Complexity of Linear Summary Statistics

Micah Pedrick

Michael Orrison, Advisor

Nicholas Pippenger, Reader

Department of Mathematics

May, 2017

Abstract

Families of linear functionals on a vector space that are mapped to each other by a group of symmetries of the space have a significant amount of structure. This results in computational redundancies which can be used to make computing the entire family of functionals at once more efficient than applying each in turn. This thesis explores asymptotic complexity results for a few such families: contingency tables and unranked choice data. These are used to explore the framework of Radon transform diagrams, which promise to allow general theorems about linear summary statistics to be stated and proved.

Contents

List of Figures

List of Tables

Acknowledgments

I would like to thank my advisor Professor Orrison, for living up to the name by providing incredibly helpful advice. I would also like to thank Caitlin Lienkaemper for reading my drafts and making many helpful suggestions along the way. Finally, I would like to express my appreciation for the many friends who have supported me throughout this year.

Chapter 1

Introduction

Imagine we are birdwatchers. It's migration season, and we are recording the directions of birds we see flying overhead (up to some precision, perhaps taking intercardinal directions like northeast, but not north-northeast). After days of data collection, we sit down with our notebooks to try to understand the migration pattern for this time of year.

We first might want to get an idea for how many birds are passing over our observation camp overall. This is easy: we just add up the number of birds spotted traveling in each direction. Another natural question is whether there is a trend in the direction the birds are going—are they headed south for summer, just generally milling about, or have some other trend? We have recorded the intercardinal direction trends directly, but if we're more interested in just the trends in cardinal directions (suppressing the difference between south and southwest, for example), it's still pretty easy: we just need to add to the count in that cardinal direction the two adjacent intercardinals (e.g., summing the south, southwest, and southeast counts). Perhaps we want to know if there's a net trend along some axis—some birds go north, some go south, but is there a net flow along the axis or just a general outward trend? This can be determined by examining the difference between either opposite directions, or between the "broad trend" counts of adjacent directions.

So far as avian studies goes, this is all very convenient—a good deal of information lies in simple counts of bird sightings supplemented with addition and subtraction. However, we are not really birdwatchers. Recovering our interest in mathematics, we may wonder whether there is some mathematical content at play here more subtle than counting and arithmetic. We ask ourselves the question, mathematically speaking: "what are birds?"

For the purposes of this thesis, we are interested in viewing birds as the components of a data vector from a vector space indexed by a finite set, the collection of directions for which we have entries. Equivalently, these values could be the values of a function defined on that finite set—the value of the function on some element of the set corresponding to the component of the vector indexed by that element.

Our analysis of the situation can then be framed as the computation of various *summary statistics*, such as the overall count statistic "sum of all birds," the dimension-aggregating "broad trend" statistics, or the difference statistics of "net flow along a (physical) axis." In particular, we're interested in *linear summary statistics*, those which can be computed as linear functions of the data vector, or equivalently linear combinations of the values of the function at each point in the set. Each of the statistics described previously fits this description, which is mainly mathematically motivated by the simplicity of linear functions.

The computation of a linear summary statistic is quite simple: applying a linear function to a vector involves only multiplications and additions, and there are not really any ways to be clever with the computation—every operation is necessary. If we are trying to compute *several* statistics, however, we can be somewhat clever with how to compute them. Suppose we have a four-element vector, and are interested in the sum of all components, the sum of the first two component, and the sum of the last two components. We can see that there's a relationship between these, namely that the first statistic decomposes as the sum of the other two statistics. One could hope, and we shall see, that this relationship can be exploited to improve the computation of families of linear summary statistics. These kinds of statistics show up in determining conditions to analyze voting methods as in Fishburn (1984), contingency tables as in Chapter 4, choice data as in Lawson et al. (2006) and Chapter 5, and ranking data.

In Chapter 2, we explore complexity theory and asymptotic analysis, the theoretical apparatus we need to formalize "improving computation." In Chapter 3, we discuss Radon transforms, which provide a useful way of viewing the computations which allows for comparison between different types of data. In Chapters 4 and 5, we analyze two different kinds of statistics and data. Finally, in Chapter 6 we contemplate how these situations may be generalized, and future directions for research. A basic familiarity with real analysis and linear algebra is assumed for the content of this work; many references require a much stronger background in various areas, including functional analysis, algebraic geometry, and more advanced linear algebra.

Chapter 2

Complexity Theory and Asymptotic Analysis

2.1 Introduction to Complexity Theory

The study of *complexity theory* seeks to understand how difficult a problem is in terms of the use of *scarce resources*. A standard textbook on complexity theory is Papadimitriou (1994). It develops a rich theory of complexity for Turing machines, a typical theoretical model for computation, as well as formal equivalences between families of problems that establish *complexity classes* trying to capture ideas of "comparably resource-intensive problems." Our interest in complexity is at a lower, more particular level, trying to study differences among algorithms for computing families of linear summary statistics. Often, though these algorithms will fall within the same complexity class, they will have meaningfully different resource requirements.

2.1.1 Scarce Resources

A scarce resource is physically motivated as whatever the most important "cost" of performing some computation is. These are often either resources of space or of time. Space resources are things like the number of memory blocks a procedure requires or the number of logic gates or electrical components required for a circuit. Time resources are things like the number of basic operations a procedure requires or the length of the longest computation path through a circuit.

For an example problem, consider a method for determining whether an input of a five-letter word is a palindrome. A procedure for this involves

checking whether the first and last letters match, then checking whether the second and fourth letters match if the first and last do, answering *yes* if and only if both conditions are met.

If we take our basic operation as character comparison and assume that "remembering" a letter can be done by the program logic rather than with physical memory (by taking different branches of an `if` statement corresponding to each letter), this can be done with only the space required to hold the five characters of input and time required for two operations. If we imagine that we have a cursor that must be moved over the input, that the basic operations are moving the cursor as well as comparison, we need the time required to move from the first to the fifth position (four moves) for the first comparison, then to move to the fourth (one move) then back to the second (two moves) to make the second comparison, for a total of nine operations. Under this second model, however, the space requirements would remain the same.

2.1.2 Complexity Functions

Often, rather than simply computing the resource use (or as we'll call it subsequently, the complexity) of a particular instance of a problem, we'll consider the complexity as a function of certain parameters describing the problem. For example, rather than simply considering five-letter words, we can describe the complexity of determining whether an n-letter word is a palindrome. Under the first model, we can see that the time complexity is

$$C_1(n) = \left\lfloor \frac{n}{2} \right\rfloor,$$

as pairs of letters must be compared from the outside in (and if the number of letters is odd, the middle need not be compared to anything). Under the second model, the time complexity of this method of palindrome checking is

$$\begin{aligned} C_2(n) &= \left\lfloor \frac{n}{2} \right\rfloor + \sum_{\left\lceil \frac{n}{2} \right\rceil \leq m \leq n} [m-1] \\ &= \frac{1}{2}\left(n - \left\lceil \frac{n}{2} \right\rceil + 1\right)\left(n + \left\lceil \frac{n}{2} \right\rceil - 2\right), \end{aligned}$$

because in addition to the same comparisons as before, the cursor has to snake back and forth to the mth position or $(n-m)$th position in the string (starting with $m = n$ and working backward).

2.1.3 Arithmetic Linear Complexity

For the purpose of understanding linear summary statistics, we'll take as our scarce resource arithmetic operations, assigning a cost of one to each addition, subtraction, or scalar multiplication (by a quantity of magnitude not equal to one), motivated by the time required by a computer processor to carry out each operation. This will define the *arithmetic linear complexity* or *linear complexity* of applying some matrix M as the smallest number of operations required to apply that matrix to an arbitrary vector in its domain. We'll denote this quantity as $L(M)$, similarly to Clausen and Baum (1993). What we have called $L(M)$, Clausen and Baum would denote $L_\infty(M)$. As we are broadly speaking untroubled by the magnitude of multiplications, we will suppress the subscript. Although a more careful analysis may restrict the allowable unit-cost multiplications based on the hardware details of the computers which would implement these algorithms, the size of integers standardly representable makes this unlikely to impact even large cases of practical interest.

For example, to apply the matrix

$$M = \begin{bmatrix} 0 & 1 & 1 & 1 \\ 1 & 0 & 1 & 1 \\ 1 & 1 & 0 & 1 \\ 1 & 1 & 1 & 0 \end{bmatrix},$$

we could simply use two additions a piece to compute each of the four elements of the output vector; this establishes that

$$L(M) \leq 8.$$

However, if we first use three additions to compute the sum of all four elements, then use four subtractions to get to each of the components required to yield the four elements of the output vector, we get

$$L(M) \leq 7.$$

Further, if we realize that on the way to computing the sum of all four, we would have an intermediate stage with three summed elements which is one of the desired outputs, removing the need to perform that subtraction, we get

$$L(M) \leq 6.$$

As we can see, it is much easier to put bounds on the linear complexity of a matrix than to establish it exactly. When we wish to make arguments about the number of operations required to solve problems of interest, we will proceed as above, and give the complexity for some algorithm of solving the problem in particular, which establishes upper bounds for the complexity of solving the problem with *any* algorithm.

2.2 Asymptotic Analysis

It may be difficult to characterize how precisely the complexity of even a specific algorithm varies with the parameters of a problem. Even when it is possible to write an exact form for the complexity function, we would like to be able to discuss rigorously the idea of significant differences in complexity. For this purpose we turn to *asymptotic analysis*—studying the comparative behavior of functions as their parameters tend towards certain "special" values. Most often, we will consider parameters tending towards infinity, although in other contexts considering parameters tending towards zero or other values (perhaps at which functions share singular behavior) is insightful. A useful textbook is de Bruijn (1981), though this will deviate slightly from his treatment.

2.2.1 Big-O Notation

One of the mainstays of asymptotic analysis is the definition that a function $f(x)$ *is asymptotic to* a function $g(x)$ as $x \to X$ if

$$\limsup_{x \to X} \left| \frac{f(x)}{g(x)} \right| < \infty.$$

We denote this

$$f(x) = O(g(x)) \text{ as } x \to X,$$

although as most often X is infinity we will generally drop the additional text. Sometimes we may also say that $f(x)$ *is* $O(g(x))$.

