Applications of Machine Learning: From Single Cell Biology to Algorithmic Fairness

by

Alexander H. S. Vargo

A dissertation submitted in partial fulfillment
of the requirements for the degree of
Doctor of Philosophy
(Mathematics)
in the University of Michigan
2020

Doctoral Committee:

Professor Anna C. Gilbert, Chair
Professor Daniel Burns
Professor Jun Z. Li
Associate Professor Indika Rajapakse
Assistant Professor Yuekai Sun

Alexander H. S. Vargo

ahsvargo@umich.edu

ORCID iD: 0000-0001-5930-8841

Dedication

For my family and friends.

Acknowledgments

First and foremost, I extend my deep gratitude to Anna Gilbert, my advisor. This dissertation would not have been possible without Anna's support, patience, and mathematical insight. Her guidance and perspective have illuminated my path from pupil to researcher.

Thanks also to Jun Li, who helped me understand the benefits of interdisciplinary collaboration. Jun's wisdom and anecdotes make research questions transcend any professional languages, and his constant questioning has assisted the development of my critical eye. I further greatly appreciate the collaboration of the other members of the Michigan Center for Single-Cell Genomic Data Analytics; our meetings and conversations were a highlight of my graduate experience.

In addition, many thanks to Yuekai Sun. Yuekai is always down to earth and relatable, but produces interesting observations at a shocking pace. He fueled my interest in algorithmic fairness, and is always open to working on another project. Thanks also to my other algorithmic fairness collaborators Martin Strauss, Suresh Venkatasubramanian, Laura Niss, Amanda Bower, and Fan Zhang. Special thanks to Laura and Amanda: our discussions together have resulted in some of the most productive and important insights of my career.

I am additionally grateful to the many others who have acted as advisors and mentors throughout my academic life. Thanks to my other committee members Daniel Burns and Indika Rajapakse. Special thanks to Karen Smith: I benefited from her mathematical prowess as her student early in my graduate experience, and she has consistently pushed and supported me since then. Also thanks to Anne Speigle and Teresa Stokes; without their help, this project would not have been completed.

Furthermore, thanks to my fellow graduate students and friends. Your discussions have consistently lightened my mood, and I am looking forward to seeing your future success. Notably, Umang Varma and I forged a similar path through academia, and I value his wisdom and friendship. Finally, a massive thank you to my parents and my brother. Their confidence in my abilities, even in my most difficult times, has propelled me to where I am today.

This research was supported in part through computational resources and services provided by Advanced Research Computing at the University of Michigan, Ann Arbor. Funding was provided in part by the University of Michigan Department of Mathematics, the Michigan Institute for Data Science, and the Chan Zuckerberg Initiative.

TABLE OF CONTENTS

LIST OF FIGURES

FIGURE

LIST OF TABLES

TABLE

ABSTRACT

It is common practice to obtain answers to complex questions by analyzing large amounts of data. Formal modeling and careful mathematical definitions are essential to extracting relevant answers from data, and establishing a mathematical framework requires deliberate interdisciplinary collaboration between the specialists who provide the questions and the mathematicians who translate them. This dissertation details the results of two of these interdisciplinary collaborations: one in single cell RNA sequencing, and the other in fairness.

High throughput microfluidic protocols in single cell RNA sequencing (scRNA-seq) collect integer valued mRNA counts from many individual cells in a single experiment; this enables high resolution studies of rare cell types and cell development pathways. ScRNA-seq data are sparse: often 90% of the collected reads are zeros. Specialized methods are required to obtain solutions to biological questions from these sparse, integer-valued data.

Determining genetic markers that can identify specific cell populations is one of the major objectives of the analysis of mRNA count data. We introduce RANKCORR, a fast method with robust mathematical underpinnings that performs multi-class marker selection. RANKCORR proceeds by ranking the mRNA count data before linearly separating the ranked data using a small number of genes. Ranking scRNA-seq count data provides a reasonable non-parametric method for analyzing these data; we further include an analysis of the statistical properties of this rank transformation.

We compare the performance of RANKCORR to a variety of other marker selection methods. These experiments show that RANKCORR is consistently one of the top-performing marker selection methods on scRNA-seq data, though other methods show similar overall performance. This suggests that the speed of the algorithm is the most important consideration for large data sets. RANKCORR is efficient and able to handle the largest data sets; as such, it is a useful tool for dealing with high throughput scRNA-seq data.

The second collaboration combines state of the art machine learning methods with formal definitions of fairness. Machine learning methods have a tendency to preserve or exacerbate biases that exist in data; consequently, the algorithms that influence our daily lives often display biases against certain protected groups. It is both objectionable and often illegal to allow daily decisions (e.g. mortgage approvals, job advertisements) to disadvantage protected groups; a growing body of literature in the field of algorithmic fairness aims to mitigate these issues. We contribute two

methods towards this goal.

We first introduce a preprocessing method designed to debias the training data. Specifically, the method attempts to remove any variation in the original data that comes from protected group status. This is accomplished by leveraging knowledge of groups that we expect to receive similar outcomes from a fair algorithm.

We further present a method for training a classifier (from potentially biased data) that is both accurate and fair using the gradient boosting framework. Gradient boosting is a powerful method for constructing predictive models that can be superior to neural networks on tabular data; the development of a fair gradient boosting method is thus desirable for the adoption of fair methods. Moreover, the method that we present is designed to construct predictors that are fair at an individual level - that is, two comparable individuals will be assigned similar results. This is different from most of the existing fair algorithms that ensure fairness at a statistical level.

CHAPTER 1

Introduction

As computer processing power increases exponentially, collecting and analyzing large amounts of data has become a viable strategy for gleaning answers to previously unanswerable questions from many disciplines. Obtaining answers from data is best accomplished by formulating a precise mathematical framework to describe the related concepts so that the questions may be stated in a explicit, rigorous manner. Developing such a framework necessitates a true interdisciplinary collaboration between mathematicians and the practitioners of other fields in order to translate and share different technical vocabularies. In this dissertation, we focus on two of these interdisciplinary collaborations, and we present methods tailored to two specific and distinct fields: single cell RNA sequencing and fairness in algorithms.

1.1 Ranking sparse single cell RNA sequencing data

Single cell RNA sequencing (scRNA-seq) refers to the collection of gene expression information at the level of individual cells. The technologies required to sequence the tiny amount of mRNA material that is present in a single cell were only established in recent years; the previous "bulk" RNA sequencing methods examine average gene expression levels across a large population of cells. A smoothie is a commonly invoked analogy: bulk sequencing corresponds to measuring properties of the fully blended smoothie, while scRNA-seq is more similar to analyzing the individual ingredients that make up the smoothie. Indeed, scRNA-seq has made it possible to characterize cellular diversity at unprecedented resolutions by determining detailed gene expression profiles of cell types and states ([MBS+15, ZTB+17]). Important applications include developing an understanding of tumor heterogeneity in cancers [ST19] and acquiring a precise picture of cell differentiation pathways from stem cells to mature cell populations (for example, [GMM+18]).

Currently, the mRNA data from more than one million cells can be collected in one experiment due to the development of high throughput microfluidic sequencing protocols [xG]. The incorporation of unique molecular identifier (UMI) technology additionally makes it possible to process these raw sequencing reads into integer valued read counts (instead of the "counts per

million fragments" types of rates that were used in bulk sequencing [MBS+15]). There are technical and mathematical questions still present at every step of the scRNA-seq data collection pipeline (including detecting instances where two cells are combined and sequenced as one [KST+17] and aligning transcript reads to known gene sequences in order to accurately convert the collected reads into mRNA molecule counts [ALC18]). In this document, we focus on analyzing fully processed integer mRNA count data in order to answer relevant biological questions.

These scRNA-seq counts exhibit high variance and are sparse (often, more than 90% of the reads are 0 [KSS14]) for both biological (e.g. transcriptional bursting) and technical (e.g. $3'$ bias in UMI based sequencing protocols) reasons. Those characteristics, in combination with the integer valued quality of the counts and the high dimensionality of the data (often, $20{,}000$ genes show nonzero expression levels in an experiment), are such that scRNA-seq data do not match many of the models that underlie common data analysis techniques and existing machine learning methods. For this reason, it is important to develop specialized methods that can deal with these sparse data.

1.1.1 Finding genetic markers

A biological question that has generated a significant amount of study in the scRNA-seq literature is the problem of finding *marker genes*. From a biological perspective, we loosely define marker genes as genes that can be used to identify a given group of cells and distinguish those cells from all other cells or from other specific groups of cells (they are "markers" for the given group of cells). Usually, these are genes that show higher (or lower) expression levels in the group of interest compared to the rest of the cell population; this provides simple ways to visualize the cell types and to experimentally test for the given cell types. In practice, certain genes are more desirable markers than others; for example, marker genes that encode surface proteins allow for the physical isolation of cell types via fluorescence activated cell sorting (FACS) procedures.

A multitude of tools for finding marker genes are present in the (sc)RNA-seq literature. These tools often inherently define marker genes to be genes that are *differentially expressed* between two cell populations. That is, in order to find the genes that are useful for separating two populations of cells, a statistical test is applied to each gene in the data set to determine if the distributions of gene expression are different between the two populations: the genes with the most significance are selected as marker genes. The commonly-used analysis tools `scanpy` [WAT18] and Seurat [BHS+18] implement differential expression methods as their default marker selection techniques; see also [SR18] for a survey of differential expression methods.

Marker selection has also received extensive study in the computer science literature, where it is known as *feature selection*. Given a data set, the goal of feature selection is to determine a (small) subset of the variables (genes) in that data set that are the most "relevant." In this case, the relevance of a set of variables is defined by some external evaluation function - different feature

selection algorithms use a variety of approaches to optimize different relevance functions.

There are generally two main classes of feature selection algorithms: greedy algorithms that select features one-by-one, computing a score at each step to determine the next marker to select (for example, forward- or backward-stepwise selection, see Section 3.3 of [HTF09]; mutual information based methods, see e.g. [BPZL12]; and other greedy methods e.g. [DK11]), and slower algorithms that are based on solving some regularized convex optimization problem (for example LASSO [Tib97], Elastic Nets [ZH05], and other related methods [LMRW18]).

At this point, we have implicitly defined three types of "marker genes":

- biological markers, i.e. genes whose expression can be used to distinguish the cells in one population from the other cells (or from other cell subpopulations);
- genes that are differentially expressed between one cell population and the other cells (or another cell subpopulation);
- and genes that are chosen according to a feature selection algorithm that statistically/mathematically characterizes the relevance of genes to the cell populations in some way (e.g. by minimizing a loss function).

Although we will use these ideas fairly interchangeably throughout this dissertation (referring, for example, to "the markers selected by the algorithm"), it is important to keep in mind the differences between them. For instance, a differentially expressed gene that shows low expression is not a particularly useful biological marker. Indeed, it would be difficult to use a low expression gene to visualize the differences between cell types and inefficient to purify cell populations based on a low expression gene with a FACS sorter.

