetts, 2001. MIT Press.

[8] Hamsa Balakrishnan, Inseok Hwang, and Claire J. Tomlin. Polynomial approximation algorithms for belief matrix maintenance in identity management. In *Proceedings of the 43rd IEEE Conference on Decision and Control, Bahamas*, pages 4874–4879, Atlantis, Paradise Island, Bahamas, 2004.

[9] Yaakov Bar-Shalom and Thomas E. Fortmann. *Tracking and Data Association*. Academic Press, 1988.

[10] John J. Bartholdi, Craig A. Tovey, and Michael Trick. Voting schemes for which it can be difficult to tell who won. *Social Choice and Welfare*, 6(2), 1989.

[11] Dave Bayer and Persi Diaconis. Trailing the dovetail shuffle to its lair. *The Annals of Probability*, 1992.

[12] Vincent Berry and Olivier Gascuel. On the interpretation of bootstrap trees: Appropriate threshold of clade selection and induced gain. *Molecular Biology and Evolution*, 13:999–1011, 1996.

[13] Xavier Boyen and Daphne Koller. Tractable inference for complex stochastic processes. In *The 14th Conference on Uncertainty in Artificial Intelligence*, UAI '98, Madison, Wisconsin, 1998.

[14] Frank J. Budden. *The Fascination of Groups*. Cambridge University Press, Cambridge, England, 1972. (Chapter 24: *Ringing the changes: groups and campanology*).

[15] Ludwig M. Busse, Peter Orbanz, and Joachim Buhmann. Cluster analysis of heterogeneous rank data. In *The 24th Annual International Conference on Machine Learning*, ICML '07, Corvallis, Oregon, June 2007.

[16] Anton Chechetka and Carlos Guestrin. Efficient principled learning of thin junction trees. In J.C. Platt, D. Koller, Y. Singer, and S. Roweis, editors, *Advances in Neural Information Processing Systems 20*, NIPS '07, pages 273–280, Cambridge, Massachusetts, 2008. MIT Press.

[17] Harr Chen, S. R. K. Branavan, Regina Barzilay, and David R. Karger. Global models of document structure using latent permutations. In *Proceedings of Human Language Technologies: The 2009 Annual Conference of the North American Chapter of the Association for Computational Linguistics*, NAACL '09, pages 371–379, Stroudsburg, PA, USA, 2009. Association for Computational Linguistics.

[18] Jin-Quan Chen. *Group Representation Theory for Physicists*. World Scientific, 1989.

[19] Michael Clausen. Fast generalized fourier transforms. *Theoretical Computer Science*, 67(1):55–63, 1989.

[20] Michael Clausen and Ulrich Baum. Fast Fourier transforms for symmetric groups: Theory and implementation. *Mathematics of Computations*, 61(204):833–847, 1993.

[21] Joseph Collins and Jeffrey Uhlmann. Efficient gating in data association with multivariate distributed states. *IEEE Transactions on Aerospace and Electronic Systems*, 28, 1992.

[22] The Sloan Consortium. Making the grade: Online education in the United States, 2006.

[23] James Cooley and John Tukey. An algorithm for the machine calculation of complex fourier series. *Mathematical Computation*, 19:297–301, 1965.

[24] Thomas M. Cover and Joy A. Thomas. *Elements of Information Theory*. Wiley, 1991.

[25] Ingemar J. Cox and Sunita L. Hingorani. An efficient implementation of Reid's multiple hypothesis tracking algorithm and its evaluation for the purpose of visual tracking. In *International Conference on Pattern Recognition*, pages 437–443, 1994.

[26] Douglas E. Critchlow. *Metric Methods for Analyzing Partially Ranked Data*. Springer-Verlag, 1985.

[27] Ingrid Daubechies. *Ten lectures on wavelets*. Society for Industrial and Applied Mathematics, Philadelphia, PA, USA, 1992.

[28] Zajj Daugherty, Alexander K. Eustis, Gregory Minton, and Michael E. Orrison. Voting, the symmetric group, and representation theory, 2007.

[29] Persi Diaconis. *Group Representations in Probability and Statistics*. Institute of Mathematical Statistics, 1988.

[30] Persi Diaconis. A generalization of spectral analysis with application to ranked data. *The Annals of Statistics*, 17(3):949–979, 1989.