This relationship has several rapidly apparent properties. It is a transitive relationship: if $f(x) = O(g(x))$ and $g(x) = O(h(x))$, then $f(x) = O(h(x))$, as

$$\limsup_{x \to X} \left| \frac{f(x)}{h(x)} \right| = \limsup_{x \to X} \left| \frac{f(x)}{g(x)} \frac{g(x)}{h(x)} \right| = \limsup_{x \to X} \left| \frac{f(x)}{g(x)} \right| \limsup_{x \to X} \left| \frac{g(x)}{h(x)} \right|.$$

It is scale-invariant in the sense that if $f(x) = O(g(x))$, then for all constants $c \neq 0$,

$$cf(x) = O(g(x)) \text{ and } f(x) = O(cg(x)).$$

If $f(x) = O(g(x))$ and $y(x) = O(g(x))$, then $f(x) + y(x) = O(g(x))$, as

$$\limsup_{x\to X}\left|\frac{f(x)+y(x)}{g(x)}\right| = \limsup_{x\to X}\left|\frac{f(x)}{g(x)}+\frac{y(x)}{g(x)}\right| = \limsup_{x\to X}\left|\frac{f(x)}{g(x)}\right|+\limsup_{x\to X}\left|\frac{y(x)}{g(x)}\right|.$$

With the scale-invariance, this means that an arbitrary linear combination of functions which are each $O(g(x))$ is also $O(g(x))$. Notably, though, the relationship is *not* symmetric, hence not an equivalence relation.

Among ordinary functions, there are three general categories with respect to the big-O relation: logarithmic, polynomial, and exponential functions. Logarithmic functions—and powers of logarithms—are asymptotic to polynomial functions, which are asymptotic to exponential functions. In general, applying the definition we see

$$\begin{aligned}\left(\log_b(x)\right)^p &= O(x^q) \text{ for all } q > 1,\\ x^p &= O(x^q) \text{ for all } 1 \leq p \leq q,\\ &= O(a^x) \text{ for all } a > 1,\\ a^x &= O(b^x) \text{ for all } 1 \leq a \leq b.\end{aligned}$$

From these and the properties above, we can see that polynomials $f(x) = a_p x^p + a_{p-1}x^{p-1} + \cdots + a_1 x + a_0$ may be described most simply in big-O terms by their order, as $f(x) = O(x^p)$. Similarly, exponentials may be described most simply by their largest base. Although we will not need it, it is an interesting consequence of the identity $\log_b(x) = \frac{\log_\beta(x)}{\log_\beta(b)}$ that the base of a logarithm makes no difference in big-O terms, and many authors simply write $f(x) = O(\log(x))$ for logarithmic functions, suppressing the base entirely.

2.2.2 Leading Term Analysis

The asymptotic to relationship essentially captures the idea of "not worse than." This makes it suitable for putting upper bounds on the complexity of a problem, as if we can exhibit an algorithm whose complexity function is $O(f(x))$, we know that the best algorithm will also be $O(f(x))$, as it can only improve on the exhibited one. However, it is a very coarse tool for comparing algorithms—it may be the case that two algorithms for the same

problem are both $O(2^x)$, but if one consistently requires only a third of the resources of the other as $x \to \infty$, that algorithm is clearly preferable, and we would like an analysis that can detect this.

Inspired by the hierarchy of functions in big-O terms, we define the relationship that $f(x)$ and $g(x)$ are *asymptotically equivalent* as $x \to X$ if

$$\lim_{x \to X} \frac{f(x)}{g(x)} = 1.$$

We will denote this as $f(x) \sim g(x)$. Similarly, we define the relationship that $f(x)$ is *asymptotically negligible with respect to* $g(x)$ as $x \to X$ if

$$\lim_{x \to X} \frac{f(x)}{g(x)} = 0.$$

We will denote this as $f(x) \ll g(x)$.

We can see that if both $f(x) \sim g(x)$ and $h(x) \ll g(x)$, then $f(x) + h(x) \sim g(x)$. This inspires *leading term analysis*—we will attempt to decompose a complexity function into a part asymptotically equivalent to some simple function, in practice a monomial or single exponential term, and a part asymptotically negligible to that simple function. We are then able to use the simpler function to compare to the complexities of other algorithms, confident that once the parameters grow sufficiently large their relationships will hold for the original functions as well.

For example, suppose that we have two algorithms for performing some task on n objects, the first with complexity $C_1(n) = n^2$, the second with complexity $C_2(n) = \frac{1}{2}n^2 + 30n$. We can see that $\lim_{n \to \infty} \frac{\frac{1}{2}n^2 + 30n}{\frac{1}{2}n^2} = 1$, so we can write $C_2(n) \sim \frac{1}{2}n^2$. Since $C_1(n)$ is already a monomial, we can easily see that $C_1(n) \sim n^2$; in order to compare the algorithms, rather than directly comparing C_1 and C_2, we compare n^2 and $\frac{1}{2}n^2$. We can pretty clearly see that, asymptotically speaking, the second algorithm requires about half of the resources of the first.

Of course, if we have small n, then the first algorithm actually has a smaller complexity. In this case, we can easily deal with the analytic forms, so we can tell that the first algorithm outperforms the second for $n \leq 60$, and that the ratio $\frac{C_2(n)}{C_1(n)}$ differs from the asymptotic ratio of $\frac{1}{2}$ by less than 0.01 for $n > 3000$. If the linear part of C_2 were larger, the points at which these changes occur would be even farther.

This makes the point that leading term analysis, and asymptotic analysis in general, doesn't make guarantees about the short-term behavior of the

functions it treats. However, when combined with experimental results for the short term—in the case of linear complexity, this could be computing the number of operations required for small-parameter cases—it allows us determine when, where observed, short term improvements will continue to hold.

Chapter 3

Radon Transforms

3.1 Introduction to Radon Transforms

One method of data analysis is determining the comparative amounts of data points in various subsets of the data space. When these subsets are physically meaningful, they can help play an interpretive role.

For an example, suppose we have access to the voting records of a hundred State Senators on tax increase bills. If we divide the Senators into several sets randomly and count the number of votes for increased taxes, normalized by the size of the set, we generally expect to get about equal values for every set. However, if we divide the Senators by political party and perform the same count, we are much more likely to get wildly different values for each party, assuming the parties have different stances on tax increase. If we happened to divide them into random sets several times, and in one case found radically different normalized vote counts, it would be reasonable to suspect that that division happened to be along party lines—or at least some significant ideological split.

3.1.1 The Continuous Radon Transform

This kind of procedure converts functions of points in the data space to functions on subsets of the data space. This procedure was first studied by Johann Radon in the context of converting from functions defined on the Euclidean plane to functions defined on arbitrary straight lines in the plane (Radon, 1986). This transformation, later known as the (*continuous*) *Radon transform*, takes the function $f(\vec{x})$ on points in $\mathbb{R}^2$ to the function $\mathcal{R}\left\{f\right\}(L)$

on lines in $\mathbb{R}^2$ such that

$$\mathcal{R}\left\{f\right\}(L) := \int_L f(\vec{s})\,\mathrm{d}\vec{s}.$$

The Radon transform is often used in the context of x-ray tomography (Quinto, 2006). The (continuous) Radon transform is explicitly invertible, so if a collection of densities integrated along various lines through a body is used to create an approximation of the Radon transform of the internal structure, this approximate Radon transform may be inverted to yield an approximation of the point density. This allows medical professionals to record the intensity of x-ray beams passed through a patient at different offsets and angles, and thereby reconstruct an accurate picture of the internal arrangement of bones and organs without surgical procedures.

Later authors have generalized the Radon transform in various ways. Integration along lines through higher-dimensional Euclidean space and integration along higher-dimensional affine subspaces have both been studied (e.g., in Ludwig (1966) and Gindikin (1998)). Similarly, analogues along different families of subsets of Euclidean space have been treated, such as the circular Radon transform (Ambartsoumian and Kuchment, 2005). The uniting theme among all of these variants is integration over a family of subsets of some particular interest.

3.1.2 The Discrete Radon Transform

A generalization of particular interest for our purposes is the *discrete Radon transform*. While the (continuous) Radon transform operates on functions of the plane or more generally Euclidean spaces, the discrete Radon transform unsurprisingly operates on discrete and usually finite sets.

For our initial treatment, we will proceed similarly to the thesis of Zentz (2008). Suppose we are given some finite set X, a function $f : X \to \mathbb{R}$, and a collection $\mathcal{S}$ of subsets of X imagined as a (not necessarily proper) subset of its power set $\mathcal{P}(X)$. Then we define the Radon transform $\mathcal{R}\left\{f\right\} : \mathcal{S} \to \mathbb{R}$ by

$$\mathcal{R}\left\{f\right\}(S) := \sum_{x \in S} f(x).$$

This is analogous to the (continuous) Radon transform with summation replacing integration and the collection $\mathcal{S}$ of subsets replacing the collection of lines in the plane. We will call it the Radon transform *between X and $\mathcal{S}$*.

For example, suppose we have a square with vertices A, B, C, and D. We

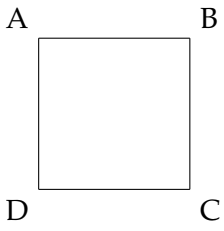

Figure 3.1 A Square.

can define the set $\mathcal{S}$ as the edges of the square, AB, BC, CD, and DA. Then the Radon transform of a function f defined on the vertices of the square is a function defined on its edges, whose value on an edge is the sum of the values of f on its endpoints.

3.2 Radon Transform Diagrams

In order to represent the Radon transform between X and $\mathcal{S}$, we define the concept of a *Radon transform diagram*. This is an undirected bipartite graph, with one class of vertices being the elements of X, the other class of vertices being elements of $\mathcal{S}$, and an edge between $x \in X$ and $S \in \mathcal{S}$ if $x \in S$. Drawing this diagram allows for a visual representation of a Radon transform. We will take the convention here that the elements of X are aligned on the left side of the diagram and the elements of $\mathcal{S}$ along the right side, but this is merely for consistency.