Nevertheless, a major drawback of many existing feature selection and differential expression algorithms is that they are not designed to handle data that contain more than two cell types. Using a differential expression method, for example, one strategy is to pick a fixed number (e.g., 10) of the most statistically significant genes for each cell type; there may be overlap in the genes selected for different cell types. This strategy does not take into account the fact that some cell types are more difficult to characterize than others, however: one cell type may require more than 10 markers to separate from the other cells, while a different cell type may be separated with only one marker. Setting a significance threshold for the statistical test does not solve this problem; a cell type that is easy to separate from other cells will often exhibit several high significance markers, while a cell type that is difficult to separate might not exhibit any high significance markers.

In Chapters 2 and 3 of this dissertation, we introduce RANKCORR, a marker selection algorithm that addresses the problem of multi-class marker selection on massive data sets in a novel manner[1].

[1]Software available for download at `https://github.com/ahsv/RankCorr`.

RANKCORR is motivated by the algorithm introduced in [CGC+17]; here, we present a fast method for solving the optimization problem [PV13] at the core of the algorithm from [CGC+17]. As a result, RANKCORR runs quickly: it uses computational resources commensurate with several fast and light simple statistical techniques and it can run on data sets that contain over one million cells. In addition, a key step of the RANKCORR method is ranking the scRNA-seq data: this provides a non-parametric way of considering the counts and eliminates the need to normalize the data. We provide some analysis as to why ranking scRNA-seq data is a useful strategy for understanding these sparse data.

To find the markers for a fixed cluster, RANKCORR attempts to separate the (ranked) cells in the cluster from all other (ranked) cells using a hyperplane that passes through the origin. We require the normal vector to the hyperplane to depend on only a small number of features; these features are the markers that RANKCORR selects for the fixed cluster. To be clear: all hyperplanes separate the ranked counts, and do not easily correspond to surfaces in the original space of UMI counts

A full description of the RANKCORR algorithm is presented in Chapter 2. Then, the remainder of Chapter 2 is dedicated to a detailed evaluation of the empirical performance of RANKCORR. Specifically, we test the performance of RANKCORR when it is applied to a collection of four experimental UMI counts data sets and an ensemble of synthetic data sets. These diverse data sets include a variety of features that are interesting to researchers, such as small isolated clusters of cells and large high dimensional structures. Each data set contains an external clustering that is based on biological factors (e.g. expression of known markers) and algorithmic techniques - RANKCORR is used to select markers for these given clusterings. Using these data sets, we compare RANKCORR to a diverse set of feature selection methods. We consider other feature selection algorithms from the computer science literature, the statistical tests used by default in the Seurat [BHS+18] and `scanpy` [WAT18] packages, and several more complex statistical differential expression methods from the scRNA-seq literature. Refer to Chapter 2 for more details.

RANKCORR tends to perform well in comparison to the other methods, especially when selecting small numbers of markers. That said, there are generally only small differences between the different marker selection algorithms, and the "best" marker selection method depends on the data set being examined and the evaluation metric in question. It is thus impossible to conclude that any method *always* selects better markers than any of the others.

The major factors that differentiate the methods examined in Chapter 2 are the computational resources (both physical and temporal) that the methods require. This suggests that fast marker selection methods should be preferred over high complexity algorithms. RANKCORR is one of the fastest and lightest algorithms considered in this text, competitive with simple statistical tests. Thus, as a fast and efficient marker selection method that takes a further step towards understanding the multi-class case, RANKCORR is a useful tool to add into computational toolboxes.

This discussion is followed by a theoretical analysis of RANKCORR in Chapter 3. The analysis in Chapter 3 consists of three main parts. In the first part, we prove that the fast solution to the optimization problem from [PV13] presented in Chapter 2 is correct. As mentioned above, this fast algorithm is the core of the RANKCORR method. We then establish the time complexity of this fast algorithm and, under some weak assumptions about the sparsity of the scRNA-seq counts, show that RANKCORR will run in average time $\mathcal{O}(n^2)$ on a data set that contains counts from n cells and a fixed number p genes.

These results are followed by a further discussion of the effects of ranking (sparse integer count) data. Ranking the data is an integral part of the RANKCORR method, and we consider reasons why the ranked data may be more informative than the original count data. Moreover, we establish some of the statistical and geometric properties of the space of ranked points (which we call rank space).

Finally, we provide an analysis of the characteristics of the markers that are selected by the RANKCORR algorithm. We are able to prove that, in an ideal scenario, when selecting markers for a fixed group of cells, RANKCORR will prioritize markers that

- exhibit a constant level of expression in every cell in the given group; and
- exhibit a different constant level of expression in every cell outside of the given group of cells.

When sparsity considerations are taken into account, we are able to show that RANKCORR will select the least sparse gene with the above expression pattern (e.g. all zero counts are inside the fixed group of cells or all zero counts are outside the fixed group of cells). These results are proved by establishing a distance on rank space, and showing that RANKCORR functions to minimize the distance between one of the ranked genes and a specific point determined by the cell group. The ideal gene is then found by determining bounds on this distance. This result concludes our analysis of scRNA-seq data.

1.2 Fairness in algorithms

As a second topic in this dissertation, we explore the connections between machine learning and fairness in society. Algorithms have become ubiquitous and are used to determine outcomes for problems ranging from mortgage approval to risk assessment in prison sentencing. Due to a multitude of factors - for example, biases that are present in training data or a lack of training data from minority groups - naively trained algorithms tend to exhibit unfair (and often illegal) preferential treatment for some protected groups over other comparable groups. There are, in fact, numerous examples of algorithms with outcomes that are unfair to members of different protected classes.

One commonly cited example is that of the COMPAS recidivism prediction score. An individual's COMPAS score is determined from data collected when that individual is first arrested, and it aims to quantify the likelihood of recidivism. An analysis of these scores [ALMK16] found a significantly higher false positive rate in the scores for black individuals when compared to the scores for white individuals - specifically, black individuals who didn't recidivate within two years were more likely to be assigned high risk scores than white individuals who didn't recidivate within two years. Much discussion has been generated by this analysis, and there are also many other documented examples of discrimination in a machine learning method (e.g. [SA10, CBN17, Das18]).

In an attempt to address this issue, a large body of literature focused on fair machine learning was recently established. Starting with [DHP+12], this work has generally fallen into four categories:

1. Mathematical or statistical definitions of fairness (e.g. [FSV16, RSZ17, KMR16]).
2. Methods of preprocessing data in order to remove inherent bias so that algorithms trained on the debiased data will be fair (e.g. [ZWS+13, FFM+15]).
3. Algorithms that are modeled to ensure fairness (e.g. [JKMR16]).
4. Methods of debiasing the outcomes of existing algorithms (a postprocessing step; e.g. [HPS+16]).

In Chapter 4 of this dissertation, we present a data preprocessing method that falls into the second category. A machine learning method creates a predictor based on the information in the provided training data - thus, it is reasonable that a predictor that is trained on biased data will produce biased results. Moreover, the data that are available for training will usually capture the inherent biases present in society. For example, when training a model for allocating a police presence to different areas ("predictive policing"), it intuitively seems reasonable to measure the danger of an area by the number of arrests that occur in that area. Arrest rates are higher in black communities than in white communities, however, regardless of the actual crime rates in those communities [BGE13]. Thus, the arrest rate in a neighbourhood is an inherently biased quantity, and a classifier trained with these data will potentially capture this bias.

The main content of Chapter 4 is a method for debiasing training data so that naive machine learning methods can be used to train fair classifiers on the debiased data. The method presented in Chapter 4 considers any variation due to changes in protected attributes (for instance, race) as unwanted variation and adapts the RUV method [GbJS13] that is commonly used in genetic analysis in order to remove the unwanted variation from the data.

This debiasing procedure requires (potentially costly) expert input to determine groups of individuals that should be treated in the same way by the final algorithm. For this reason, debiasing in this way is not always an option: sometimes, it is necessary to train a classifier on biased data. In

Chapter 5 we present a method of constructing individually fair gradient boosted classifiers that create fair classifiers from potentially biased data (this falls in the third category of fairness work). Two terms deserve further exposition at this point: "individually fair" and "gradient boosting."

The myriad of fairness definitions in the literature generally fall into two categories: group fairness definitions and individual fairness definitions [CR18]. Group fairness definitions mostly prescribe that a statistical measure remains (nearly) invariant across given (fixed) groups of individuals. Methods constructed to satisfy group fairness constraints are often simple to analyze due to the statistical nature of these definitions; however, it is straightforward to construct methods that satisfy these group definitions but are obviously not fair to some individuals in the protected subpopulations. Examples of these constructions can be found in [DHP+12].

Individual fairness definitions, on the other hand, attempt to guarantee fairness for every individual in the population. This is usually formalized as the requirement that comparable individuals are assigned to similar outcomes. Inherent in this formalization of individual fairness is the nebulous notion of the *comparability* of individuals; this is something that is difficult to define from both sociological and mathematical perspectives. Partially for this reason, significantly less effort has been directed towards individually fair machine learning. Several recent works [Ilv19, WGL+19, YBS20] establish methods for obtaining a similarity metric between the individuals represented in a data set, however. Thus, in Chapter 5, we assume that we have access to a similarity metric for the data set and focus on the less-developed individual fairness framework.

On the other hand, boosting is a powerful framework for creating accurate predictors by iteratively adding simple classifiers (such as short trees) together; gradient boosting formulates this process as a functional gradient descent algorithm (the classifier that is added in the i-th step is essentially the descent direction of the loss functional). Gradient boosted decision tree (GBDT) methods in particular are known to produce classifiers that are comparable to or better than neural networks (NNs) when working with structured, tabular data (as opposed to unstructured data like pictures).

Previously known individual fairness methods are not able to train non-smooth ML models (e.g. [YBS20]) or are unable to adequately handle flexible non-parametric models (e.g. [YRWC19, CZBH19]); thus, unfortunately there are no existing methods for constructing individually fair decision trees. Therefore, in Chapter 5 we develop a method to enforce individual fairness in gradient boosting; unlike other individually fair training methods, our approach also works with non-smooth ML models such as GBDTs. The framework introduced in Chapter 5 is based on the field of distributionally robust optimization (DRO); since it is a method for gradient **b**oosting, we call this framework BuDRO.

BuDRO is inspired by [YBS20] and trains an individually fair classifier by optimizing a robust loss function L that is defined in Section 5.2.2. Essentially, this robust loss L considers perturbations

of the individuals represented in the input data (while retaining their originally assigned labels). The perturbations in the input data during training must be small according to the given fair metric. If an individual in the training set can be moved to a point that is nearby according to the fair metric, and these two points are classified in different ways by the proposed classifier, then a penalty term may be added to the loss function. To allow for boosting with non-smooth classifiers, BuDRO restricts the perturbations that are allowed during training so that individuals can only be moved to other individuals that are present in the training set (rather than to arbitrary places in the space of individuals).

In Section 5.2.2, we express the evaluation of L in terms of a linear program and use this fact to explicitly perform functional gradient descent using L. In Section 5.3, we further show that the robust loss L generalizes to a natural "true" population loss function as the size of the input data increases; this allows for L to provide a certificate of the individual fairness of a trained classifier. Section 5.4 considers implementation strategies that are aimed to speed up the evaluation of the linear program used to evaluate L.