[31] Arnaud Doucet, Nando de Freitas, Kevin Murphy, and Stuart Russell. Rao-Blackwellised particle filtering for dynamic Bayesian networks. In *Proceedings of the Sixteenth Conference on Uncertainty in Artificial Intelligence*, UAI '00, pages 176–183, Stanford, 2000.

[32] David S. Dummit and Richard M. Foote. *Abstract Algebra*. Wiley, 3 edition, 2003.

[33] Bradley Efron and Robert Tibshirani. *An introduction to the bootstrap*. Chapman and Hall, London, 1993.

[34] Alexei A. Efros, Alexander C. Berg, Greg Mori, and Jitendra Malik. Recognizing action at a distance. In *IEEE International Conference on Computer Vision*, pages 726–733, Nice, France, 2003.

[35] Vivek Farias, Srikanth Jagabathula, and Devavrat Shah. A data-driven approach to modeling choice. In Y. Bengio, D. Schuurmans, J. Lafferty, C. K. I. Williams, and A. Culotta, editors, *Advances in Neural Information Processing Systems 22*, NIPS '08, pages 504–512. 2009.

[36] Michael A. Fligner and Joseph S. Verducci. Distance based ranking models. *Journal of the Royal Statistical Society*, 48, 1986.

[37] Richard Foote, Gagan Mirchandani, and Daniel Rockmore. Two-dimensional wreath product transforms. *Journal of Symbolic Computation*, 37(2):187–207, 2004.

[38] Yoav Freund, Raj Iyer, Robert E. Schapire, and Yoram Singer. An efficient boosting algorithm for combining preferences. *Journal of Machine Learning Research (JMLR)*, 4:933–969, 2003. ISSN 1533-7928.

[39] Nir Friedman. Learning belief networks in the presence of missing values and hidden variables. In *Proceedings of the Fourteenth International Conference on Machine Learning*, ICML '97, pages 125–133, San Francisco, CA, USA, 1997. Morgan Kaufmann Publishers Inc.

[40] Nir Friedman. The bayesian structural em algorithm. In *The 14th Conference on Uncertainty in Artificial Intelligence*, UAI '98, Madison, Wisconsin, 1998.

[41] Jason Fulman. The combinatorics of biased riffle shuffles. *Combinatorica*, 18(2):173–184, 1998.

[42] Giorgio Gallo, Giustino Longo, Stefano Pallottino, and Sang Nguyen. Directed hypergraphs and applications. *Discrete Applied Mathematics*, 42(2-3), 1993.

[43] Oleg Gleizer and Alexander Postnikov. Littlewood-richardson coefficients via yang-baxter equation. *International Mathematics Research Notices*, (14):741–774, 2000.

[44] Claire Gormley and Brendan Murphy. A latent space model for rank data. In *Proceedings of the 2006 conference on Statistical network analysis*, ICML'06, pages 90–102, Berlin, Heidelberg, 2007. Springer-Verlag.

[45] Ulf Grenander. *Probabilities on Algebraic Structures*. Wiley, 1963.

[46] Mohinder S. Grewal and Angus P. Andrews. *Kalman Filtering: Theory and Practice Using MATLAB*. John Wiley and Sons, Inc., Canada, 2001.

[47] Leonidas J. Guibas. The identity management problem — a short survey. In *Proceedings of the International Conference on Information Fusion*, 2008.

[48] John Guiver and Edward Snelson. Bayesian inference for plackett-luce ranking models. In *Proceedings of the 26th Annual International Conference on Machine Learning*, ICML '09, pages 377–384, New York, NY, USA, 2009. ACM.

[49] David P. Helmbold and Manfred K. Warmuth. Learning permutations with exponential weights. In *The Twentieth Annual Conference on Learning Theory*, COLT 2007, 2007.

[50] Jonathan Herlocker, Joseph A. Konstan, Al Borchers, and John Riedl. An algorithmic framework for performing collaborative filtering. In *Proceedings of the 22nd annual international ACM SIGIR conference on Research and development in information retrieval*, SIGIR '99, pages 230–237, New York, NY, USA, 1999. ACM.

[51] Klaus U. Hoffgen. Learning and robust learning of product distributions. In *Computational Learning Theory*, pages 77–83, 1993.

[52] Susan Holmes. Phylogenies: an overview. *IMA series, Statistics and Genetics*, 112:81–119, 1999.

[53] Susan Holmes. Bootstrapping phylogenetic trees: theory and methods. *Statistical Science*, 18(2):241–255, 2003.