In our example of the square, the Radon transform diagram is then given by Figure 3.2.

Figure 3.2 Radon Transform Diagram for the Square.

Inspired by our concept of a Radon transform diagram, we will slightly modify our definition for the discrete Radon transform. In a Radon transform diagram, we have two sets—the set X and the "interesting subsets" $\mathcal{S}$—and an incidence relation between them—the membership relation $x \sim S$ if $x \in S$.

Rather than insist that the second set be a collection of subsets of X and the incidence relation be membership, we will simply suppose we have two sets X and Y with an incidence relation $\sim$ defined between them, and define the Radon transform of a function $f : X \to \mathbb{R}$ as the function $\mathcal{R}\left\{f\right\} : Y \to \mathbb{R}$ such that

$$\mathcal{R}\left\{f\right\}(y) := \sum_{x \sim y} f(x).$$

This definition is entirely analogous to the previous definition, as we could choose the set $\mathcal{S}_Y := \left\{S_y : y \in Y\right\} \subseteq \mathcal{P}(X)$, where

$$S_y := \left\{x \in X : x \sim y\right\}.$$

However, using it allows us to more easily discuss Radon transforms onto collections of "subobjects" more sophisticated than subsets. For example, we could take X to be the set of connected graphs on n labeled vertices, and Y to be the set of trees on n labeled vertices, with $x \sim y$ if y is a subgraph of x, which is a far more natural description to follow than reasoning out the relationship between graphs that share a particular subtree.

3.3 Arithmetic Linear Complexity of Discrete Radon Transforms

Equipped with a definition for discrete Radon transforms and their diagrams, we are prepared to explore their complexity. Suppose that we order X as $(x_1, x_2, \ldots, x_n)$ and Y as $(y_1, y_2, \ldots, y_m)$. Then given a function $f : X \to \mathbb{R}$, we define a vector $\vec{v}_f \in \mathbb{R}^n$ such that

$$\vec{v}_f := \begin{bmatrix} f(x_1) \\ f(x_2) \\ \vdots \\ f(x_n) \end{bmatrix}.$$

For a fixed y, the Radon transform $\mathcal{R}\left\{f\right\}(y)$ is a linear combination of $\left\{f(x_1), f(x_2), \ldots, f(x_n)\right\}$ with coefficients either zero or one. Thus, there exists an $m \times n$ matrix R with entries zero and one such that

$$\vec{w}_f := \begin{bmatrix} \mathcal{R}\left\{f\right\}(y_1) \\ \mathcal{R}\left\{f\right\}(y_2) \\ \vdots \\ \mathcal{R}\left\{f\right\}(y_m) \end{bmatrix} = R\vec{v}_f.$$

This matrix R depends only on X and Y, and encodes the computation of the Radon transform of any function $f : X \to \mathbb{R}$. We will call R the *analysis matrix* for the Radon transform from X to Y, and define the *arithmetic linear complexity of the Radon transform* as $L(R)$. Happily, we are able to relate the analysis matrix of a Radon transform to the diagram of that transform: R is precisely the incidence matrix for the diagram whose rows are indexed by Y and columns are indexed by X, where by convention we will order both sets as they are written in the diagram.

For our example of the square then, with Radon transform diagram given in Figure 3.2 and with X and Y ordered lexicographically, the analysis matrix is

$$R = \begin{bmatrix} 1 & 1 & 0 & 0 \\ 0 & 1 & 1 & 0 \\ 0 & 0 & 1 & 1 \\ 1 & 0 & 0 & 1 \end{bmatrix}.$$

There exists a naïve algorithm for applying these analysis matrices, namely computing each linear function $\mathcal{R}\left\{f\right\}(y)$ independently. This allows us to place an upper bound on the complexity of the Radon transform, easiest to state in terms of its diagram:

$$L(R) \leq \sum_{y \in Y} \left[\deg(y) - 1\right] \leq m(n-1)$$

as there are m functions to compute, each summing at most n terms and hence requiring at most $(n-1)$ additions. The rest of our efforts will be devoted to exploring cases where structure in Y gives rise to redundancy in the functions $\mathcal{R}\left\{f\right\}(y)$ which we can exploit to put tighter bounds on this complexity.

Chapter 4

Contingency Table Data

4.1 Binary Contingency Table Data

Suppose that we are contracted by a clothing store to help them determine how to stock their inventory: in particular, they want to know what customers seem to be looking for in T-shirts. The store has four ways to classify T-shirts—whether or not it has a fancy pattern, whether or not it is brightly colored, whether or not it has a vee neck, and whether or not it has long sleeves—and records how many of each kind of shirt (e.g., patterned bright vee necks with short sleeves) are sold. This is an example of Binary Contingency Table data, where objects are classified according to p properties each of which can take on two distinguishable values, and our data is a vector of counts indexed by the 2^p possible classifications. The statistics we are interested in are q-property statistics: given q properties that we are interested in, aggregating the counts for all objects that are defined by a particular setting of those properties, irrespective of the values of the remaining $(p - q)$ properties.

These statistics are a nested family, where $q = p$ is the original data, and the $q = r$ case can be recovered from all $q > r$ cases (because one can aggregate in stages to recover the same information). This allows us to determine the relative influence of particular settings of q properties in determining the distribution. For the situation described above, it would allow us to determine if some collection of q properties best explained the sales—for example, it could be that patterned, brightly colored T-shirts are incredibly popular, regardless of neck and sleeve cut.

The problem we would like to consider is the arithmetic linear complexity of computing all q-statistics for a given value of p. This is realized

by computing the slate of all Radon transforms from p-classifications to q-classifications, with incidence determined by agreement among the q specified values. We'll denote the analysis matrix for the Radon transform for p and q as $M_{p,q}$, so the arithmetic linear complexity of each Radon transform is a function

$$A(p, q) := L(M_{p,q}).$$

We'll denote the complexity of computing the entire slate with $0 \leq q \leq p$ as the function $\bar{A}(p)$, noting

$$\bar{A}(p) \leq \sum_{0 \leq q \leq p} L(M_{p,q}).$$

For the T-shirt example above, the relevant Radon transforms have diagrams isomorphic to those given in Figure 4.1. The labels have been given in the form of binary vectors representing presence or absence of each of the four properties (pattern, bright colors, vee neck, and long sleeves), and there are $\binom{4}{q}$ copies of the diagram for each q, corresponding to which q of the properties are being specified. Here, we take the last q properties.

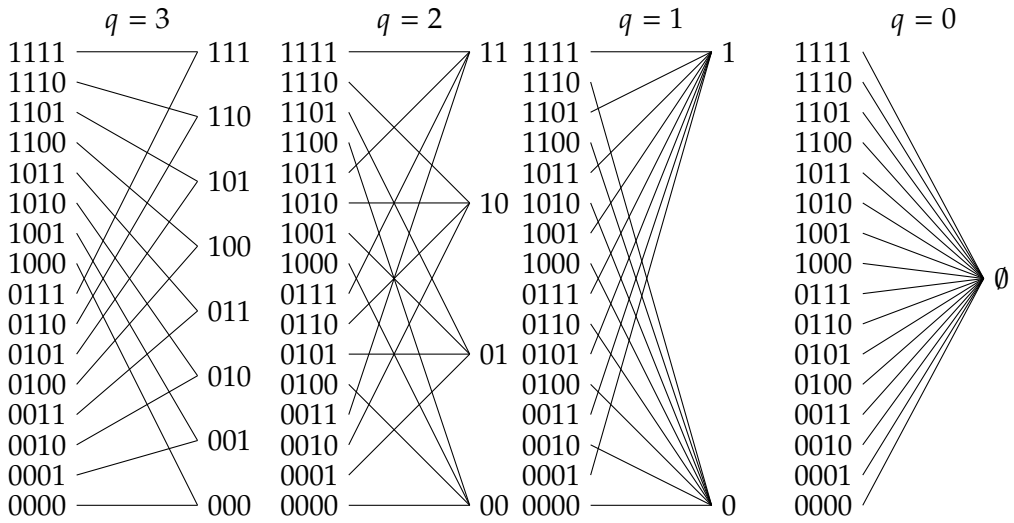

Figure 4.1 T-Shirt Sales Radon Transform Diagrams.

4.2 The Naïve Algorithm

For each of these Radon transforms, the dimension of the statistics space is $\binom{p}{q}2^q$, from selecting which properties to specify, and specifying a value for each. Each q-classification is contained in 2^{p-q} of the p-classifications, so each q statistic requires $2^{p-q} - 1$ additions to compute independently by summing over these. Thus, the naïve computation of the Radon transform onto q-statistics has arithmetic linear complexity

$$A_N(p, q) = \binom{p}{q}2^q \left[2^{p-q} - 1\right] = \binom{p}{q}\left[2^p - 2^q\right].$$

This gives an upper bound for the complexity of applying the matrix in general, as $A(p, q) \leq A_N(p, q)$, as the most efficient algorithm for applying it can do no worse than any particular algorithm. However, we hope and shall shortly show that this bound is not strict.