Finally, in Section 5.5, we examine the performance of GBDT classifiers constructed according to the BuDRO framework on several tabular data sets that are commonly used as baselines in the fairness literature. We compare BuDRO to other methods for the creation of fair decision trees and other individually fair methods. Whenever possible, we evaluate the fairness of the learned classifiers using metrics that are designed to evaluate individual fairness, but we also report group fairness metrics to allow for easier comparisons with the existing literature. We find that the (unfair) baseline GBDT classifier does indeed outperform a one-layer fully connected NN on these data sets. Furthermore, the BuDRO framework trains individually fair classifiers that exploit the power of GBDT methods, resulting in high accuracy individually fair classifiers.

We further observe a trade-off between accuracy and group fairness metrics: allowing larger perturbations of the training individuals generally improves the group fairness of the learned classifier and decreases the accuracy of the predictions, while taking the allowed perturbations to zero recovers the baseline classifier. Thus, we observe the following two benefits of the BuDRO method on the tabular data sets examined in Chapter 5 (where baseline GBDTs outperforms NNs in terms of accuracy):

1. BuDRO can produce GBDT classifiers with higher accuracy than both the classifiers created by other methods for constructing fair GBDTs and the classifiers created by several methods for creating fair classifiers using NNs. These accurate BuDRO classifiers are individually fair, unlike the other fair GBDTs. In addition, these BuDRO classifiers have significantly improved group fairness properties compared to the baseline GBDT, but they do not always improve upon the group fairness metrics of the other fair classifiers.

2. BuDRO can produce GBDT classifiers with similar accuracy to the classifiers created by other methods for constructing fair GBDTs and the classifiers created by several methods for creating fair classifiers using NNs. These BuDRO classifiers are individually fair, unlike the other fair GBDTs. In addition, the group fairness properties of these BuDRO classifiers match or exceed the group fairness properties of the other classifiers.

These results provide evidence for the fact that BuDRO is a useful individually fair gradient boosting method: we manage to obtain individual fairness as well as utilize the power of GBDT methods on tabular data.

1.3 Account of contributions

The work in this dissertation was accomplished with collaborators. Chapter 2 is adapted from work done with Anna Gilbert, and was guided by many discussions with the members of the Michigan Center for Single-Cell Genomics, especially Jun Li. Chapter 3 is a write-up of related results that were guided by Anna Gilbert.

Chapter 4 is adapted from a joint work [BNSV18] with Yuekai Sun, Amanda Bower, and Laura Niss that was presented at FATML 2018. All authors contributed equally to the original publication, though Amanda Bower performed most of the experimental work and Laura Niss discovered the initial proofs of the results.

Finally, the work in Chapter 5 involves collaboration with Yuekai Sun, Mikhail Yurochkin, and Fan Zhang. I was the leading contributor to this project, and I wrote the original version of the Chapter (all collaborators have contributed tweaks, edits, and reworkings of sections). Yuekai Sun and I developed the original ideas for the method, and Yuekai has been consistently contributing towards the evolution of the method. Mikhail Yurochkin was originally included to contribute his valuable experimental expertise: he helped us to find data sets, he provided working implementations of other fairness algorithms from the literature, and he helped me to learn how to write code so that it will run on a GPU. After joining the project, he additionally contributed towards refining the method and finding ways to get it to run quickly and scalably. Fan Zhang contributed a proof of convergence of the individually fair gradient boosting method; this proof is not included in this document, but can be found in the published version. He also contributed towards writing the introduction of Chapter 5 and towards running experiments using the baseline GBDT methods.

1.4 General notation conventions

Here, we cover notation that we will take for granted over the entirety of this dissertation; each chapter will establish some notation specific to that section.

Let $\mathbb{R}$ denote the set of real numbers, $\mathbb{Z}$ denote the set of integers, and $\mathbb{N}$ denote the set of natural numbers. We use the notation $\|x\|_p$ to represent the p-norm of the vector x. For example, $\|x\|_2$ is the standard Euclidean norm of x and $\|x\|_1 = \sum_{i=1}^{n} |x_i|$. The notation $\|x\|_0$ represents the number of nonzero elements in x.

CHAPTER 2

A Rank-Based Marker Selection Method for High Throughput ScRNA-Seq Data

2.1 Background

This chapter contains a study of genetic marker selection on scRNA-seq counts data. Assuming that the cells in the scRNA-seq data are partitioned into groups ("cell types"), we would like to find the genes that are most informative about the grouping. These informative genes are called *marker genes*. Here, we introduce a new method for marker selection. This method, RANKCORR, is computationally efficient and can scale to the largest data sets that currently exist. Moreover, it performs as well as (or better than) the state-of-the-art marker selection methods from the literature.

Essentially, to select markers for a fixed group of cells, RANKCORR proceeds by ranking the scRNA-seq counts before finding the hyperplane passing though the origin that best (according to a specific loss function) separates the (ranked) cells in the group from the remainder of the (ranked) cells. It appears that ranking the counts helps to mitigate the ample noise present in these data - the relative values of the counts are often less noisy than the actual values of the counts. We explore this idea further in Chapter 3.

Multi-class feature selection is difficult, and there are no particularly efficient established solutions present in the literature. As discussed in Section 1.1.1, many marker selection methods in the scRNA-seq literature are based on differential expression and approach the multi-class setting by setting a significance threshold; any gene that exceeds the threshold for any cell type is labeled as a marker. Other methods simply take a fixed number of top genes from each group. RANKCORR handles the multi-class case in a one-vs-all fashion: it selects markers for each group of cells compared to all other cells. Instead of providing a score for every gene in each cluster and requiring for the user to manually trim these lists down, RANKCORR selects an informative number of markers for each cluster based on one input parameter. In the general case, different numbers of markers will be selected for different clusters. The union of the markers selected for all clusters provides a set of markers that is informative about the entire clustering. Unlike the method of choosing a significance threshold with a differential expression method, cell types that are more

difficult to separate from others will generally contribute more markers to the final set. It might be possible to establish a better multi-class method using group sparsity methods, but this is left as an open problem for future work. RANKCORR is a fast method that takes a step further than the commonly implemented standard.

In this chapter, we also provide an empirical evaluation of the RANKCORR algorithm. We consider selecting markers on a representative sample of different types of UMI counts data sets, both experimentally generated and synthetic. These data sets contain up to one million cells, and include examples of well-differentiated cell types as well as continuous differentiation trajectories. Moreover, each data set comes equipped with a cell type classification; we consider cell-type classifications that are biologically motivated as well as clusters that are algorithmically created. See Table 2.2 for a summary of the data sets; the full descriptions of the experimental data sets can be found in Section 2.5.1 and the synthetic data set construction process is described in Section 2.5.3. In these experiments, we compare RANKCORR to a large collection of other commonly considered marker selection methods; refer to Table 2.3 for a detailed list of these methods and Section 2.7 for further descriptions and implementation details.

There is currently no definitive ground truth set of markers for any experimental scRNA-seq data set. Known markers for cell types have usually been determined from bulk samples, and treating these as ground truth markers neglects the individual cell resolution of single cell sequencing. Moreover, we argue that the set of known markers is incomplete and that other genes could be used as effectively as (or more effectively than) known markers for many cell types. Indeed, finding new, better markers for rare cell types is one of the coveted promises of single cell sequencing.

Since one goal of marker selection is to discover heretofore unknown markers, we cannot easily evaluate the efficacy of a marker selection algorithm by testing to see if an algorithm recovers a set of previously known markers on experimental scRNA-seq data sets. For this reason we evaluate the quality of the selected markers by measuring how much information the selected markers provide about the given clustering. In this work, we propose several metrics that attempt to quantify this idea.

According to these evaluation metrics, all of the algorithms considered in this manuscript produce reasonable markers, in the sense that they all perform significantly better than choosing genes uniformly at random. In addition to this, RANKCORR tends to be one of the most optimal methods on many of the data sets, especially when only a small number of markers are selected. Overall, however, there are only slight differences in performance between the different marker selection algorithms, and the optimal method depends on the evaluation metric as well as the data set in consideration.

Despite this, there are large differences in the computational resources (both physical and temporal) required by the different methods. Since the algorithms show similar overall quality,

researchers should prefer marker selection methods that are fast and light. RANKCORR is one of the most efficient algorithms considered in this dissertation; in Chapter 3, we leverage the sparsity of the data to show that it runs in average time $\mathcal{O}(n^2)$ to select markers on n cells (as a quick comparison, this is the same as the time complexity required to read an $n \times n$ matrix). Empirically, we find that it exhibits computational requirements comparable to basic statistical techniques. This efficiency, combined with the fact that RANKCORR represents a step towards principled multi-class marker selection when compared to the procedures that are common in most existing methods, shows that RANKCORR is a worthy arrow to add to the quiver of commonly considered marker selection methods.

2.1.1 Related work: towards a precise definition of marker genes

In the Introduction, we discussed three definitions of "marker genes." We reproduce the list here for convenience:

- biological markers, i.e. genes whose expression can be used in a laboratory setting to distinguish the cells in one population from the other cells (or from other cell subpopulations);
- genes that are differentially expressed between one cell population and the other cells (or another cell subpopulation);
- and genes that are chosen according to a feature selection algorithm that statistically/mathematically characterizes the relevance of genes to the cell populations in some way (e.g. by minimizing a loss function).

Several recent marker selection tools start to bridge the gap between differentially expressed genes and biological markers. For example, [IK19] incorporates a high expression requirement in a heuristic mathematical definition of marker genes, and [DSC+19] utilizes a test for differential expression that is robust to small differences between population means. For a given cell type and candidate marker gene, the test used in [DSC+19] also incorporates both a lower bound on the number of cells that must express the candidate marker within the cell type and an upper bound on the number of cells that can express a marker outside of the cell type. See the discussion of marker selection methods in Section 2.7.1 for some further information.

In any case, it is worthwhile to establish a more precise biological definition of a marker gene in order to provide a solid theoretical framework for marker selection. For example, one biological definition of markers requires the cells to be grouped before markers can be determined; this assumes that markers are inherently associated with known cell types or states. This approach is influenced by the computational pipeline that many researchers are currently following (clustering followed by marker selection, e.g. [GWP+15, ZWT+17]) and is the approach we consider in this

manuscript. An alternative is to define markers as genes that naturally separate the cells into groups in some nice way; the discovered groups would then be classified into different cell types (i.e. allow marker selection to guide clustering). Another recent method [DVME19] defines markers in terms of their overall importance to a clustering, eschewing the notion of markers for specific cell types. Their framework also incorporates finding markers for hierarchical cell type classifications (instead of "flat" clustering). We leave full considerations of rigorous definitions for future work.

2.1.2 Notation and definitions

Consider an scRNA-seq experiment that collects gene expression information from n cells, and assume that p different mRNAs are detected during the experiment. After processing, for each cell that is sequenced, a vector $x \in \mathbb{R}^p$ is obtained: x_j represents the number of copies of a specific mRNA that was observed during the sequencing procedure. When all n cells are sequenced, this results in n vectors in $\mathbb{R}^p$, which we arrange into a data matrix $X \in \mathbb{R}^{n \times p}$. The entry $X_{i,j}$ represents the number of counts of gene j in cell i. Note that this is the *transpose* of the data matrix that is common in the scRNA-seq literature.

Let $[n] = \{1, \ldots, n\}$. For a matrix X, let X_i denote column i of X. Given a vector x, let $\mu(x) = \bar{x}$ denote the average of the elements of x and let $\sigma(x)$ represent the standard deviation of the elements in x; that is, $\sigma(x) = \sqrt{\frac{1}{n}\sum_{i=1}^{n}(x_i - \mu(x))^2}$.