[54] Jonathan Huang and Carlos Guestrin. Riffled independence for ranked data. In Yoshua Bengio, Dale Schuurmans, John Lafferty, Chris K. I. Williams, and Aron Culotta, editors, *Advances in Neural Information Processing Systems 22*, NIPS '08, pages 799–807. 2009.

[55] Jonathan Huang and Carlos Guestrin. Learning hierarchical riffle independent groupings from rankings. In *Proceedings of the 27th Annual International Conference on Machine Learning*, ICML '10, pages 455–462, Haifa, Israel, 2010.

[56] Jonathan Huang and Carlos Guestrin. Uncovering the riffled independence structure of ranked data. *http://arxiv.org/abs/1006.1328*, 2011. URL http://arxiv.org/abs/1006.1328.

[57] Jonathan Huang, Carlos Guestrin, and Leonidas Guibas. Efficient inference for distributions on permutations. In J.C. Platt, D. Koller, Y. Singer, and S. Roweis, editors, *Advances in Neural Information Processing Systems 20*, NIPS '07, pages 697–704. MIT Press, Cambridge, MA, 2008.

[58] Jonathan Huang, Carlos Guestrin, and Leonidas Guibas. Inference for distributions over the permutation group. Technical Report CMU-ML-08-108, Machine Learning Department, Carnegie Mellon University, May 2008.

[59] Jonathan Huang, Carlos Guestrin, and Leonidas J. Guibas. Fourier theoretic probabilistic inference over permutations. *Journal of Machine Learning Research (JMLR)*, 10:997–1070, 2009.

[60] Jonathan Huang, Carlos Guestrin, Xiaoye Jiang, and Leonidas J. Guibas. Exploiting probabilistic independence for permutations. *Journal of Machine Learning Research - Proceedings Track*, 5:248–255, 2009.

[61] Jonathan Huang, Ashish Kapoor, and Carlos Guestrin. Efficient probabilistic inference with partial ranking queries. In *The 27th Conference on Uncertainty in Artificial Intelligence*, UAI '11, Barcelona, Spain, July 2011.

[62] Mark Huber and Jenny Law. Fast approximation of the permanent for very dense problems. In *Proceedings of the 19th annual ACM-SIAM symposium on Discrete algorithms*, SODA '08, pages 681–689, Philadelphia, PA, USA, 2008. Society for Industrial and Applied Mathematics.

[63] Yuri Ivanov, Alexander Sorokin, Christopher Wren, and Ishwinder Kaur. Tracking people in mixed modality systems. Technical Report TR2007-11, MERL: Mitsubishi Electronic Laboratories, 2007.

[64] Srikanth Jagabathula and Devavrat Shah. Inferring rankings under constrained sensing. In Daphne Koller, Dale Schuurmans, Yoshua Bengio, and Leon Bottou, editors, *Advances in Neural Information Processing Systems 21*, pages 753–760. 2009.

[65] Gordon James and Adelbert Kerber. *The Representation Theory of the Symmetric Group*. Addison-Wesley, 1981.

[66] Xiaoye Jiang, Lek-Heng Lim, Yuan Yao, and Yingyu Ye. Learning to rank with combinatorial hodge theory. *CoRR*, abs/0811.1067, 2008.

[67] Xiaoye Jiang, Jonathan Huang, and Leonidas Guibas. Fourier-information duality in the identity management problem. In *The European Conference on Machine Learning and Principles and Practice of Knowledge Discovery in Databases (ECML 2011)*, ECML '11, Athens, Greece, September 2011.

[68] Thorsten Joachims. Optimizing search engines using clickthrough data. In *Proceedings of the eighth ACM SIGKDD international conference on Knowledge discovery and data mining*, KDD '02, pages 133–142, New York, NY, USA, 2002. ACM.

[69] Ramakrishna Kakarala. A signal processing approach to fourier analysis of ranking data: The importance of phase. *IEEE Transactions on Signal Processing*, 59(4):1518–1527, 2011.

[70] Toshihiro Kamishima. Nantonac collaborative filtering: recommendation based on order responses. In *Proceedings of the ninth ACM SIGKDD international conference on Knowledge discovery and data mining*, KDD '03, pages 583–588, New York, NY, USA, 2003. ACM.

[71] Maurice Kendall. A new measure of rank correlation. *Biometrika*, 30, 1938.

[72] Zia Khan, Tucker Balch, and Frank Dellaert. Mcmc data association and sparse factorization updating for real time multitarget tracking with merged and multiple measurements. *IEEE Pattern Analysis and Machine Intelligence (PAMI)*, 28(12), 2006.