We are also interested in bounding $\bar{A}(p)$, the complexity for the whole slate of Radon transforms $0 \leq q \leq p$. In order to do this, naïvely we simply perform each transform in turn. This gives

$$\bar{A}_N(p) = \sum_{0 \leq q \leq p} A_N(p, q),$$

which can be solved as an application of the Binomial Theorem to see that

$$\bar{A}_N(p) = 4^p - 3^p.$$

4.3 The Cascade Algorithm

A significant performance boost can come from cascading the computations from high to low q. For any value of q, if we know the $(q + 1)$-classifications, we can recover the q-classifications at a cost of a single addition a piece by simply summing the two $(q + 1)$-statistics which match in all q specified properties. This is equivalent to factoring the analysis matrix as

$$M_{p,q} = M_{q+1,q} M_{q+2,q+1} \ldots M_{p-1,p-2} M_{p,p-1}.$$

Then by starting with $q = (p - 1)$—because $q = p$ is just the original data—we have the complexity of computing all of the statistics with this "cascade" algorithm as

$$\bar{A}_C(p) = \sum_{0 \leq q < p} \binom{p}{q}2^q,$$

which can be solved as

$$\bar{A}_C(p) = 3^p - 2^p.$$

This is an asymptotic improvement over $\bar{A}_N(p)$, as $\frac{\bar{A}_C(p)}{\bar{A}_N(p)} = O\left(\left(\frac{3}{4}\right)^p\right)$, which is a function which tends exponentially quickly to zero as p tends to infinity. That is, the complexity of the cascade algorithm is asymptotically negligible compared to the naïve algorithm. Direct computation shows that $\bar{A}_C(p)$ is less than 1% of $\bar{A}_N(p)$ for $p > 16$.

For the T-shirt example, we present a partial Radon transform diagram in Figure 4.2. As before, there is really more multiplicity than is shown in this diagram, as we have suppressed the different copies of each value of q corresponding to different subsets of q properties. The diagram instead shows the connections along the path eliminating always the furthest left property.

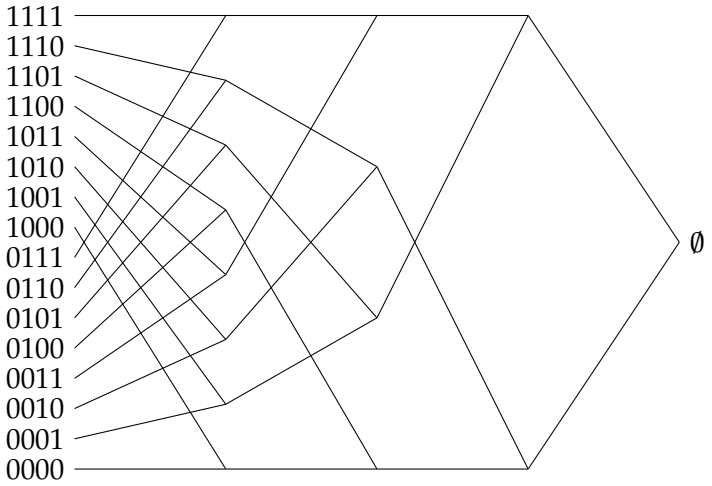

Figure 4.2 T-Shirt Sales Cascade Algorithm Radon Transform Diagram.

4.4 d-Ary Contingency Tables, and More

There is no reason why we must restrict ourselves to binary contingency tables. If we are considering properties each of which can take on some $d \geq 2$ number of values, our analysis is essentially unchanged. We have a statistics space of dimension $\binom{p}{q}d^q$, and each q-classification is contained in

d^{p-q} of the p-classifications, so the naïve application of the analysis matrix of the Radon transform diagram requires complexity

$$A_N(d,p,q) = \binom{p}{q} d^q \left[d^{p-q} - 1\right] = \binom{p}{q} \left[d^p - d^q\right].$$

Then the whole slate requires complexity

$$A_N(d,p) = \sum_{0 \le q \le p} A_N(d,p,q) = (2d)^p - (d+1)^p.$$

The Radon transform diagrams for this case are similar to those above, except each statistic has an in-degree of d^{p-q}, generalizing 2^{p-q}.

The cascade structure reduces each computation to $(d-1)$ additions when done in the correct order, for a complexity of the whole slate of

$$A_C(d,p) = \sum_{0 \le q < p} \binom{p}{q} d^q \left[d-1\right] = (d-1)\left[(d+1)^p - d^p\right].$$

The ratio of the cascade complexity to naïve complexity is then $O\left(d\left(\frac{d+1}{2d}\right)^p\right)$. We can see from this that as d becomes larger, we get a larger multiplicative factor, but also an increased rate of decay in p. This means that generally speaking contingency tables benefit more from the cascade algorithm both as the number of properties and the number of options for each increase.

We can also conceive of contingency tables whose properties can each take on different numbers of values. This slightly complicates our analysis, although by taking d as the largest number of possible values the preceding bound $A_C(d,p)$ may be used as an overestimate. Our Radon diagrams become more complicated, as now statistic in-degrees are not consistent among all q-statistics, and we may have difficulties cleanly expressing general forms for the complexities.

Chapter 5

Unranked Choice Statistics

5.1 Unranked Choice Data

Suppose at a certain dorm in a college, six candidates run for Dorm President, and the residents of the dorm are asked to choose three of them to win. This is an instance of what could be called Unranked Choice data, where respondents specify k-element subsets of an n-element "universe" (above, $k = 3$ and $n = 6$). We will in general only consider $k \leq \frac{n}{2}$, as if $k > \frac{n}{2}$, we could equivalently choose the $(n - k)$ elements absent from the list, so the $k \leq \frac{n}{2}$ case captures all of the interesting features of the data. The statistics we are interested in for this data are j-subset statistics: for each j-element subset of the "universe," how many times does that j-subset appear in a chosen subset, irrespective of the other $(k - j)$ elements of that subset.

This is a nested family of statistics, where $j = k$ recovers precisely the original data, and the $j = i$ case can be recovered from all $j > i$ cases (because one can sum over precomputed sums of subsets to recover the same information). It allows us to determine "j^{th} order effects," for example the relative importance of individuals, pairs, etc., in determining the composition of selected k-subsets. For something like the dorm president election, this would allow us to detect particularly popular (or unpopular) candidates, or perhaps even identify political parties or viewpoints by finding collections of candidates which are often selected together—representing the same position—or very rarely selected together—representing mutually opposing positions.

The problem we would like to consider is the arithmetic linear complexity of computing the slate of all j-statistics (for $j = 0, 1, \ldots, k$) for a given profile

of n and k. This is realized as computing the slate of all Radon transforms from k-subsets of the "universe" to j-subsets, with incidence determined by the subset relation. In order to do this, we will also consider the linear complexity of computing each Radon transform from k-subsets to j-subsets for fixed k and j. We'll denote the analysis matrix for the Radon transform from k-subsets of an n-set to j-subsets of that set as $M_{n,k,j}$, so that the complexity of each Radon transform is a function

$$A(n, k, j) := L(M_{n,k,j}).$$

We'll also denote the arithmetic linear complexity of the entire slate of Radon transforms as the function $\bar{A}(n, k)$, noting that

$$\bar{A}(n, k) \leq \sum_{0 \leq j \leq k} L(M_{n,k,j}).$$

Returning to the example of the Dorm President election, the relevant Radon transforms have diagrams given in Figure 5.1, where we have chosen a letter to represent each of the n candidates.

5.2 The Naïve Algorithm

For each of these Radon transforms, the dimension of the statistics space is $\binom{n}{j} = O(n^j)$. Each j-subset is a subset of $\binom{n-j}{k-j} = O(n^{k-j})$ of the k-subsets, so each j statistic requires $\binom{n-j}{k-j} - 1$ additions to compute independently. Thus, we can define the arithmetic linear complexity $A_N(n, k, j)$ of computing the entire Radon transform with a naïve algorithm of applying each row in turn as

$$A_N(n, k, j) = \binom{n}{j}\left[\binom{n-j}{k-j} - 1\right] = O(n^k).$$

This provides an upper bound for the complexity of applying the matrix in general, $A(n, k, j) \leq A_N(n, k, j)$. As with contingency table data, we shall show in following sections that this bound is not tight.

We are also interested in the complexity of computing the whole slate of Radon transforms. In order to do this, naïvely we would simply sum the complexities for each $j = 0, 1, \ldots, k$. This gives

$$\bar{A}_N(n, k) = \sum_{0 \leq j \leq k} A_N(n, k, j).$$

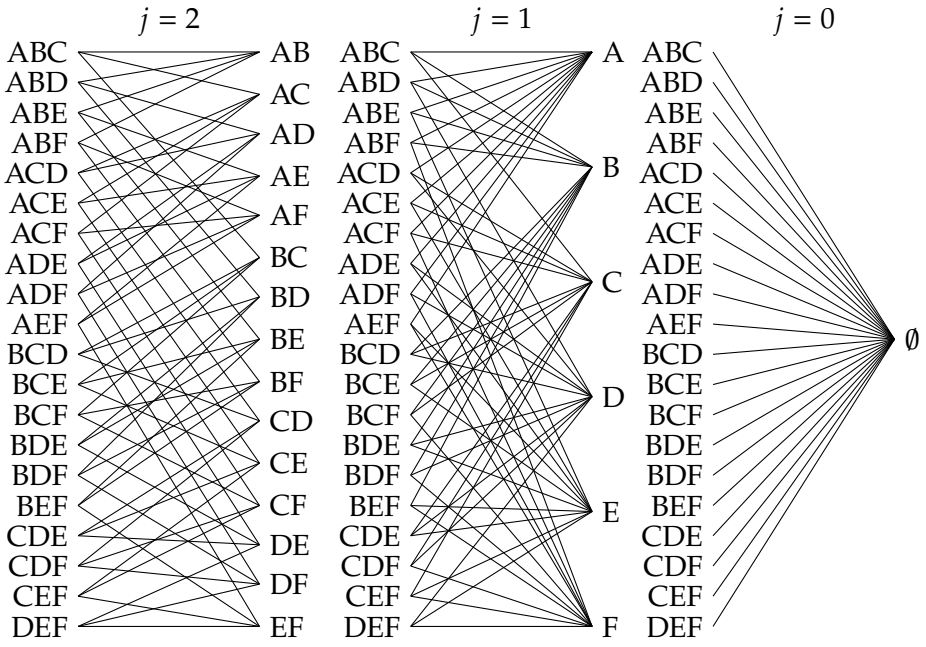

Figure 5.1 Dorm President Election Radon Transform Diagrams.

Computer algebra systems can execute this sum; however, the result is in terms of hypergeometric functions, and is more unwieldy than insightful. Being primarily concerned with the asymptotic behavior of this solution, we turn to leading term analysis.