2.2 Ranking scRNA-seq data

The first step of RANKCORR is to *rank* the entries of an scRNA-seq counts matrix X. In this section, we make the notion of ranking precise, and we establish some intuition as to why the rank transformation produces intelligible results on scRNA-seq UMI counts data. We can do much more formal analysis in regards to the behavior of the rank transformation on scRNA-seq data; some of this analysis appears in Chapter 3 of this dissertation.

Consider a vector $x \in \mathbb{R}^n$. For a given index i with $1 \leq i \leq n$, let $S_i(x) = \{\ell \in [n] \colon x_\ell < x_i\}$ and $E_i(x) = \{\ell \in [n] \colon x_\ell = x_i\}$ (note that $i \in E_i(x)$). We have that $|S_i(x)|$ is the number of elements of x that are strictly smaller than x_i and $|E_i(x)|$ is the number of elements of x that are equal to x_i (including x_i itself).

Definition 2.2.1. *The* rank transformation $\Phi \colon \mathbb{R}^n \to \mathbb{R}^n$ *is defined by*

$$\Phi(x)_i = |S_i(x)| + \frac{|E_i(x)| + 1}{2}. \tag{2.1}$$

Note that $\Phi(x)_i$ is the index of x_i in an ordered version of x (i.e. it is the *rank* of x_i in x). If multiple elements in x are equal, we assign their ranks to be the average of the ranks that would be

assigned to those elements (that is, for fixed i, all elements x_j for $j \in E_i(x)$ will be assigned the same rank).

Example. Let $n = 5$, and consider the point $x = (17, 17, 4, 308, 17)$. Then $\Phi(x) = (3, 3, 1, 5, 3)$. This value will be the same as the rank transformation applied to any point in $x \in \mathbb{R}^5$ with $x_3 < x_1 = x_2 = x_5 < x_4$.

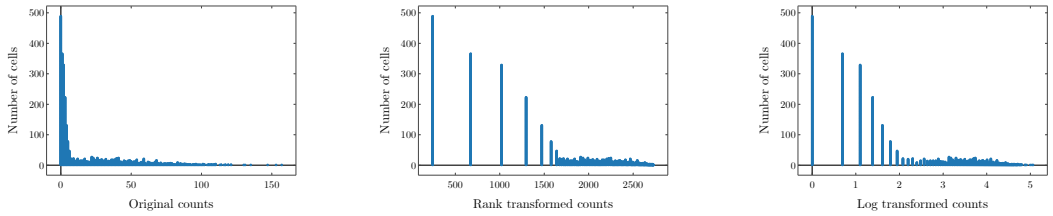
Ranking is commonly used in non-parametric statistical tests - ranking scRNA-seq data allows for statistical tests to be be performed on the data without any assumptions about the underlying distribution for the counts. This is important, since models for the counts distribution are continually evolving. As the measurement technology develops, different statistical models become more (or less) appropriate.

In addition, the rank transformation seems to be especially suited to UMI counts data, which is sparse and has a high dynamic range. When looking at the expression of a fixed gene g across a population of cells, it is intuitive to separate the cells in which g is expressed from the cells in which no expression of g is observed. Among the cells in which g is expressed, it is important to distinguish between low expression of g and high expression of g. The actual counts of g in cells with high expression (say a count of 500 vs a count of 1000) are often not especially important. Under the rank transformation, the largest count will be brought adjacent to the second-largest - no gap will be preserved. On the other hand, since there are many entries that are 0, the gap between no expression (a count of 0) and some expression will be significantly expanded (in the equation (2.1), the set $E_i(x)$ will be large for any i such that $x_i = 0$). See Figure 2.1 for a visualization of these ideas on experimental scRNA-seq data.

Stratifying the gene expression in this way intuitively seems useful for determining which genes are important in identifying cell types: a gene that shows expression in many of the cells of a given cell type can be used to separate that cell type from all of the others and thus is a useful marker gene. Thus, by enforcing a large separation between expression and no expression (when compared to the separation between low expression and high expression), it will be easier to identify markers. For these reasons, and since the rank transformation has shown promise in other scRNA-seq tools (for example NODES [SRL+16]) we use the rank transform in the RANKCORR marker selection algorithm.

A final note is that a connection can be made between the rank transformation and the log normalization that is commonly performed in the scRNA-seq literature. Often, the counts matrix X is normalized by taking $X_{ij} \mapsto \log(X_{ij} + 1)$. This is a nonlinear transformation that helps to reduce the gaps between the largest entries of X while leaving the entries that were originally 0 unchanged (and preserving much of the gap between "no expression" and "some expression"). With this in

mind, the rank transformation can be viewed as a more aggressive log transformation.



(a) Original PRTN3 data (b) Rank transformed PRTN3 data (c) Log transformed PRTN3 data

Figure 2.1: Counts of gene PRTN3 in bone marrow cells in the PAUL data set (See). Each point corresponds to a cell; the horizontal axis shows the number of reads and the vertical axis shows the number of cells with a fixed number of reads. No library size or cell size normalization has been carried out in these pictures. Note that the tail of the log transformed data is subjectively longer, while the gap between zero counts and nonzero counts appears larger in the rank transformed data

.

2.3 RANKCORR: A fast feature selection algorithm involving the rank transformation

RANKCORR,is based on the ideas presented in [CGC+17]. It is a fast algorithm that chooses an informative number of genes for each cluster by first ranking the scRNA-seq data, and then splitting the clusters in the ranked data with sparse separating hyperplanes that pass through the origin.

We start with an outline of how RANKCORR works in Section 2.3.1 so as to explain why it is such an efficient algorithm as compared to alternatives. A full description of the RANKCORR algorithm is then found in Section 2.3.2.

2.3.1 Intuitive description of RANKCORR

RANKCORR requires three inputs:

- An scRNA-seq counts matrix $X \in \mathbf{R}^{n \times p}$ (n cells, p genes). The rank transformation provides a non-parametric normalization of the counts data, and thus there is no need to normalize the counts data before starting marker selection. As discussed in Section 2.1, the rank transformation exploits the large number of ties present in scRNA-seq data to increase the separation between no expression of a gene and some expression of a gene (and thus extra normalization could be detrimental to the performance of the algorithm).

- A vector of labels $y \in \mathbf{Z}^n$ that defines a grouping of the cells ($y_i = k$ means that cell i belongs to group k). We think of y as separating the cells into distinct cell types or cell states, but in general the groups defined by y could consist of any arbitrary (non-overlapping) subsets of cells.

- A sparsity parameter s that indirectly controls the number of markers to select. For a fixed input X, increasing s will produce more markers.

Before selecting markers, RANKCORR starts by ranking and standardizing the input data X to create the matrix $\overline{X}$ defined by

$$\overline{X}_j = \frac{\Phi(X_j) - \mu(\Phi(X_j))}{\sigma(\Phi(X_j))}. \tag{2.2}$$

where Φ is the rank transformation, as defined in (2.1), and X_j denotes the j-th column of X[1]. Markers are then selected for the clustering defined by y in a one-vs-all manner; thus, we need to describe how RANKCORR selects markers for one group. For a single group $k \in y$, define $\tau \in \{\pm 1\}^n$ such that $\tau_i = +1$ if $y_i = k$ (that is, if cell i is in group k) and $\tau_i = -1$ otherwise; we refer to τ as the *cluster indicator vector* for the group k. To select markers for group k, RANKCORR constructs the vector $\overline{\tau} = \Phi(\tau) - \mu(\Phi(\tau))$. Following this, using the input parameter s, the algorithm determines the vector $\hat{\omega}$ as the solution to the optimization problem (2.3):

$$\begin{aligned} \hat{\omega} &= \arg\max_{\omega} \sum_{i=1}^{n} \overline{\tau}_i \langle \overline{x}_i, \omega \rangle \\ &\text{subject to } \|\omega\|_2 \leq 1, \|\omega\|_1 \leq \sqrt{s} \end{aligned} \tag{2.3}$$

where $\overline{x}_i$ denotes the i-th row of $\overline{X}$. RANKCORR returns the genes that have nonzero support in $\hat{\omega}$ as the markers for group k.

Note that the output $\hat{\omega}$ to (2.3) can be viewed as the normal vector to a hyperplane that passes through the origin and attempts to split the cells in group k from the cells that aren't in group k. For example, if cell i is in group k (i.e. $y_i = k$), then the term $\overline{\tau}_i \langle \overline{x}_i, \omega \rangle$ is positive exactly when $\langle \overline{x}_i, \omega \rangle > 0$. Thus, the objective function in (2.3) increases when more cells from group k are on the same side of the hyperplane with normal vector $\hat{\omega}$.

The optimization (2.3) was originally introduced in [PV13] in the context of sparse signal recovery and was adapted to the context of feature selection in a biological setting in [CGC+17]. The speed of RANKCORR is due to a fast algorithm (presented in Section 2.3.2.2) that allows us to quickly jump to the solution of the optimization (2.3) without the use of specialized optimization software.

[1]Recall that X_j represents the counts of gene j across the entire population of cells. Since the genes are ranked separately, a count of 0 will be given a different rank in each gene.

2.3.2 Details of the RANKCORR algorithm

Given a vector $x \in \mathbb{R}^p$ and a parameter $\beta \in \mathbb{R}$, we define the soft-thresholding operator $T_\beta(x)\colon \mathbb{R}^p \to \mathbb{R}^p$ by

$$T_\beta(x)_j = \begin{cases} \text{sign}(x_j)|x_j - \beta| & : \quad |x_j| > \beta \\ 0 & : \quad \text{otherwise} \end{cases} \tag{2.4}$$

We say that $T_\beta(x)$ is a soft-thresholding of the vector x.

2.3.2.1 Setup

Recall the notation from the previous section: let $X \in \mathbf{R}^{n\times p}$ be a scRNA-seq count matrix (n cells, p genes). Label the cells with the numbers in $[n]$. Given a subset $S \subset [n]$ of cells, define $\tau \in \{\pm 1\}^n$ such that $\tau_i = +1$ if cell i is in the subset (that is, if i is in S) and $\tau_i = -1$ otherwise. We refer to τ as the *cluster indicator vector* for the set S.

To find markers for S, we desire a vector $\omega \in \mathbb{R}^p$ such that

$$\tau = \text{sign}(\overline{X}\omega) \tag{2.5}$$

where $\overline{X}$ denotes a transformed version of X (we use the specific transformation (2.2) for RANKCORR). Note that, if cell i is in S, then $\langle \overline{x}_i, \omega\rangle > 0$; otherwise, $\langle \overline{x}_i, \omega\rangle < 0$. Thus, ω is the normal vector to a hyperplane passing through the origin that separates the cells that are in S from all other cells. In this framework, the nonzero entries of ω are marker genes for the subset S - they are the features that separate the given cell type from the other cells. To obtain a small number of markers, we desire a sparse solution ω; that is, a solution ω with few nonzero entries.

Unfortunately, it is computationally infeasible to find the sparsest vector ω^* that satisfies (2.5) (see [AK98]). In addition, for noisy experimental data, there is probably no vector ω that will perfectly satisfy (2.5).