[73] Allen Knutson and Terence Tao. The honeycomb model of $\mathfrak{gl}_n(\mathbb{C})$ tensor products i: Proof of the saturation conjecture. *Journal of the American Mathematical Society*, 12(4):1055–1090, 1999.

[74] Daphne Koller and Nir Friedman. *Probabilistic Graphical Models: Principles and Techniques*. MIT Press, 2009.

[75] Risi Kondor. $\mathbb{S}_n$`ob`: a C++ library for fast Fourier transforms on the symmetric group, 2006. Available at `http://www.cs.columbia.edu/~risi/Snob/`.

[76] Risi Kondor. The skew spectrum of functions on finite groups and their homogeneous spaces, 2007. `http://arXiv.org/abs/0712.4259`.

[77] Risi Kondor. *Group theoretical methods in machine learning*. PhD thesis, Columbia University, 2008.

[78] Risi Kondor. A fourier space algorithm for solving quadratic assignment problems. In *Proceedings of the Twenty-First Annual ACM-SIAM Symposium on Discrete Algorithms*, SODA '10, pages 1017–1028, Philadelphia, PA, USA, 2010. Society for Industrial and Applied Mathematics.

[79] Risi Kondor and Karsten M. Borgwardt. The skew spectrum of graphs. In *Proceedings of the 25th international conference on Machine learning*, ICML '08, pages 496–503, Helsinki, Finland, 2008.

[80] Risi Kondor, Andrew Howard, and Tony Jebara. Multi-object tracking with representations of the symmetric group. In Marina Meila and Xiaotong Shen, editors, *Proceedings of the Eleventh International Conference on Artificial Intelligence and Statistics March 21-24, 2007, San Juan, Puerto Rico*, volume Volume 2 of JMLR: W&CP of *AISTATS '07*, March 2007.

[81] Risi Imre Kondor and Marconi S. Barbosa. Ranking with kernels in fourier space. In Adam Tauman Kalai and Mehryar Mohri, editors, *The 23rd Conference on Learning Theory, Haifa, Israel*, COLT '10, pages 451–463, 2010.

[82] K.L. Kueh, T. Olson, D. Rockmore, and K.S. Tan. Nonlinear approximation theory on finite groups. Technical Report PMA-TR99-191, Department of Mathematics, Dartmouth College, 1999.

[83] Serge Lang. *Algebra*. Addison-Wesley, 1965.

[84] Guy Lebanon and John Lafferty. Cranking: Combining rankings using conditional probability models on permutations. In *Proceedings of the Nineteenth International Conference on Machine Learning*, ICML '02, pages 363–370, San Francisco, CA, USA, 2002. Morgan Kaufmann Publishers Inc.

[85] Guy Lebanon and John Lafferty. Conditional models on the ranking poset. In S. Thrun S. Becker and K. Obermayer, editors, *Advances in Neural Information Processing Systems 15*, NIPS '02, pages 415–422, Cambridge, MA, 2003. MIT Press.

[86] Guy Lebanon and Yi Mao. Non-parametric modeling of partially ranked data. In John C. Platt, Daphne Koller, Yoram Singer, and Sam Roweis, editors, *Advances in Neural Information Processing Systems 20*, NIPS '07, pages 857–864, Cambridge, MA, 2008. MIT Press.

[87] Jun Liu and Rong Chen. Sequential monte carlo methods for dynamic systems. *Journal of the American Statistical Association*, 93:1032–1044, 1998.

[88] Jonathan Leonard Long, Jonathan Huang, and Carlos Guestrin. Props | probabilistic reasoning on permutations toolbox, 2010. Available at `http://www.select.cs.cmu.edu/code/sntools/index.html`.

[89] David G. Lowe. Distinctive image features from scale-invariant keypoints. *International Journal of Computer Vision*, 60:91–110, 2004.

[90] Tyler Lu and Craig Boutilier. Learning mallows models with pairwise preferences. In *The 28th Annual International Conference on Machine Learning*, ICML '11, Bellevue, Washington, June 2011.

[91] Colin L. Mallows. Non-null ranking models. *Biometrika*, 44, 1957.

[92] John I. Marden. *Analyzing and Modeling Rank Data*. Chapman & Hall, 1995.

[93] Jerrold Marsden and Michael Hoffman. *Elementary Classical Analysis*. W.H. Freeman, 1993.