Since $\binom{p}{q} \sim \frac{1}{q!}p^q$, we observe that

$$A_N(n,k,j) = \frac{n^k}{j!(k-j)!} + O(n^{k-1})$$

for $0 \leq j < k$ (and $A_N(n,k,j) = 0$ for $j = k$). With this, we can write

$$\bar{A}_N(n,k) = \sum_{0 \leq j < k} \left[\frac{1}{j!(k-j)!} n^k + O(n^{k-1}) \right] = \frac{2^k - 1}{k!} n^k + O(n^{k-1}).$$

The overall complexity of calculating all statistics for small parameter values is presented in Table 5.1 (bearing in mind that we are only interested in cases where $n \geq 2k$).

$k\backslash n$	4	6	8	10	12	14
2	13	38	75	124	185	258
3		118	355	784	1461	2442
4			957	2974	7126	14545
5				7426	23758	60591

Table 5.1 Naïve Algorithm Complexity $\bar{A}(n,k)$ for Unranked Choice Data.

For the example of the dorm president election, this involves applying row by row the collection of matrices

$$M_{6,3,0} = \begin{bmatrix} 1&1&1&1&1&1&1&1&1&1&1&1&1&1&1&1&1&1&1&1 \end{bmatrix},$$

$$M_{6,3,1} = \begin{bmatrix} 1&1&1&1&1&1&1&1&1&1&0&0&0&0&0&0&0&0&0&0 \\ 1&1&1&1&0&0&0&0&0&0&1&1&1&1&1&1&0&0&0&0 \\ 1&0&0&0&1&1&1&0&0&0&1&1&1&0&0&0&1&1&1&0 \\ 0&1&0&0&1&0&0&1&1&0&1&0&0&1&1&0&1&1&0&1 \\ 0&0&1&0&0&1&0&1&0&1&0&1&0&1&0&1&1&0&1&1 \\ 0&0&0&1&0&0&1&0&1&1&0&0&1&0&1&1&0&1&1&1 \end{bmatrix},$$

$$M_{6,3,2} = \begin{bmatrix} 1&1&1&1&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0 \\ 1&0&0&0&1&1&1&0&0&0&0&0&0&0&0&0&0&0&0&0 \\ 0&1&0&0&1&0&0&1&1&0&0&0&0&0&0&0&0&0&0&0 \\ 0&0&1&0&0&1&0&1&0&1&0&0&0&0&0&0&0&0&0&0 \\ 0&0&0&1&0&0&1&0&1&1&0&0&0&0&0&0&0&0&0&0 \\ 1&0&0&0&0&0&0&0&0&0&1&1&1&0&0&0&0&0&0&0 \\ 0&1&0&0&0&0&0&0&0&0&1&0&0&1&1&0&0&0&0&0 \\ 0&0&1&0&0&0&0&0&0&0&0&1&0&1&0&1&0&0&0&0 \\ 0&0&0&1&0&0&0&0&0&0&0&0&1&0&1&1&0&0&0&0 \\ 0&0&0&0&1&0&0&0&0&0&1&0&0&0&0&0&1&1&0&0 \\ 0&0&0&0&0&1&0&0&0&0&0&1&0&0&0&0&1&0&1&0 \\ 0&0&0&0&0&0&1&0&0&0&0&0&1&0&0&0&0&1&1&0 \\ 0&0&0&0&0&0&0&1&0&0&0&0&0&1&0&0&1&0&0&1 \\ 0&0&0&0&0&0&0&0&1&0&0&0&0&0&1&0&0&1&0&1 \\ 0&0&0&0&0&0&0&0&0&1&0&0&0&0&0&1&0&0&1&1 \end{bmatrix}.$$

5.3 Cascade Algorithm

A rather significant performance boost in computing the entire slate of transforms can come from cascading the computations from high j to low j. The j-statistics may be computed from the $(j+1)$ statistics by summing over all supersets of each j-subset, then dividing by $k(k-1)\cdots(k-j+1)$ to compensate for the multiple paths each input could take to reach a $(j+1)$-statistic. This is equivalent to factoring the analysis matrix as

$$M_{n,k,j} = M_{n,j+1,j}\frac{1}{k-j+1}M_{n,j+2,j+1}\cdots\frac{1}{k-1}M_{n,k-1,k-2}\frac{1}{k}M_{n,k,k-1}.$$

and observing that the other $M_{n,k,j}$ factor into multiples of the first few matrices applied in this factorization. Thus, we can apply the first part of $M_{n,k,1}$, record the result (as it's the result of a Radon transform we're interested in), then divide by the appropriate factor and apply the rest.

This doesn't change the complexity of applying a single Radon transform. However, it gives a different recurrence relation for the slate of all of them, defining the complexity $\bar{A}_C(n,k)$ with this cascade algorithm; rather than the sum iterating only over the final parameter, it couples k and j:

$$\bar{A}_C(n,k) = \sum_{1\le\kappa\le k}\left[A_N(n,\kappa,\kappa-1) + \binom{n}{\kappa-1}\right] = \sum_{1\le\kappa\le k}\binom{n}{\kappa-1}(n-\kappa+1)\,.$$

Thus, instead of summing k terms of $O(n^k)$ complexity, we have only one with a monomial of this degree. This means that $\bar{A}_C(n,k) = \frac{n^k}{(k-1)!} + O(n^{k-1})$.

We haven't made an asymptotic improvement to bounding $\bar{A}(n,k)$, since $\bar{A}_C(n,k)$ is still $O(n^k)$. However, the coefficient has dropped from $\frac{2^k-1}{k!}$ to $\frac{1}{(k-1)!}$; the ratio of leading terms of the cascade algorithm to naïve is $\frac{k}{2^k-1}$, which approaches zero rapidly for large k (the leading term of the cascade algorithm has a coefficient less than 1% that of naïve as soon as $k \ge 10$, for example). The overall arithmetic complexity for calculating all statistics for small parameter values is presented in Table 5.2.

We can see that even for n slightly above k, the cascade algorithm can significantly reduce the number of operations required to compute all of the statistics we seek (the value of $\bar{A}_C(14,5)$ from the cascade algorithm is about 25% of $\bar{A}_N(14,5)$).

$k\backslash n$	4	6	8	10	12	14
2	16	36	64	100	144	196
3		96	232	460	804	1288
4			512	1300	2784	5292
5				2560	6744	15302

Table 5.2 Cascade Algorithm Complexity $\bar{A}_C(n,k)$ for Unranked Choice Data.

In the dorm president example, this involves factoring

$$
\begin{aligned}
M_{6,3,0} =& M_{6,1,0} M_{6,2,1} M_{6,3,2} \\
=& \frac{1}{6} \begin{bmatrix} 1 & 1 & 1 & 1 & 1 & 1 \end{bmatrix} \cdot \\
& \cdot \begin{bmatrix}
1 & 1 & 1 & 1 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
1 & 0 & 0 & 0 & 0 & 1 & 1 & 1 & 1 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 1 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 1 & 1 & 1 & 0 & 0 & 0 \\
0 & 0 & 1 & 0 & 0 & 0 & 1 & 0 & 0 & 1 & 0 & 0 & 1 & 1 & 0 \\
0 & 0 & 0 & 1 & 0 & 0 & 0 & 1 & 0 & 0 & 1 & 0 & 1 & 0 & 1 \\
0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 1 & 0 & 0 & 1 & 0 & 1 & 1
\end{bmatrix} \cdot \\
& \cdot \begin{bmatrix}
1 & 1 & 1 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
1 & 0 & 0 & 0 & 1 & 1 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 1 & 0 & 0 & 1 & 0 & 0 & 1 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 1 & 0 & 0 & 1 & 0 & 1 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 1 & 0 & 0 & 1 & 0 & 1 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 1 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 1 & 1 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 1 & 0 & 1 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 1 & 1 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 1 & 1 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 1 & 0 & 1 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 1 & 1 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 1 & 0 & 0 & 1 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 1 & 0 & 1 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 1 & 1
\end{bmatrix}
\end{aligned}
$$

While each matrix is applied row by row, by working right to left and multiplying by an appropriate part of the constant each time every desired statistic is computed.

It is insightful to consider the Radon transform diagram of this factorization. Each node leading from the "initial dataset"—the $\binom{6}{3}$ subsets of $\{A, B, C, D, E, F\}$ of size 3—has an out-degree of 3 to the first middle layer, corresponding to the $\binom{6}{2}$ pairs of elements. Thus, each vote will be triply counted in the Radon transform—it contributes to three pairs. We want this for the pairs statistic, but in terms of creating a new dataset to Radon transform onto singletons, we will need to divide by three to prevent overcounting. Similarly, we will need to divide by two after taking pairs to singletons and recording the statistics of interest before adding each entry to determine the total number of votes. This accounts for the $\frac{1}{6}$ factor in the decomposition above, and in general shows how to divide up the numerical factors to get the right results—divide at each layer by the out-degree of the previous layer. For unranked choice data, each layer in the Radon transform diagram will have a constant out-degree, so this is possible.

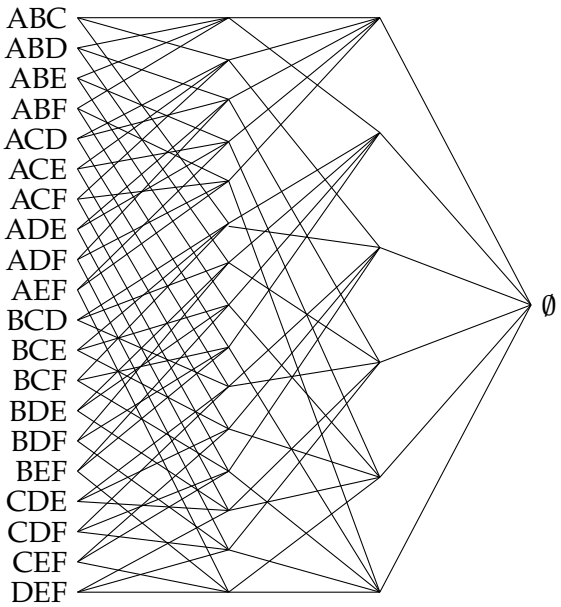

Figure 5.2 Cascade Algorithm Radon Transform Diagram ($n = 6, k = 3$).