In [PV13], the authors circumvent these issues by assuming that there is a vector ω (and a value t such that $\|\omega\|_0 \leq t$) that *mostly* satisfies (2.5) (i.e. the vector equality does not need to hold in all coordinates). They present the convex optimization (2.3) that uses $\overline{X}$, τ and an input sparsity parameter s to produce an approximate solution $\hat{\omega}$ that is "close" (in a technical sense) to this true sparse ω. Proposition 3.3.1 in Chapter 3 shows us that replacing $\overline{\tau}$ with τ does not affect the solution $\hat{\omega}$ of (2.3)[2]; thus, [PV13] actually presents the optimization printed below, also labelled (2.3) due to

[2]Using $\overline{\tau}$ results in an easier analysis of the algorithm, but τ is more convenient for the discussion in this section.

the similarity:

$$\begin{aligned} \hat{\omega} &= \arg\min_{\omega} \sum_{i=1}^{n} \tau_i \langle \overline{x}_i, \omega \rangle \\ &\text{subject to } \|\omega\|_2 \leq 1, \|\omega\|_1 \leq \sqrt{s}, \end{aligned} \tag{2.3}$$

where $\overline{x}_i$ denotes the i-th row of $\overline{X}$. We refer to s as a sparsity parameter due to the fact that it influences the number of zeros in the approximation $\hat{\omega}$. Specifically, s controls the size of the set that the approximation $\hat{\omega}$ will be chosen from: when $s \geq t$ (so that the true signal ω has $\|\omega\|_0 \leq s$), then ω will be in the feasible region of the optimization.

A couple of technical points deserve mention here: first, there are other efficient methods for obtaining an approximate solution $\hat{\omega}$ to (2.5). As mentioned in the overview of RANKCORR in Section 2.3.1, however, the optimization (2.3) has previously [CGC$^+$17] been developed into SPA, a feature selection algorithm for use with sparse biological data (specifically, mass spectrometry data in proteomics). We thus focus on the optimization (2.3) to solve (2.5) for this work.

Moreover, there are algorithms for finding general sparse separating hyperplanes (e.g. sparse support vector machines, see Section 4.5 of [HTF09]). These methods don't assume that the hyperplane passes through the origin, but they are somewhat slow, and it would not be reasonable to use them with the massive scRNA-seq data sets that are considered in this paper. Thus, to develop a fast marker selection method, we keep the additional assumption that the separating hyperplanes pass through the origin.

Finally, the fact that we are searching for a hyperplane passing through the origin necessitates a good choice of the transformation that yields $\overline{X}$ from X. For example, if X is left unchanged, then all of the cells lie in the first orthant of $\mathbb{R}^p$. In this case there is almost certainly no ω that satisfies (2.5); many hyperplanes, for example, do not even pass through the first orthant.

For the SPA method, the authors of [CGC$^+$17] construct the input $\overline{X}$ by "quasi-standardizing" the columns of the data matrix X: the column $\overline{X}_j$ is a linear combination of the centered version of X_j and the standardized version of X_j; that is,

$$\overline{X}_j = \alpha^{2(1-|\rho_j|)} \cdot \lambda \cdot (X_j - \mu(X_j)) + (1 - \alpha^{2(1-|\rho_j|)}) \cdot \left(\frac{X_j - \mu(X_j)}{\sigma(X_j)} \right) \tag{2.6}$$

where α and λ are hyperparameters and ρ_j is the empirical correlation of X_j with τ.

The goal of the quasi-standardization in SPA is to provide more weight to the features that are highly correlated with the labels τ (regardless of their expression levels) while also downweighting high expression genes that are not well correlated with the labels τ (that is, λ should be quite large and α should be less than 1; see Section 2.7 for more details about SPA). Similar behavior is accomplished in RANKCORR through the use of the rank transformation, see Section 2.2 for more information. Thus, RANKCORR has the benefit that there are no hyperparameters to tune.

In experiments, we see that RANKCORR runs much more quickly than the method SPA from [CGC+17] and generally produces better results on scRNA-seq data.

2.3.2.2 A fast algorithm for solving the optimization (2.3)

Given a matrix $\overline{X}$ and a signal ω (with $\|\omega\|_0 \leq s$), let $\tau = \text{sign}(\overline{X}\omega)$. Recall that the optimization (2.3) uses $\overline{X}$, τ, and s to provide an approximation $\hat{\omega}$ that is "close" to ω.

In both [Gen15] and [ZYJ14], the authors show that the solution $\hat{\omega}$ to (2.3) is given by a normalized soft thresholding of the vector

$$v = \sum_{i=1}^{n} \tau_i x_i, \tag{2.7}$$

where x_i represents the i-th row of $\overline{X}$. That is,

$$\hat{\omega} = T_\beta(v)/\|T_\beta(v)\|_2, \tag{2.8}$$

where β is a parameter that depends on s and the cluster indicator vector τ in a non-trivial manner. For feature selection, we are interested only in the support of $\hat{\omega}$. Thus, the optimization (2.3) can be solved simply by soft thresholding at each coordinate of v. Algorithm SELECT implements this idea to quickly find the support of $\hat{\omega}$. It is defined in Algorithm 1.

Intuitively, Algorithm 1 works in the following manner. For $s > 1$, note that the feasible set $\{x \in \mathbb{R}^p \colon \|x\|_1 \leq \sqrt{s}, \|x\|_2 \leq 1\}$ looks like the set $\{x \in \mathbb{R}^p \colon \|x\|_1 \leq \sqrt{s}\}$ with the corners chopped off and rounded. Thus, we start by creating $\tilde{v}$, a non-zero soft thresholding of v that has as few nonzero entries as possible[3]. We then soft threshold v by smaller and smaller values so that $\tilde{v}$ gains more non-zero coordinates and thus points further away from a coordinate axis. Since $\tilde{v}$ is always normalized, it is on the 2-sphere $\{x \colon \|x\| = 1\}$. Thus, we can stop when $\tilde{v}$ is also on the 1-sphere $\{x \colon \|x\|_1 = \sqrt{s}\}$. Then, $\tilde{v}$ is a point where the 1-sphere intersects the 2-sphere, and the support of this $\tilde{v}$ are the features that we are interested in selecting.

There is also a faster algorithm for solving a problem that is equivalent to (2.3) presented in [ZYJ14]. This algorithm does not allow for an interesting generalization to the multi-class problem, however.

We use SELECT in our implementations of both SPA and RANKCORR; this also means that our implementation of SPA is faster than it would have appeared in past work, including [CGC+17] (where SPA is introduced).

[3] In the usual case with noisy experimental data, v has a unique largest entry, and thus $\tilde{v}$ will have one nonzero entry so that it is pointing along one of the coordinate axes.

Algorithm 1 Solving the optimization (2.3)

procedure SELECT(A, an $n \times p$ matrix; $\tau \in \mathbb{R}^n$; s, the sparsity parameter.)
    Let $v \leftarrow \sum_{i=1}^{n} \tau_i a_i$ where a_i is the i-th row of A.
    Sort v by the magnitudes of its entries and let $\beta = v_1 = \|v\|_\infty$.
    Let k be the smallest index such that $|v_j| < \beta$.
    Let $j \leftarrow k$.
    Set $\beta \leftarrow |v_j|$.
    Let $\xi \leftarrow \|T_\beta(v)\|_1 / \|T_\beta(v)\|_2$.
    while $\xi \leq \sqrt{s}$ **do**
        $j \leftarrow j + 1$.
        $\beta \leftarrow v_j$
        $\xi \leftarrow \|T_\beta(v)\|_1 / \|T_\beta(v)\|_2$.
    end while
    if $j = k$ **then** ▷ Deal with corner cases, see Section 3.1.
        return the support of $T_{v_j}(v)$
    else if $j = n$ **then**
        return the support of v
    end if
    return the support of $T_{v_{j-1}}(v)$
end procedure

2.3.2.3 Applying SELECT to rank transformed data

This section contains an algorithm RANKBIN (defined in Algorithm 2) that uses SELECT along with the rank transformation to select markers for a fixed cell type in a way that is motivated by SPA. The inputs are a UMI counts matrix $X \in \mathbb{R}^{n \times p}$, a vector $\tau \in \{\pm 1\}^n$, and a sparsity parameter s. Note that this is still a binary marker selection method, since the entries of τ are either $+1$ or -1. The extension to the multi-class case is described in the next section.

Algorithm 2 Marker selection for one cluster using rank correlation

procedure RANKBIN(X, a potentially rank-transformed $n \times p$ matrix of counts, $\tau \in \{\pm 1\}^n$; s, the sparsity parameter)
    Construct the matrix $\overline{X}$ in the following manner: for all $1 \leq \ell \leq p$, let

$$\overline{X}_\ell \leftarrow \frac{\Phi(X_\ell) - \mu(\Phi(x_\ell))}{\sigma(\Phi(X_\ell))}$$

    .
    Let $\tau^c = \Phi(\tau) - \mu(\Phi(\tau))$
    return SELECT($\overline{X}, \tau^c, s$)
end procedure

In Algorithm 2, the construction of $\overline{X}$ is motivated by the quasi-standardization (2.6) of the data matrix X in SPA. In RANKBIN, the columns of the data matrix X are standardized *after* they are rank transformed. Moreover, motivated by the work in [CGC+17] and [Gen15], the vector τ is replaced with $\Phi(\tau) - \mu(\Phi(\tau))$ in RANKBIN. That is, the rank transformation is applied both to the data matrix X and the class indicator τ. In this case, the vector v that we soft threshold when we call SELECT (see (2.7), above) has entries given by

$$v_j = \sum_{i=1}^{n} (\Phi(\tau)_i - \mu(\Phi(\tau))) \frac{\Phi(X_j)_i - \mu(\Phi(X_j))}{\sigma(\Phi(X_j))}. \tag{2.9}$$

That is, entry j of v is (proportional to) the Spearman rank correlation between gene j and the vector τ. Thus, RANKBIN will select the genes that have the highest (absolute) Spearman correlation with the vector of class labels. It is possible to show that replacing τ with $\Phi(\tau) - \mu(\Phi(\tau))$ has no effect on the markers that are selected by the algorithm (Proposition 3.3.1); RANKBIN is written with $\overline{\Phi(\tau)}$ instead of τ to emphasize the connection with the Spearman rank correlation. (Similar calculations show that the markers selected by SPA are generally those that have high correlation with the cell type labels.)

Tangentially, note that the rows of the rank transformed and standardized data matrix $\overline{X}$ in Algorithm 2 come from a bounded - and thus sub-Gaussian - distribution with mean 0 and variance 1. Thus, this matrix $\overline{X}$ matches many of the hypotheses of the theoretical guarantees about the solution to (2.3) that are presented in [ALPV14] (the rows of $\overline{X}$ are not independent, however).

2.3.2.4 RANKCORR: Multi-class marker selection

RANKCORR, defined in Algorithm 3 works by fixing a parameter s and applying RANKBIN to each of the cell types in the data set. Specifically, fix a sparsity parameter s; this parameter will be the same for all of the cell types. For cell type j, construct the vector τ^j with $\tau_i^j = 1$ if cell i is in cell type j and $\tau_i^j = -1$ otherwise. Then run RANKBIN on the data matrix X, τ^j, and the fixed sparsity parameter s to get the markers for cell type j. This will usually result in a different (informative) number of markers selected for each cell type.