[94] David Maslen. The efficient computation of Fourier transforms on the symmetric group. *Mathematics of Computation*, 67:1121–1147, 1998.

[95] David K. Maslen, Michael E. Orrison, and Daniel N. Rockmore. Computing isotypic projections with the lanczos iteration. *SIAM J. Matrix Anal. Appl.*, 25(3):784–803, 2003. ISSN 0895-4798.

[96] Claire Mathieu and Warren Schudy. Yet another algorithm for dense max cut: go greedy. In *Proceedings of the nineteenth annual ACM-SIAM symposium on Discrete algorithms*, SODA '08, Philadelphia, PA, USA, 2008. Society for Industrial and Applied Mathematics.

[97] Marina Meila, Kapil Phadnis, Arthur Patterson, and Jeff Bilmes. Consensus ranking under the exponential model. Technical Report 515, University of Washington, Statistics Department, April 2007.

[98] Nicholas Metropolis, Arianna W. Rosenbluth, Marshall N. Rosenbluth, Augusta H. Teller, and Edward Teller. Equations of state calculations by fast computing machine. *Journal of Chemical Physics*, 21:1087–1091, 1953.

[99] Thomas P. Minka. Old and new matrix algebra useful for statistics, 1997. URL citeseer.ist.psu.edu/minka97old.html.

[100] Rajeev Motwani and Prabhakar Raghavan. Randomized algorithms. *ACM Computational Survey*, 28(1), 1996.

[101] James Munkres. Algorithms for the assignment and transportation problems. *Journal of the Society of Industrial and Applied Mathematics*, 5(1): 32–38, 1957.

[102] Francis D. Murnaghan. The analysis of the kronecker product of irreducible representations of the symmetric group. *American Journal of Mathematics*, 60(3):761–784, 1938.

[103] Katta G. Murty. An algorithm for ranking all the assignments in order of increasing cost. *Operations Research*, 16:682–687, 1968.

[104] Hari Narayanan. On the complexity of computing kostka numbers and littlewood-richardson coefficients. *Journal of Algebraic Combinatorics*, 24(3):347–354, 2006.

[105] Songhwai Oh and Shankar Sastry. A polynomial-time approximation algorithm for joint probabilistic data association. In *Proceedings of the American Control Conference*, Portland, Oregon, 2005.

[106] Songhwai Oh, Stuart Russell, and Shankar Sastry. Markov chain Monte Carlo data association for general multiple-target tracking problems. In *Proceedings of the IEEE International Conference on Decision and Control*, Atlantis, Paradise Island, Bahamas, 2004.

[107] M. H. Peel. Specht modules and the symmetric group. *Journal of Algebra*, 36:88–97, 1975.

[108] James Petterson, Tiberio Caetano, Julian McAuley, and Jin Yu. Exponential family graph matching and ranking. *CoRR*, abs/0904.2623, 2009.

[109] Aubrey B. Poore. Multidimensional assignment and multitarget tracking. In *Partitioning Data Sets*, volume 19, pages 169–196. DIMACS Series in Discrete Mathematics and Theoretical Computer Science, 1995.

[110] Ryan Prescott Adams and Richard S. Zemel. Ranking via Sinkhorn Propagation. *ArXiv e-prints*, June 2011.

[111] Lawrence Rabiner. A tutorial on hidden markov models and selected applications in speech recognition. *Proceedings of the IEEE*, 77(2):257–286, 1990.

[112] Donald B. Reid. An algorithm for tracking multiple targets. *IEEE Transactions on Automatic Control*, 6:843–854, 1979.

[113] Daniel N. Rockmore. The FFT: An algorithm the whole family can use. *Computing in Science and Engineering*, 02(1):60–64, 2000.

[114] Bruce E. Sagan. *The Symmetric Group*. Springer, April 2001. ISBN 0387950672.

[115] Brad Schumitsch, Sebastian Thrun, Gary Bradski, and Kunle Olukotun. The information-form data association filter. In *Proceedings of Conference on Neural Information Processing Systems (NIPS)*, NIPS '05, Cambridge, MA, 2006. MIT Press.

[116] Brad Schumitsch, Sebastian Thrun, Leonidas Guibas, and Kunle Olukotun. The identity management Kalman filter (imkf). In *Proceedings of Robotics: Science and Systems*, RSS '06, Philadelphia, PA, USA, August 2006.

[117] Jean-Pierre Serre. *Linear Representations of Finite Groups*. Springer-Verlag, 1977.