5.4 Matrix Blocking and the $(j+1)$ Oracle

Cascading the computations improves the complexity of computing the entire slate of Radon transforms, but leaves the complexity of applying any one $M_{n,k,j}$ as the naïve method. It is natural to seek to improve our upper bound for the complexity $A(n,k,j)$ of applying a *single* Radon transform, in the hopes that this will provide even further improvement later. In order to do this, we examine the structure of the matrices $M_{n,k,j}$.

5.4.1 The Blocking Decomposition

When the indices are taken in lexicographical order, each analysis matrix can be divided into four principal blocks, corresponding to two classes of data indices—those components indexed by subsets containing the first element of the universe and not—and two classes of statistic indices —likewise. The upper right block is a zero matrix (statistics which include the first element do not depend on data from subsets which don't), while the others are analysis matrices with smaller indices, as prescribing the inclusion or exclusion of the first element effectively reduces the number of choices to be made in constructing the subsets. The decomposition is

$$M_{n,k,j} = \left[\begin{array}{c|c} M_{n-1,k-1,j-1} & 0 \\ \hline M_{n-1,k-1,j} & M_{n-1,k,j} \end{array}\right].$$

Of course, we could iterate the decomposition if our parameters are large enough (we don't want to go beyond $j = 0$ or $k = j$ or $n = k$, for example):

$$M_{n,k,j} = \left[\begin{array}{c|c|c|c} M_{n-2,k-2,j-2} & 0 & 0 & 0 \\ \hline M_{n-2,k-2,j-1} & M_{n-2,k-1,j-1} & 0 & 0 \\ \hline M_{n-2,k-2,j-1} & 0 & M_{n-2,k-1,j-1} & 0 \\ \hline M_{n-2,k-2,j} & M_{n-2,k-1,j} & M_{n-2,k-1,j} & M_{n-2,k,j} \end{array}\right].$$

In doing this, we see that the matrix $M_{n-2,k-2,j-1}$ is applied to the first block of the data vector twice (in the first column second row and first column third row). If it were computed separately for one of these families of statistics, the values could be used in the other without recomputation.

Putting this observation another way, in general the upper left block of $M_{n,k,j+1}$ matches the lower right of $M_{n,k,j}$, and is applied to the same part of the data vector. If the results of applying this block (the first $\binom{n-1}{j}$ $(j+1)$-statistics) are available during the application of $M_{n,k,j}$, they could

be used with a cost of only one addition a piece (for combining with the lower right block). We're thus interested in the problem of applying a particular $M_{n,k,j}$ given an *oracle* for the partial computation of all subblocks of the bottom left block—that is, supposing we can use the results of any computations that would have occurred within that block at no arithmetic cost (as if they were given to us by an oracle). Then we could apply the lower left block of a given matrix, and use the computations in its upper left subblock as our oracle for applying the upper left block of the full matrix.

5.4.2 Oracle Assisted Complexity

It's easy to see from the decomposition that applying the full matrix $M_{n,k,j}$ consists of applying the upper left block to the first part of the data vector, applying the lower right block to the second part of the data vector, then combining the results of the lower right computation with the oracular information about the lower left computation. There are $\binom{n-1}{j}$ rows in the lower part of the matrix, so it will require this many additions to do the combination. Thus, we can write a recurrence relation for the *oracle-assisted complexity* $A'_B(n,k,j)$ of applying $M_{n,k,j}$ for $j > 1$,

$$A'_B(n,k,j) = A'_B(n-1,k-1,j-1) + A'_B(n-1,k,j) + \binom{n-1}{j}.$$

For $j = 1$, the upper left block of $M_{n,k,j}$ is a row vector of ones; we can't recurse beyond this point, but we *can* use the information from the oracle to apply this row vector comparatively efficiently. The lower left block will also have $j = 1$, so its upper left subblock will also be a row vector of ones, and its lower right subblock will have the same structure of ones as well. This gives us a "staircase" of row vectors of ones running down the lower left block of $M_{n,k,1}$, such as the bolded ones in $M_{6,3,1}$:

$$\begin{bmatrix}
1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
\mathbf{1} & \mathbf{1} & \mathbf{1} & \mathbf{1} & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 1 & 1 & 1 & 1 & 1 & 0 & 0 & 0 & 0 \\
1 & 0 & 0 & 0 & \mathbf{1} & \mathbf{1} & \mathbf{1} & 0 & 0 & 0 & 1 & 1 & 1 & 0 & 0 & 0 & 1 & 1 & 1 & 0 \\
0 & 1 & 0 & 0 & 1 & 0 & 0 & \mathbf{1} & \mathbf{1} & 0 & 1 & 0 & 0 & 1 & 1 & 0 & 1 & 1 & 0 & 1 \\
0 & 0 & 1 & 0 & 0 & 1 & 0 & 1 & 0 & \mathbf{1} & 0 & 1 & 0 & 1 & 0 & 1 & 1 & 0 & 1 & 1 \\
0 & 0 & 0 & 1 & 0 & 0 & 1 & 0 & 1 & 1 & 0 & 0 & 1 & 0 & 1 & 1 & 0 & 1 & 1 & 1
\end{bmatrix}$$

The length of each of these row vectors is $mk - 2$ for $k - 2 \leq m \leq n - 2$, as they correspond to summing over subsets containing the first element

and another fixed element, so we must choose other elements to sum to k in the index subset. Thus, we can recover the upper row of $M_{n,k,1}$ by adding together the results of these $n-k+1$ computations, at a cost of $n-k$ operations. This replaces the $A'(n-1,k-1,j-1)$ term from the recurrence above, and $\binom{n-1}{1} = (n-1)$, so for $j=1$ we have the recurrence

$$\begin{aligned} A'_B(n,k,1) &= (n-k) + A'_B(n-1,k,1) + (n-1) \\ &= A'_B(n-1,k,1) + 2n - (k+1). \end{aligned}$$

We rewrite this as

$$A'_B(n,k,1) - A'_B(n-1,k,1) = 2n - (k+1)$$

then take advantage of the fact that this defines a telescoping sum to write

$$A'_B(n,k,1) = \sum_{k \le m \le n-1} [2m - (k-1)]$$

(even though we're not interested in *analyzing* situations with $n < 2k$, we need to use the complexity of applying the matrices which would apply to such situations down to $n = k$, whose matrix is an identity and does not further subdivide).

Wolfram Alpha gives that

$$\sum_{a \le m \le b} m^p = \frac{B_{p+1}(b+1) - B_{p+1}(a)}{p+1}$$

where $B_p(x) = \sum_{0 \le k \le n} \binom{n}{k} b_{n-k} x^k$ is a Bernoulli polynomial (the b_i are Bernoulli numbers). This means that we can analytically solve

$$\begin{aligned} A'_B(n,k,1) &= B_2(n) - B_2(k) + (n-k)(k-1) \\ &= n(n-k). \end{aligned}$$

With this analytic result, we can return to the $j > 1$ recurrence we described earlier, rewriting it as

$$A'_B(n,k,j) - A'_B(n-1,k,j) = A'_B(n-1,k-1,j-1) + \binom{n-1}{j}.$$

This describes a telescoping sum, so we can write

$$A'_B(n,k,j) = \sum_{k \le m \le n-1} \left[A'_B(m,k-1,j-1) + \binom{m}{j} \right].$$

Analytically solving this recurrence is possible in an iterative fashion, stepping up from previous j values to find the next highest. However, we can find an asymptotic expression for the leading term of $A'_B(n,k,j)$ all at once with leading term analysis.

Theorem 5.1 (Asymptotic Oracle-Assisted Complexity of $M_{n,k,j}$)**.** *The oracle-assisted linear complexity of applying the analysis matrix $M_{n,k,j}$ for j-statistics of (n,k)-data is asymptotically*

$$A'_B(n,k,j) = \frac{1}{j!}n^{j+1} + O(n^j).$$

Proof. Motivated by $A'_B(n,k,1)$, we make the hypothesis that

$$A'_B(n,k,j) = a'_{k,j}n^{j+1} + O(n^j).$$

Our form for $A'_B(n,k,1)$ implies that $a'_{k,1} = 1$ for all k. Then we have for $j > 1$

$$\begin{aligned} A'_B(n,k,j) = a'_{k,j}n^{j+1} + O(n^j) &= \sum_{k\le m\le n-1}\left[a'_{k-1,j-1}m^j + \frac{1}{j!}m^j + O(m^{j-1})\right] \\ &= \left[a'_{k-1,j-1} + \frac{1}{j!}\right]\frac{B_{j+1}(n) - B_{j+1}(k)}{j+1} + O(n^j). \end{aligned}$$

One can see from the formula that $B_p(x) = x^p + O(x^{p-1})$, so $B_{j+1}(n) - B_{j+1}(k) = n^{j+1} + O(n^j)$. Thus, we can equate the coefficients on the n^{j+1} terms on both side of the equation to yield the recurrence equation for coefficients

$$a'_{k,j} = \frac{a'_{k-1,j-1}}{j+1} + \frac{1}{(j+1)!}$$

where, again, $a'_{k,1} = 1$ for all k. This can be solved by Wolfram Alpha to give the solution

$$a'_{k,j} = \frac{1}{j!}.$$

□

5.4.3 Unassisted Complexity of j-Statistics

With this, we can consider the *unassisted* complexity of j-statistic analysis. Applying $M_{n,k,j}$, we must compute the lower left and lower right blocks unassisted by any oracles, but may compute the upper left with an oracle, because of the subcomputations involved in computing the lower left block. This means the *unassisted arithmetic complexity* recurrence equation for $j > 1$ is

$$A_B(n,k,j) = A'_B(n-1,k-1,j-1)+A_B(n-1,k-1,j)+A_B(n-1,k,j)+\binom{n-1}{j}.$$

As before, we have to deal with the $j = 1$ case separately, to get a base case for the above recurrence. Analogously to before, but with computation of the lower left block and not just use of its values, we get

$$A_B(n,k,1) = (n-k) + A_B(n-1,k-1,1) + A_B(n-1,k,1) + (n-1).$$

Here, as we have changes in both k and n, we will have to employ asymptotic analysis at this stage, with the idea that we will do so again for larger j.