Note that the computation of $\overline{X}$ does not depend on the group j that we are selecting markers for; thus, $\overline{X}$ can be computed one time at the start of marker selection. To explicitly avoid this extra computation, we write RANKCORR without calling RANKBIN; the ideas behind the steps in the loop are provided in our discussion of RANKBIN, however.

In RANKCORR, we take the union of all the markers selected for each cluster to get a set of markers that will represent all of the given cell types. This step is to allow for easier collection of benchmarking statistics - we would like to capture how well a selected set of markers informs us

Algorithm 3 Multi-class marker selection

procedure RANKCORR(X, an $n \times p$ matrix of counts, $y \in \mathbb{N}^n$, a vector of cell type labels; s, the sparsity parameter)
    Construct the matrix $\overline{X}$ in the following manner: for all $1 \leq \ell \leq p$, let

$$\overline{X}_\ell \leftarrow \frac{\Phi(X_\ell) - \mu(\Phi(x_\ell))}{\sigma(\Phi(X_\ell))}$$

    .
    Let $m \leftarrow \min(y)$ and $M \leftarrow \max(y)$.
    Let $S = \emptyset$
    for $m \leq j \leq M$ **do**
        Construct τ^j in the following manner: $\tau^j_i = 1$ if $y_i = j$ and $\tau^j_i = -1$ otherwise.
        Let $\tau^c = \Phi(\tau^j) - \mu(\Phi(\tau^j))$
        Let $S_j = \text{SELECT}(\overline{X}, \tau^c, s)$
        $S \leftarrow S \cup S_j$
    end for
    return S
end procedure

about an entire clustering. In practice, the sets of markers could be kept separate to give information about individual cell types. Note that there could still be duplicate markers in these sets - here, we do not address the problem of dealing with duplicates in a smart way.

As a final note, when we are not interested in a full set of markers, we can quickly compare two genes to see which gene is favored by RANKCORR for a fixed cluster q: gene j will be chosen as a marker more favorably than gene k if the norm of the Spearman correlation between τ^q and X_j is larger than the norm of the Spearman correlation between τ^q and X_k.

The effect of fixing s across groups is complex. In the full description of RANKCORR found in Section 2.3.2.3, we show that, for a fixed cluster with cluster indicator vector τ, the markers that RANKCORR selects are the genes that have the highest (in magnitude) Spearman correlation with τ. That is, similar to a differential expression method, RANKCORR can be thought of as generating lists of gene scores (one for each group) that are used to select the proper markers - instead of p-values, the scores considered by RANKCORR are magnitudes of specific Spearman correlations.

The parameter s does not directly control the number of high-correlation markers that are selected, however, and fixing s results in different numbers of markers for each cluster. Thus, the set of markers selected by RANKCORR is different from the set obtained by choosing a constant number of top scoring genes for each cluster. Additionally, fixing s is not equivalent to picking a correlation threshold ρ and selecting all genes that exhibit a Spearman correlation greater than ρ with any cluster indicator vector. Determining a precise characterization of the numbers of markers selected for each cluster is left for future work.

The important point for this chapter is that RANKCORR contains a new method of merging lists of scores that is based on an algorithm with known performance guarantees [PV13]. Whether or not this one-vs-all method is an improvement over common heuristics for merging lists requires further exploration. There is some evidence, collected using simple synthetic test data, that selecting a constant number genes with the top Spearman correlation scores for each cluster results in comparable performance to RANKCORR. This has not been studied in the context of experimental data, however, and does not lead to any appreciable time savings over RANKCORR. It is also possible that the merging method used by RANKCORR could be adapted to work with p-values and provide an alternative method of merging the lists that are produced by differential expression methods. Thus we focus on RANKCORR as it is currently presented.

2.4 Evaluation of marker sets when ground truth markers are not known

To interpret, evaluate, and simply to present our results, we must quantify how much information a set of selected markers provides about a given clustering when ground truth markers are not known for certain (e.g., when selecting markers on an experimental data set). We propose two general procedures to accomplish this and present results using these:

- Supervised classification (Section 2.4.2): train a classifier on the data contained in the selected markers using the ground truth clustering as the target output.
- Unsupervised clustering (Section 2.4.3): cluster the cells using the information in the set of selected markers without reference to the ground truth clustering.

In this study, we implemented algorithms that accomplish each general procedure.

Assuming that the data set contains n points in k clusters, the result obtained by either classifying or clustering the data is a vector of predicted cell type labels $\hat{y} \in \mathbb{Z}^n$. We would like to compare this to the "ground truth" cluster label vector $y \in [k]^n$. The full information about the similarity between y and $\hat{y}$ can be presented in terms of a confusion matrix; this is unwieldy when many such comparisons are required, however. For this reason, many summary statistics have been developed in the machine learning literature for the classification [sld19b] and clustering [sld19a] settings. For each general procedure, we choose to examine several of these metrics; the full list is summarized in Table 2.1, along with the abbreviations that we will use to refer to the metrics.

The three supervised classification metrics (error rate, precision, and Matthews correlation coefficient) generally provide similar information. Thus, for most selected marker sets, we present results from five of the metrics (NCC classification error, RFC classification error, ARI, AMI, and FMS). The precision and Matthews correlation coefficient data can be found in Appendix A.

Procedure	Methods	Metrics
Supervised classification	Nearest centroid classifier (NCC) Random forests classifier (RFC)	Classification error (1 - accuracy) Average precision Matthews correlation coefficient
Unsupervised clustering	Louvain clustering	Adjusted rand index (ARI) Adjusted mutual information (AMI) Fowlkes-Mallows score (FMS)

Table 2.1: Evaluation metrics for marker sets on experimental data. The "Average precision" metric is a weighted average of precision over the clusters. The Matthews correlation coefficient is a summary statistic that incorporates all information from the confusion matrix. See [sld19b] for more information about the classification metrics and [sld19a] for more information about the clustering metrics.

It is important to note that these metrics represent some summary statistical information about the selected markers - they do not capture the full information contained in a set of genes. Results on synthetic data (Section 2.5.4) suggest that the metrics are informative but not fine-grained enough to capture all differences between methods. Therefore, we would advise considering these metrics as "tests" for marker selection methods; that is, these metrics should mostly be used to identify marker selection methods that don't perform well.

2.4.1 Cross validation

In order to avoid overfitting, we perform all marker selection, classification, and clustering using 5-fold cross validation. Cross validation is commonly used in the computer science literature, as it helps to avoid overfitting while allowing for all of the data to be considered in test sets (we never test directly on the data that we trained with) - see Section 7.10 of [HTF09]. See Figure 2.2 for a summary of this procedure.

Specifically, we split the cells into five groups (called "folds"). For each fold, we combine the other four folds into one data set, find the markers on the dataset containing four folds, and train the classifier using the selected markers on the dataset containing four folds as the training data. We then apply the trained classifier to the initial (held-out) fold and perform clustering on the initial fold using the markers that were selected on the other four folds. In this way, the initial fold is "test data" for the classifier/clustering metrics.

Repeating this process for all five folds creates a classification for the entire data set. On the other hand, we get a separate clustering for each fold, and these clustering solutions may be incompatible (they may contain different numbers of clusters, for example). See Section 2.4.3.2 for how we reconcile this.

Finally, using 5-fold cross validation means that we always select markers on $\frac{4}{5}$ of the cells

and then use those markers to classify (or cluster) the other $\frac{1}{5}$ of the cells. The timing information reported in the following sections represents the time needed to select markers on one fold (not on the entire data set).

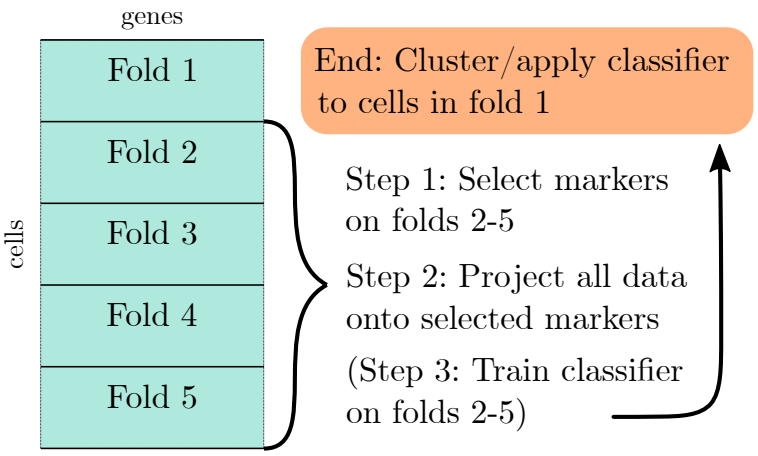


Figure 2.2: A visual description of 5 fold cross-validation

2.4.2 Evaluation metrics based on supervised classification

To incorporate information about the ground truth clustering into an evaluation metric, we train a multi-class classifier on the scRNA-seq data using the cluster labels as the target output. In order to evaluate the selected marker genes, we train the classifier using only the marker genes as the input data. When applied to a vector of counts (e.g. the counts of the markers in a cell), the classifier outputs a prediction of which cluster the vector belongs to.

2.4.2.1 Training a classifier

The specifics of how a classifier is trained are presented in Algorithm 4.

Algorithm 4 Classification training

1: **procedure** TRAINCLASSIFER($y \in \mathbb{N}^n$, a vector of cluster labels; $X \in \mathbb{R}^{n \times p}$, a scRNA-seq counts matrix; $S = \{s_1, \ldots, s_\ell\} \subset [p]$, a set of markers; CLASS, a classification algorithm)
2: Normalize X
3: Form a matrix $\Xi \in \mathbb{R}^{n \times |S|}$ from X by ignoring coordinates that aren't in S (i.e. $\Xi_i = X_{s_i}$).
4: Train the classifier CLASS with Ξ as the input vectors and y as the target labels.
5: Let $h \colon \mathbb{R}^{|S|} \to \mathbb{N}$ be the classification function output by CLASS.TRAIN(Ξ, y).
6: **return** h
7: **end procedure**

In line 2 of Algorithm 4, we normalize the matrix X. It is possible to use any normalization for this step; for the purposes of our analysis we use a log normalization procedure that is commonly

found in the scRNA-seq literature. Specifically, we perform a library-size normalization so that the sum of the entries in each row of X is $10,000$ and follow this by taking the base 2 logarithm of (1 plus) each entry of X to create a "log normalized" counts matrix.

Library size normalization was introduced in [GKvO14] to account for differences in capture efficiency between cells and taking a logarithm has it roots in bulk RNA-seq where it is used to attenuate technical variance [LHA14]. Since log normalization of this type is often applied when clustering scRNA-seq counts data in a data processing pipeline, we apply log normalization when attempting to recover the information in the given clusters. It is important to note that the marker selection algorithms that we examine in this work (Table 2.3) do not assume that the input counts data are normalized (apart from when noted in their descriptions; see Section 2.7).

2.4.2.2 Classification evaluation metrics.

We select markers and classify the cells using 5-fold cross-validation; see Figure 2.2. Once we have classified all cells in the data set, we examine how well the vector of classification labels matches the vector of ground truth cluster labels. Since we are in a classification framework, we use multi-class classification evaluation metrics for this purpose. In particular, we examine the classification error (1- accuracy) and precision of the classification compared to the known ground truth. For precision in a multi-class setting, we compute the precision for each class (as in a binary classification setting) and then take a weighted average of the per-class precision values, weighted by the class sizes. Finally, we also examine the Matthews correlation coefficient, which is a summary statistic that incorporates information about the entire confusion matrix. See [sld19b] for more information about these statistics.