[118] Dafna Shahaf, Anton Chechetka, and Carlos Guestrin. Learning thin junction trees via graph cuts. *Journal of Machine Learning Research - Proceedings Track*, 5:113–120, 2009.

[119] Jianbo Shi and Jitendra Malik. Normalized cuts and image segmentation. In *Proceedings of the 1997 Conference on Computer Vision and Pattern Recognition (CVPR '97)*, CVPR '97, pages 731–, Washington, DC, USA, 1997. IEEE Computer Society.

[120] Jaewon Shin, Leonidas Guibas, and Feng Zhao. A distributed algorithm for managing multi-target identities in wireless ad-hoc sensor networks. In *Proceedings of the 2nd international conference on Information*

processing in sensor networks, IPSN'03, pages 223–238, Berlin, Heidelberg, 2003. Springer-Verlag.

[121] Jaewon Shin, Nelson Lee, Sebastian Thrun, and Leonidas Guibas. Lazy inference on object identities in wireless sensor networks. In *Proceedings of the 4th international symposium on Information processing in sensor networks*, IPSN '05, Piscataway, NJ, USA, 2005. IEEE Press.

[122] Mark Steyvers, Michael D. Lee, Brent Miller, and Pernille Hemmer. The wisdom of crowds in the recollection of order information. In *Advances in Neural Information Processing Systems, 22*, NIPS '09, Vancouver, Canada, December 2009. MIT Press.

[123] Francis E. Su. Making history by card shuffling. Math Fun Facts. `http://www.math.hmc.edu/funfacts`.

[124] Mingxuan Sun, Guy Lebanon, and Kevyn Collins-Thompson. Visualizing differences in web search algorithms using the expected weighted hoeffding distance. In *Proceedings of the 19th international conference on World wide web*, WWW '10, pages 931–940, New York, NY, USA, 2010. ACM.

[125] Peter Swerling. A proposed stagewise differential correction procedure for satellite tracking and prediction. Technical Report P-1292, RAND Corporation, January 1958.

[126] Michael Taylor, John Guiver, Stephen Robertson, and Tom Minka. Softrank: optimizing non-smooth rank metrics. In *Proceedings of the international conference on Web search and web data mining*, WSDM '08, pages 77–86, New York, NY, USA, 2008. ACM.

[127] Audrey Terras. *Fourier Analysis on Finite Groups and Applications*. London Mathematical Society, 1999.

[128] Sebastian Thrun. Particle filters in robotics. In *Proceedings of the 17th Annual Conference on Uncertainty in Artificial Intelligence (UAI)*, 2002.

[129] Louis Leon Thurstone. A law of comparative judgement. *Psychological Review*, 34:273–286, 1927.

[130] Ravi Vakil. A geometric littlewood-richardson rule. *Annals of Mathematics*, 164(2):371–422, 2006.

[131] Jack van Lint and Richard M. Wilson. *A Course in Combinatorics*. Cambridge University Press, 2001.

[132] Charles F. van Loan. The ubiquitous kronecker product. *Journal of Computational and Applied Mathematics*, 123(1-2):85–100, 2000. ISSN 0377-0427.

[133] Anatoly Vershik and Andrei Okounkov. A new approach to the representation theory of symmetric groups. ii. *Journal of Mathematical Sciences*, 131(2):5471–5494, 2006.

[134] Ryen White and Steven Drucker. Investigating behavioral variability in web search. In *Proceedings of the 16th international conference on World Wide Web*, WWW '07, Banff, Alberta, Canada, 2007. ACM.

[135] Kelly Wieand. Eigenvalue distributions of random permutation matrices. *The Annals of Probability*, 28(4):1563–1587, 2000.

[136] Alan Willsky. On the algebraic structure of certain partially observable finite-state markov processes. *Information and Control*, 38:179–212, 1978.

[137] Karl Wimmer. Agnostically learning under permutation invariant distributions. In *Proceedings of the 2010 IEEE 51st Annual Symposium on Foundations of Computer Science*, FOCS '10, pages 113–122, Washington, DC, USA, 2010. IEEE Computer Society.

[138] Hongyuan Zha, Xiaofeng He, Chris Ding, Horst Simon, and Ming Gu. Bipartite graph partitioning and data clustering. In *Proceedings of the 10th international conference on Information and knowledge management*, CIKM '01, pages 25–32, New York, NY, USA, 2001. ACM.