For $k = 2$ (the first case of interest), we observe that the lower left subblocks will be identity matrices (since $M_{n,1,1} = I_n$ for all n), and require no arithmetic to compute. That is, $A_B(n,2,1) = (n-2)+A_B(n-1,2,1)+(n-1)$, or

$$A_B(n,2,1) = A_B(n-1,2,1) + 2n - 3.$$

Solving the resulting telescoping sum as before yields

$$A_B(n,2,1) = n(n-2).$$

That is, even without the oracle, $A_B(n,2,1) = A'_B(n,2,1)$. We use this in the recurrence relation above for $A_B(n,k,1)$ to solve for $A_B(n,3,1)$, yielding

$$\begin{aligned} A_B(n,3,1) &= (n-3) + A_B(n-1,2,1) + A_B(n-1,3,1) + (n-1) \\ &= A_B(n-1,3,1) + n^2 - 2n - 1 \\ &= \frac{1}{3}\left(n^3 - \frac{3}{2}n^2 - \frac{11}{2}n + 3\right), \end{aligned}$$

where in the last line we can use either the telescoping sum trick or simply computer algebra software and the initial condition $A_B(3,3,1) = 0$ (as $M_{3,3,1}$ is simply a column vector, and requires no arithmetic).

Notably, $A_B(n,3,1) = O(n^3)$ rather than the $O(n^2)$ of $A'_B(n,3,1)$—the absence of the oracle forces an asymptotically non-negligible increase in complexity. We can calculate the naïve arithmetic complexity of applying $M_{n,3,1}$ as $n\left[\binom{n-1}{2}-1\right] = \frac{1}{2}\left(n^3-3n^2\right)$. This too is $O(n^3)$, the same as with our computation order. *However*, we can see that the leading-term coefficient for $A_B(n,3,1)$ through our algorithm is smaller than that of the naïve algorithm; as n tends to infinity, the ratio between $A_B(n,3,1)$ and $A_N(n,3,1)$ will approach $\frac{2}{3}$.

This computation suggests that we've managed to make some gains in efficiency through the algorithm suggested by the block decomposition of the analysis matrices, and suggests that we should make the hypothesis that $A_B(n,k,1) = a_{k,1}n^k + O(n^{k-1})$—that the unassisted complexity is a polynomial whose degree depends on k, rather than j as before. To do so, we write the recurrence for $A(n,k,1)$ as a telescoping sum and iterate it down to a base case as usual, obtaining for $k > 1$

$$\begin{aligned} A_B(n,k,1) &= \sum_{k\le m\le n-1}\left[A_B(n-1,k-1,1)+2m-(k-1)\right] \\ a_{k,1}n^k + O(n^{k-1}) &= \sum_{k\le m\le n-1}\left[a_{k-1,1}n^{k-1}+O(n^{k-2})\right] \\ &= \frac{a_{k-1,1}}{k}n^k + O(n^{k-1}), \end{aligned}$$

at which point we may equate coefficients on the leading terms to get the recurrence relation for their coefficients

$$a_{k,1} = \frac{1}{k}a_{k-1,1}.$$

Using the base case $a_{2,1} = 1$ from our explicit form for $A_B(n,2,1)$ yields the solution

$$a_{k,1} = \frac{2}{k!}.$$

In order to do the same form of analysis for $j > 1$, we need a base case for each j. As it turns out, if $k = j+1$, then our algorithm does no better than naïve.

Lemma 5.1 (Arithmetic Linear Complexity of $M_{n,j+1,j}$)**.** *For all $j \ge 1$,*

$$A_B(n,j+1,j) = \binom{n}{j}\left(n-j-1\right) = A_N(n,j+1,j).$$

Proof. This can be shown inductively. For $j = 1$, we can see that

$$M_{n,2,1} = \begin{bmatrix} \mathbf{1} & \mathbf{0} \\ I_{n-1} & M_{n-1,2,1} \end{bmatrix}$$

where $\mathbf{1}$ is a row vector of ones and $\mathbf{0}$ is a row vector of zeros. The upper block must be computed naïvely, as the lower left cannot provide meaningful information (it is just an identity and has no helpful subcomputations). The lower left simply contributes $n - 1$ additions to fuse to the lower right block, which must be computed. The same structure holds for $M_{n-1,2,1}$, and so on, until we reach $M_{2,2,1}$, which is a column vector of ones, which requires no additions. At no point were we able to reuse any computations, so $M_{n,2,1}$ does not benefit from blocking and has the naïve computation time $n(n-2)$. For $j > 1$, if the statement holds for $j - 1$, then we observe that the upper and lower left blocks must be applied naïvely by hypothesis; the lower right block is another instance of the same form (with lower n), so the pattern continues until we reach a column vector, and conclude that $M_{n,j+1,j}$ must be computed naïvely as well. □

Using this, we are ready to prove an analogue of Theorem 5.1 for unassisted complexity:

Theorem 5.2 (Asymptotic Unassisted Linear Complexity of $M_{n,k,j}$)**.** *The linear complexity of applying the analysis matrix $M_{n,k,j}$ for j-statistics of Unranked Choice data without oracle assistance is*

$$A_B(n,k,j) = \frac{j+1}{k!}n^k + O(n^{k-1}).$$

Proof. We make the hypothesis that

$$A_B(n,k,j) = a_{k,j}n^k + O(n^{k-1}).$$

Lemma 5.1 implies that $a_{j+1,j} = \frac{1}{j!}$ for all j. Then we have for $k > j+1$

$$\begin{aligned} A_B(n,k,j) = a_{k,j}n^k + O(n^{k-1}) &= \sum_{k \le m \le n-1} \left[a'_{k-1,j-1}n^j + a_{k-1,j}n^{k-1} + \binom{m}{j} \right] \\ &= a_{k-1,j}\frac{B_k(n) - B_k(k)}{k} + O(n^{k-1}). \end{aligned}$$

Knowing $B_k(n) = n^k + O(n^{k-1})$, we can equate coefficients on the leading terms on both sides of the equation to yield the recurrence equation for coefficients

$$a_{k,j} = \frac{a_{k-1,j}}{k},$$

which with our initial condition of $a_{j+1,j} = \frac{1}{j!}$ implies that

$$a_{k,j} = \frac{j+1}{k!}.$$

□

For the dorm president example, and in particular the computation of the $j = 1$ statistics, the blocking algorithm corresponds to the factorization

$$M_{6,3,1} = \begin{bmatrix}
1&1&1&1&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0\\
0&0&0&0&1&1&1&0&0&0&0&0&0&0&0&0&0&0&0&0\\
0&0&0&0&0&0&0&1&1&0&0&0&0&0&0&0&0&0&0&0\\
0&0&0&0&0&0&0&0&0&0&1&1&1&0&0&0&0&0&0&0\\
0&0&0&0&0&0&0&0&0&0&0&0&0&1&1&0&0&0&0&0\\
0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&1&1&0&0\\
1&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0\\
0&1&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0\\
0&0&1&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0\\
0&0&0&1&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0\\
0&0&0&0&1&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0\\
0&0&0&0&0&1&0&0&0&0&0&0&0&0&0&0&0&0&0&0\\
0&0&0&0&0&0&1&0&0&0&0&0&0&0&0&0&0&0&0&0\\
0&0&0&0&0&0&0&1&0&0&0&0&0&0&0&0&0&0&0&0\\
0&0&0&0&0&0&0&0&1&0&0&0&0&0&0&0&0&0&0&0\\
0&0&0&0&0&0&0&0&0&1&0&0&0&0&0&0&0&0&0&0\\
0&0&0&0&0&0&0&0&0&0&1&0&0&0&0&0&0&0&0&0\\
0&0&0&0&0&0&0&0&0&0&0&1&0&0&0&0&0&0&0&0\\
0&0&0&0&0&0&0&0&0&0&0&0&1&0&0&0&0&0&0&0\\
0&0&0&0&0&0&0&0&0&0&0&0&0&1&0&0&0&0&0&0\\
0&0&0&0&0&0&0&0&0&0&0&0&0&0&1&0&0&0&0&0\\
0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&1&0&0&0&0\\
0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&1&0&0&0\\
0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&1&0&0\\
0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&1&0\\
0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&1
\end{bmatrix} \cdot$$

$$\cdot \begin{bmatrix}
1&1&1&0&0&0&0&0&0&0&0&0&0&0&0&1&0&0&0&0&0&0&0&0&0&0\\
0&1&0&0&0&0&1&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0\\
0&0&1&0&0&0&0&1&0&0&1&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0\\
0&0&0&0&0&0&0&0&1&0&0&1&0&1&0&1&0&0&0&0&0&0&0&0&0&0\\
0&0&0&0&0&0&0&0&0&1&0&0&1&0&1&1&0&0&0&0&0&0&0&0&0&0\\
0&0&0&1&1&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&1&0&0&0&0\\
0&0&0&1&0&1&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&1&0\\
0&0&0&0&1&1&0&0&0&0&0&0&0&0&0&0&1&0&0&0&0&0&0&0&0&1\\
0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&1&0&1&0&1&1&0&1&1&\\
0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&1&0&1&1&0&1&1&1&\\
1&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0
\end{bmatrix} \cdot$$

$$\cdot \begin{bmatrix}
1&0&0&0&0&0&0&0&0&0&0\\
0&0&0&0&0&1&0&0&0&0&1\\
0&1&0&0&0&0&1&0&0&0&0\\
0&0&1&0&0&0&0&1&0&0&0\\
0&0&0&1&0&0&0&0&1&0&0\\
0&0&0&0&1&0&0&0&0&1&0
\end{bmatrix}$$

5.4.4 Computing All Radon Transforms

With the asymptotic expressions from Theorems 5.1 and 5.2, we are ready to consider the problem of applying all of the Radon transforms for some fixed n and k, rather than simply a particular $M_{n,k,j}$. Lemma 5.1 shows us that the cascade algorithm will not benefit from the improved performance of the blocking algorithm over naïve application, as it factors $M_{n,k,1}$ into a product of matrices of the form $M_{n,j+1,j}$, whose blocking cost equals their naïve cost.