In all of the tests that we perform in this work, the precision and Matthews correlation coefficient curves look subjectively similar to each other (though the actual values of the statistics do differ), while the classification error appears very similar to the other curves except it is flipped upside down. It is not clear why these summary statistics look as similar as they do. In any case, we generally only present the classification error rate in this document; the precision and Matthews correlation coefficient pictures can be found in Appendix A (Figures A.1 and A.2 for ZEISEL; Figures A.4 and A.5 for PAUL; Figures A.7-A.10 for ZHENGFILT and ZHENGFULL; and Figure A.12 for 10XMOUSE).

2.4.2.3 Classifiers.

We examine two classifiers to evaluate the marker sets (so that we are computing two classifications for each selected set of markers, and looking at all three metrics for both classifications).

The first is a simple (and fast) nearest centroids method that uses information about the original

clustering to determine the locations of the cluster centroids. We refer to this as the Nearest Centroids Classifier (NCC). See the end of this section for a full description of the NCC. In the second, we use the Random Forest Classifier (RFC) that is implemented in the python package `scikit-learn` ([PVG+11]), version 0.20.0, with `nestimators` = 100.

The summary statistics of the classifications produced when using the RFC are always better (more optimal) than the statistics that are produced when using the NCC. In addition, the overall shape of the curves produced using the RFC mostly mirror the curves produced using the NCC. The RFC is too slow to run on the largest data sets that we examine for testing. Since the RFC and NFC curves look similar for the smaller data sets, we are not concerned that we are missing information here.

Also note that, even with `nestimators` = 100, there is a significant amount of variability in the classification results obtained through the RFC. That is, running the RFC multiple times with the same set of markers will produce different classification results. See Appendix A, Figure A.13 for a visualization of the differences in error rate that can be obtained when running the RFC twice on the same sets of markers (this example is created using the PAUL data set; see the discussion of experimental data sets in Section 2.5.1).

2.4.2.4 The nearest centroids classifier (NCC).

The `NearestCentroids.train` method is presented in Algorithm 5.

Algorithm 5 Training a nearest centroids classifier

procedure NEARESTCENTROIDS.TRAIN($\Xi \in \mathbb{R}^{n \times \ell}$, an scRNA-seq counts matrix; $y \in \mathbb{N}^n$, a vector of cluster labels)
  Let $S(y) = \{i \in \mathbb{N} \colon i \in y\}$ be the unique entries of y.
  For each $k \in S(y)$, let $C_k = \{i \colon y_i = k\}$.
  For each $k \in S(y)$, let $c_k = \frac{1}{|C_k|} \sum_{i \in C_k} \xi_i$, where ξ_i is the i-th row of Ξ.
  Let $h \colon [n] \to \mathbb{N}$ be defined by $h(j) = \arg\min_k \|\xi_j - c_k\|_2$ (using Euclidean distance).
  return h, the classification function.
end procedure

2.4.3 Unsupervised clustering

Another natural way to measure the information in a selected set of markers is to cluster the data using only the selected coordinates in an unsupervised manner and compare this new clustering to the original clustering. Clustering scRNA-seq is itself a complicated problem that has inspired a great deal of study; here we restrict ourselves to Louvain clustering as implemented in the `scanpy` (version 1.3.7) package. Louvain clustering was introduced for use with scRNA-seq experiments in

[LSB+15] and it is currently the recommended method for clustering scRNA-seq data in several commonly-used software packages including `scanpy` [WAT18] and Seurat [BHS+18].

2.4.3.1 The clustering procedure.

A method for performing Louvain clustering on scRNA-seq data is presented in Algorithm 6. Note that we do not perform any dimensionality reduction (e.g. PCA) before finding nearest neighbors or performing the clustering. This is due to the fact that we project the data onto the selected markers. These markers are meant to capture the important dimensions in the data - they are the features that have the most information about the clustering according to a marker selection algorithm. Thus, we work in the space spanned by these markers without performing any additional dimensionality reduction.

Algorithm 6 Louvain clustering on scRNA-seq data

1: **procedure** CLUSTER($X \in \mathbb{R}^{n \times p}$, a scRNA-seq counts matrix; $S = \{s_1, \ldots, s_\ell\} \subset [p]$, a set of markers; r, a resolution parameter for Louvain clustering; k, the number of nearest neighbors to consider in Louvain clustering)
2: Normalize X
3: Form a matrix $\Xi \in \mathbb{R}^{n \times |S|}$ from X by ignoring coordinates that aren't in S (i.e. $\Xi_i = X_{s_i}$).
4: For each cell i, find the k nearest neighbors to i according to Euclidean distance between the rows of Ξ.
5: Run Louvain clustering with resolution r using the nearest neighbor data calculated in the previous step.
6: Let $h \colon [n] \to \mathbb{N}$ be a function that specifies the clustering. That is, $h(i) = j$ means that cell i was placed into cluster j.
7: **return** h.
8: **end procedure**

In line 2, of Algorithm 6, we normalize the counts matrix X. As in the case of the supervised classification metrics (Section 2.4.2), we apply log-normalization for this step.

2.4.3.2 Clustering evaluation metrics.

The unsupervised clustering is compared to the ground truth clustering using three metrics from the machine learning literature: the Adjusted Rand Index (ARI), Adjusted Mutual Information (AMI), and the Fowlkes-Mallows score (FMS). All three of these scores attempt to capture the amount of similarity between two groupings of one data set (e.g. the unsupervised clustering produced using a selected marker set and the ground truth clustering). They are also normalized scores: values near zero indicate that the cluster labels are close to random, while positive values indicate better performance. All of the scores have a maximum value of $+1$. Moreover, all three of

these metrics do not make any assumption about the number of clusters: the unsupervised clustering can have a different number of clusters from the ground truth clustering and these indices can still be computed. See [sld19a] for more information about these metrics.

We again use 5-fold cross-validation to compute the clustering performance metrics. Note that the clustering solutions for the different folds may be incompatible: for example, the number of clusters in the Louvain cluster solution for the first fold may be different from the number of clusters in the Louvain cluster solution for the second fold, and there may be no obvious way to relate the clusters in the first fold to the clusters in the second fold. For this reason, we compute the clustering performance metrics separately on each fold, comparing the Louvain cluster solution to the ground truth clustering restricted to the fold. The scores that we report are averaged over all of the folds (and when we optimize over the resolution parameter r, discussed below, we find the optimal value of the average over the folds).

Note that some of the fine structure from the ground truth clustering may not be maintained in a specific fold and thus it is impossible to capture this structure when performing Louvain clustering on the fold. This means that the actual values of these metrics are not particularly informative - it is more useful to compare the different methods along a metric. In addition, in all of the Louvain clusterings for a specific data set, we fix the the value of k, the number of nearest neighbours that we consider. Thus, small differences between the scores are not particularly informative, as they could disappear if k was selected perfectly for each method. Nonetheless, it is useful to get an idea as to how well the markers selected by different algorithms could be used in an unsupervised manner to recover a given clustering.

2.4.3.3 Choice of Louvain clustering parameters.

Louvain clustering requires the input of a number k of nearest neighbors and a resolution parameter r. It would be ideal to optimize both k and r for each set of markers on each data set for each clustering comparison metric; then we would be comparing the "optimal" performances of the marker selection algorithms under each metric. This is not computationally realistic for all of the data sets in consideration here, however.

Thus, for a given data set, we fix the value of k. For each set of markers on the data set, we compute the k nearest neighbors (once), and then quickly optimize over the resolution parameter r. To optimize r, we examine a grid from $r = 0.1$ to $r = 3.0$ with a step size of 0.1. This allows us to compute approximately optimal values of each of the metrics for each each set of selected markers in a computationally efficient manner. Importantly, the resolution parameter is optimized for each metric using each marker selection algorithm.

2.4.3.4 Choosing k.

See Section 2.5.1 for information on the data sets that we consider in this work. On the PAUL and ZEISEL data sets, we examined values of k varying from 15 to 30 (with a step size of 5). For each value of k, we used the RANKCORR algorithm to optimize over r (varying from $r = 0.1$ to $r = 3.0$ with a step size of 0.1), and we varied the number of markers selected to get a picture of the entire parameter space. The curves that are produced by this process are quite similar for different values of k (see Appendix A, Figures A.14 and A.15). The value of k is chosen to be the one that subjectively appears to optimize the performance of the majority of the metrics.

On the ZEISEL dataset, it appears that $k = 15$ nearest neighbors does not capture quite enough of the cluster structure, while $k = 30$ nearest neighbors results in lower scores than $k = 25$. We thus fix k at 25 for the unsupervised clustering evaluation on the ZEISEL data set. See Appendix A, Figure A.14 for the data that was used for this determination.

For the PAUL data set, we observed that changing the number of nearest neighbors used in the Louvain clustering has little effect on the ARI, AMI, or FM scores. It appeared that the scores were slightly improved for $k = 30$ when small numbers of markers were selected, thus we fixed k at 30 for the PAUL data set. See Appendix A, Figure A.15.

The ZHENGFULL and ZHENGFILT data sets are large, and thus we focus on the ZHENGFILT data set when considering the unsupervised clustering metric. To estimate a value of k, the fixed number of nearest neighbours that we use for all of the clusterings, we computed a Louvain clustering that looks quite similar to the bulk labels in a UMAP plot. This clustering used 25 nearest neighbors (and used the top 50 PCs); thus we fix k at 25 for the ZHENG data sets. See Appendix A, Figure A.16 to see a comparison of the bulk labels and the generated Louvain clustering in UMAP space.

2.5 Empirical performance of RANKCORR

Here, we present evidence that RANKCORR selects markers that are generally similar in quality to (or better than) the markers that are selected by other commonly used marker selection methods.[4] Moreover, RANKCORR runs quickly, and only requires computational resources comparable to those required to run simple statistical tests. Thus, RANKCORR is a useful marker selection tool for researchers to add to their computational libraries.

We evaluated the performance of RANKCORR on four experimental data sets and a collection of synthetic data sets. These data sets are listed in Table 2.2; the experimental data sets and synthetic data sets are further described in Sections 2.5.1 and 2.5.4.1 respectively. Each experimental data set in Table 2.2 is equipped with a "ground truth" clustering; we determine markers for the ground

[4]The Wilcoxon method is the default marker selection method in the Seurat [BHS+18] package, while the `scanpy` [WAT18] package defaults to the version of the t-test that we include here.

Data set	cells	non-zero genes	description	reference
ZEISEL	3005	4999	mouse neurons (well-separated clusters)	[ZMMC+15]
PAUL	2730	3451	mouse myeloid progenitor cells (differentiation trajectory)	[PAG+15]
ZHENGFULL	68579	20387	human PBMCs	[ZTB+17]
ZHENGFILT		5000		
10XMOUSE	1.3 million	24015	mouse neurons	[xG]
ZHENGSIM	5000	varies	two clusters simulated from human CD19+ B cells	simulated using [ZPO17]

Table 2.2: Data sets considered in this work. The 68k PBMC data set from [ZTB+17] appears twice in this table: ZHENGFILT contains a subset of the full data set ZHENGFULL. See Section 2.5.1 for more information. ZHENGSIM is a collection of simulated data sets created with the Splatter R package; see Section 2.5.3 for more information.

truth clusters and apply the evaluation metrics from Table 2.1 to evaluate the quality of the selected markers. The large number of cells in these data sets reflects the fact that modern scRNA-seq data sets tend to be larger: we are focused on high-throughput sequencing protocols in this work. We compare RANKCORR to the marker selection methods listed in the leftmost column of Table 2.3. See Section 2.7.1 for more marker selection method implementation details.