However, our contemplation of an oracle for $M_{n,k,j}$ given the computations of $M_{n,k,j+1}$ suggests a different approach. First we compute $M_{n,k,k-1}$, saving the results of the subcomputations in its upper left block. We then use these as an oracle for $M_{n,k,k-2}$, making its cost the oracle-assisted complexity, rather than the unassisted complexity; we again save the results of its upper left block subcomputations. We proceed in this manner, using each matrix to provide an oracle for the next, until we reach $M_{n,k,1}$ and have computed our whole slate of Radon transforms.

The linear complexity of all Radon transforms using this algorithm $\bar{A}_B(n,k)$ is then

$$\bar{A}_B(n,k) = A_B(n,k,k-1) + \sum_{1 \le j \le k-2} A'_B(n,k,j).$$

Since each $A'_B(n,k,j) = O(n^{j+1}) = O(n^{k-2})$, the only term with exponent k in $\bar{A}_B(n,k)$ is the leading term of $A_B(n,k,k-1)$. Thus, we have that

$$\bar{A}_B(n,k) = \frac{1}{(k-1)!} n^k + O(n^{k-1}).$$

To leading term, $\bar{A}_B(n,k) = \bar{A}_C(n,k)$. However, we compute the complexity for small parameter values, presenting it in Table 5.3.

$k \backslash n$	4	6	8	10	12	14
2	8	24	48	80	120	168
3		63	180	385	702	1155
4			368	1050	2384	4690
5				1925	5551	13230

Table 5.3 Blocking Algorithm Complexity $\bar{A}_B(n,k)$ for Unranked Choice Data.

We can see that the lower-order terms for low parameter values favor the blocking algorithm over the cascade algorithm, though not as dramatically

as cascade beats naïve (the complexity $\bar{A}_B(14,5)$ is better than $\bar{A}_C(14,5)$ by about 13.5%, rather than the nearly 75% improvement of cascade over $\bar{A}_N(14,5)$). Computing the difference $\bar{A}_B(n,k) - \bar{A}_C(n,k)$ for $2 \leq k \leq 5$ in Mathematica reveals that, in these cases, the difference is negative for all n of degree $k-1$. This is of course not enough to conclude that this holds in general, but it seems a fair and not difficult to verify conjecture that $\bar{A}_B(n,k) - \bar{A}_C(n,k) = O(n^{k-1})$, that it is negative at least for all $n \geq 2k$, and perhaps even that in particular $\bar{A}_B(n,k) - \bar{A}_C(n,k) = \frac{-1}{(k-1)!}n^{k-1} + O(n^{k-2})$ as suggested by the first few k.

Chapter 6

Conclusions and Future Work

6.1 Comparisons Between Types of Data

With two examples of linear summary statistics, we can examine the relative improvements we were able to make. For contingency table data, cascading the statistics gives a complexity asymptotically negligible compared to the naïve expression. For unranked choice data, both the cascade algorithm and the algorithm defined by the blocking structure do not give complexities asymptotically negligible to naïve; instead, they reduce the leading term coefficient, although somewhat dramatically. It should be noted that although from our descriptions in terms of parameters the complexities for contingency table data appear to be exponential and for unranked choice appear to be polynomial, when we consider them in terms of the number of dimensions in the vector space of data, both are low-degree polynomials for all parameter values—there are d^p dimensions for d-ary contingency tables with p properties, and $\binom{n}{k} = O(n^k)$ dimensions for unranked choice of k-subsets of an n-element universe.

Examining our expressions for the complexity in each case, we see that for contingency tables we have reduced the complexity of every term in the sum in the same asymptotically meaningful way. However, for unranked choice we reduce the complexity of successive terms in less and less meaningful ways until the originally largest term is not reduced at all. We can qualitatively see this in the Radon transform diagrams for the cascade algorithms as well: the diagram for contingency tables breaks down into nodes each of which branches only twice, while for unranked choice the complexity increases from $\emptyset$ through each layer back to the input.

This suggests a heuristic comparison that the unranked choice data is somehow more "tangled" than the contingency table data. One way of thinking of this is to consider (subsets of) automorphisms of the data and statistics spaces. These automorphisms have a group structure, depending on which ones we view as natural.

In contingency tables, for each positioning of q-statistics, the product of cyclic groups $\mathbb{Z}_2^q$ acts on the statistics by toggling each property, while a somewhat more complicated group changes which statistics are being specified. It can be thought of as an action of S_p on the labels, but it is not simply a permutation of the statistics because the labels of unspecified properties may be permuted without changing the statistic and change among the specified properties has already been handled. In unranked choice data, for each value of j, the only change is due to a more complicated group acting. Again, this is an action of a symmetric group which is not simply a permutation. This time we can think of S_n acting on the candidates, ignoring changes in the $n - j$ positions not in the statistic.

It seems reasonable to conjecture that these transformations are related to the computational redundancies which allow for the speedups we have observed. If so, then it further seems reasonable to conjecture that the simple transformations associated with contingency tables are related to the "better" (asymptotic rather than leading term coefficient) speedup they see. This would be something along the lines of an argument that these statistics are more similar to each other because the transformations are simpler, so they feature more redundancy.

6.2 The Role of Discrete Radon Transforms

In the preceding chapters, the formalism of discrete Radon transforms allowed for the visual representation of statistics and their computation through Radon transform diagrams. It has more promise than merely this, however, as it gives a way to talk about the general problem of computing linear statistics of some class of data, without specifying the structure of either statistics or data. By making gentle and general assumptions—for example, that the statistics all have the same in-degree, or some group acts transitively on the data—we can make and hopefully prove conjectures about linear statistics as a whole, rather than specific classes of them.

Likewise, it allows for the conversion of theorems about Radon transforms to be converted to statements about linear statistics. Much of the

literature on Radon transforms focuses on their invertibility (Ludwig, 1966; Ambartsoumian and Kuchment, 2005; Diaconis and Graham, 1985; Quinto, 2006). These statements can be used to determine when a family of statistics is complete enough to fully capture all features of a data set to reconstruct it. While the statistics considered here were essentially constructed as nested families starting with the data itself as the first member, this may not be true in general, or the description of the statistics may obscure this nested structure, so this is potentially valuable.

6.3 Extensions

For contingency table-type data, there does not seem to be much more to be done than deal with contingency tables each of whose properties is allowed to have a different number of possible values. Even this seems unlikely to give particular new insights, as cascading will still provide benefits as discussed at the end of Chapter 4, and this seems likely to be optimal.

There is more to be done for choice-type data. Fully-ranked choice data, where respondents provide a total ordering among the candidates, and partial ranking data, where respondents sort candidates into some number of classes of specified size (this encompasses unranked choice as described here, with one class of size k and one of size $n - k$), both describe voting situations and seem to be of interest (Diaconis, 1988). It is clear that these types of data have strong similarities to unranked choice, but are more complex, and it is not immediately clear how effective the blocking strategy would be, or even how to implement it.

Finally, there are other forms of data which do not seem to be precise analogues of the two considered here, but may be of sufficient interest to investigate in particular. These include the "birdwatching" statistics from the introduction, data defined on graphs or simplicial complexes with statistics summing across subgraphs or faces of the complex, or data drawn from other voting methods such as approval voting (like unranked choice, but where each respondent may select as many "winners" as they like).

Bibliography

Ambartsoumian, Gaik, and Peter Kuchment. 2005. On the injectivity of the circular Radon transform. *Inverse Problems* 21(2):473–485. URL http://stacks.iop.org/0266-5611/21/i=2/a=004.

Clausen, Michael, and Ulrich Baum. 1993. *Fast Fourier Transforms*. Wissenschaftsverlag.

de Bruijn, N. G. 1981. *Asymptotic Methods in Analysis*. Dover.

Diaconis, Persi. 1988. Group representations in probability and statistics. *Lecture Notes-Monograph Series* 11:i–192.

Diaconis, Persi, and RL Graham. 1985. The Radon transform on Z*. *Pacific J Math* 118(2):323–345.

Diaconis, Persi, Julia Salzman, et al. 2008. *Projection Pursuit for Discrete Data*. Institute of Mathematical Statistics.

Fishburn, Peter C. 1984. Discrete mathematics in voting and group choice. *SIAM Journal on Algebraic Discrete Methods* 5(2):263–275.

Gindikin, Simon. 1998. Real integral geometry and complex analysis. In *Integral Geometry, Radon Transforms and Complex Analysis*, 70–98. Springer.

Lawson, Brian L, Michael E Orrison, and David T Uminsky. 2006. Spectral analysis of the supreme court. *Mathematics Magazine* 79(5):340–346.

Ludwig, Donald. 1966. The Radon transform on Euclidean space. *Communications on Pure and Applied Mathematics* 19:49–81. URL http://onlinelibrary.wiley.com/doi/10.1002/cpa.3160190105/epdf.

Papadimitriou, Christos M. 1994. *Computational Complexity*. Reading, Massachusetts: Addison-Wesley.

Quinto, Eric Todd. 2006. An introduction to X-ray tomography and Radon transforms. *Proceedings of Symposia in Applied Mathematics* 63:1–23.

Radon, Johann. 1986. On the determination of functions from their integral values along certain manifolds. *IEEE Transactions on Medical Imaging* 5(4):170–176. doi:10.1109/TMI.1986.4307775.

Zentz, Andrea S. 2008. The finite Radon transform: an honors thesis [(HONRS 499)] .