2.5.1 Experimental data sets

We examine four publicly available experimental scRNA-seq data sets in this work. We focus on data sets that have been clustered, with clusters that have been biologically verified in some way. In addition, we mostly examine data sets that were collected using microfluidic protocols (Drop-seq, 10X) with UMIs. This is due to the fact that these protocols tend to collect a smaller number of reads in a larger number of cells (producing large amounts of sparse data). These data sets are summarized in Table 2.2. We discuss them further below. See the statement on data availability in Section 2.7.4 for how to obtain these data and for more information about the scripts used for pre-processing.

ZEISEL. We work with one well-known reference fluidigm data set. This is ZEISEL, a data set consisting of mouse neuron cells that was introduced in [ZMMC+15]. Neuron cells are generally well-differentiated, and thus this data set contains distinct clusters that should be quite easy to

Method	Data sets	Package	Version	Ref
RANKCORR	All	custom		
SPA	ZEISEL, PAUL	custom		[CGC+17]
t-test	All	`scanpy`	1.3.7*	
Wilcoxon	All	`scanpy`	1.3.7*	
edgeR	ZEISEL, PAUL, ZHENGFILT	`edgeR`, `rpy2` v2.9.4	3.24.1	[RMS09]
MAST	ZEISEL, PAUL, ZHENGFILT	`MAST`, `rpy2` v2.9.4	1.8.1	[FMY+15]
scVI	ZEISEL, PAUL	Source from GitHub	0.2.4	[LRC+18]
Elastic Nets	ZEISEL, PAUL	`scikit-learn`	0.20.0	[ZH05]
Log. Reg.	All	`scanpy`	1.3.7*	[NYMP18]
Random selection	ZEISEL, PAUL, ZHENGFILT, ZHENGFULL			

Table 2.3: Differential expression methods tested in this paper. The "Data sets" column lists the data sets that each method is tested on in this work. The top block contains the methods that are presented in this work; implementations of these methods can be found in the repository linked in the data availability disclosure, Section 2.7.4. The second block of methods consists of general statistical tests. We use a slightly modified copy of `scanpy` version 1.3.7; see Section 2.7.1 for specifics. The third block consists of methods that were designed specifically for scRNA-seq data. The fourth block consists of standard machine learning methods; Log. Reg. stands for logistic regression. More information about the `scikit-learn` package can be found in [PVG+11]. We also consider selecting markers randomly without replacement. See Section 2.7.1 for more information about these methods.

separate. In [ZMMC+15], the authors have additionally used in-depth analysis with known markers in combination with a biclustering method of their own design to painstakingly label each cell with a specific cell type. This labeling is the closest to an actual ground truth clustering of a dataset in the scRNA-seq literature - this fact makes ZEISEL a valuable data set for our benchmarking purposes.

For our ground truth clustering, we consider only the nine major classes that the authors define in [ZMMC+15]. In addition, we pre-process the data set using the `cell_ranger` flavor of the `filter_genes_dispersion` function in the `scanpy` python package after library size normalization. We ask for the top 5000 most variable genes; the `filter_genes_dispersion` function only returns 4999 genes, however. We perform this pre-processing to speed up the marker selection process for the slower methods.

PAUL. The smallest data set that we examine is PAUL, a data set consisting of 2730 mouse bone marrow cells that was introduced in [PAG+15] and collected using the MARS-seq protocol. As opposed to ZEISEL, bone marrow cells consist of progenitor cells that are in the process of differentiating. Thus, there are no well separated cell types in the PAUL data - the data appear in a continuous trajectory. The authors of PAUL define discrete cell types along this trajectory based on known markers, however: we consider this clustering from [PAG+15] to be the ground truth for our analysis in this manuscript.

ZHENG data sets. We perform an analysis of the data set introduced in [ZTB+17] that consists of around 68 thousand human PBMCs from a single donor; we refer to this full data set as ZHENGFULL. These data were collected using 10x protocols.

The ground truth clustering for the ZHENG data set that we examine in this manuscript contains 11 clusters. It was constructed in [ZTB+17] by maximizing correlation with purified reference cell populations. This clustering contains some cell types that should be easy to separate (e.g. B cells vs T cells), as well as some cell types that have mostly overlapping profiles (e.g. several types of T cells are included as different clusters). The cluster labels can be found on the `scanpy_usage` GitHub repository at `https://github.com/theislab/scanpy_usage/blob/master/170503_zheng17/data/zheng17_bulk_lables.txt` (we use commit `54607f0`).

To be more precise, these clusters were determined in the following manner: first, the full data set was clustered (using k-means), and the clusters were assigned biological types based on known markers. The authors of [ZTB+17] then took more cells (from the same donor) and isolated a set cells of each cell type that they found in their clustering of ZHENGFULL. They then sequenced the cells from the individual types. Finally, they used these pure samples to cluster the ZHENGFULL data set again: each cell is assigned to the type whose (average) profile correlates most strongly

with the cell's profile.

We additionally generate a data set ZHENGFILT from ZHENGFULL by restricting to the top 5000 most variable genes. We select the 5000 most variable genes by performing a library size normalization on the ZHENGFULL data set and then using the `cell_ranger` flavor of the `filter_genes_dispersion` function in the `scanpy` python package. We can run the slower methods on ZHENGFILT.

1.3 million mouse neurons. Finally, we examine 10XMOUSE, a data set consisting of 1.3 million mouse neurons generated using 10x protocols [xG]. As noted above, neurons are well-differentiated into cell types, so this data set should contain well-separated clusters. The "ground truth" clustering that we consider for this data set is a graph-based (Louvain) clustering performed on the full 10XMOUSE dataset by the team behind `scanpy`. It can be found from the `scanpy_usage` GitHub repository (`https://github.com/theislab/scanpy_usage/tree/master/170522_visualizing_one_million_cells`; we consider commit `ba6eb85`) As far as we know, this clustering has not been verified in any biological manner.

2.5.2 Evaluating RANKCORR on experimental data

We have examined the relationship between the number of markers selected and the marker set performance metrics (see Section 2.4) on the different data sets. Selecting markers genes uniformly at random ("random selection") produces poor results, so those data are omitted from this section; the marker selection methods all perform better than random selection on the experimental data sets. See Appendix A (Figures A.1-A.11) for random marker selection data.

Performance summaries of the of the methods on the ZEISEL, PAUL, and ZHENGFILT data sets are presented in Figure 2.3. In this figure, the colors of the boxes indicate relative performance: a blue box indicates performance that is better than the majority of the other methods, a yellow box indicates standard performance, and an orange box indicates performance that is worse than the other methods. The coloring in the figures is based on Figures 2.5-2.11; the numbers of markers in the bins (in the top row) are selected to emphasize features found in these plots. The first row for each method represents the classification metrics and the second row represents the clustering metrics.

The values in the columns in Figure 2.3 correspond to a heuristic ranking of the methods, with 1 the optimal method (on average) in the indicated range of markers; see Section 2.7.3 for a full description of how the ranking is calculated. The classification metrics and clustering metrics are ranked separately (so that each column contains two full rankings of the methods; e.g. in Figure 2.3(b), every column contains the numbers 1 to 9 twice). Since these numbers are ranks, they do not

capture the magnitude of the gaps in performance between methods. For example, if two methods differ by one rank (e.g. the method ranked 2 vs the method ranked 3), there could be a large gap in performance between the two methods. The colors of the cells are meant to capture the larger differences between (tiers of) methods, and methods with the same color in a column perform comparably (regardless of the difference in rank). For example, in some cases, the top four methods (ranked 1 through 4) are similar to each other and clearly better than the others, so they will be colored blue. In other cases, no method will appear significantly better than the others, so none of the boxes will be colored blue.

The ZHENGFULL and 10XMOUSE data sets were too large for the majority of the methods to handle; thus, data were only collected for the RANKCORR, t-test, logistic regression, and Wilcoxon methods (the fastest methods) on these data sets. We include the 10XMOUSE data specifically as a stress test for the methods to see which could handle the largest data sets. It is impressive that these methods are able to run on such a large data set in a reasonable amount of time. The performance characteristics of the methods on these data sets are found later in this section.

Generally, the different methods select sets of markers that are of similar quality: the performance of any "optimal" method is usually not much better than several of its competitors. For example, the true differences in performance between the yellow and blue boxes in Figure 2.3 are often quite small. In addition, there is no method that consistently selects the best markers. The optimal method depends on the choice of data set, the evaluation metric, and the number of markers that are selected.

That said, the RANKCORR method tends to perform well on these data sets: it is generally competitive with the best methods in terms of performance. In particular, it especially excels when selecting less than 100 markers on all three data sets according to both the clustering an classification metrics.

Since the algorithms generally exhibit similar performances under the metrics considered in this work, efficient algorithms have a significant advantage. The computational resources (CPU time and memory) required to select markers on the experimental data sets are presented in Figure 2.4. The fastest and lightest methods are RANKCORR, the t-test, and Wilcoxon: notably, RANKCORR is nearly as fast and light as the two very simple statistical methods (the t-test and Wilcoxon). It is thus these three methods that show a clear advantage over the other methods for working with experimental data. Logistic regression also runs quickly on the smaller data sets, but does not scale as well as the three methods mentioned above, and significantly slows down on the larger data sets. In addition, logistic regression shows inconsistent performance, and is often one of the worst performers when selecting small numbers of markers. The other methods are significantly slower or require large computational resources compared to the size of the data set: see Figure 2.4 for further discussion.

Markers	1-30	30-60	60-100	100-150	150-200	200+
RankCorr	2	3	5	5	6	6
	3	1	1	2	7	5
Spa	N/A	6	7	6	5	5
	N/A	4	6	4	3	2
t-test	1	2	1	1	2	2
	5	9	9	3	6	1
Wilcoxon	4	4	3	3	3	1
	4	5	8	6	1	3
Log. Reg.	5	1	2	2	1	4
	8	8	4	7	5	8
E. nets	6	7	4	N/A	N/A	N/A
	1	2	3	N/A	N/A	N/A
edgeR	7	8	8	8	7	7
	6	7	7	8	8	6
MAST	3	5	6	4	4	3
	2	3	5	5	4	4
scVI	8	9	9	7	8	8
	7	6	2	1	2	7

(a) Zeisel

Markers	1-70	70-100	100-250	250-400	400+
RankCorr	5	5	8	5	3
	4	5	5	5	5
Spa	4	3	3	3	2
	5	7	6	6	7
t-test	2	2	7	7	7
	2	4	2	1	1
Wilcoxon	1	4	5	8	8
	1	1	1	2	2
Log. Reg.	3	1	2	4	6
	8	8	8	7	6
E. nets	8	7	1	1	4
	7	6	7	8	8
edgeR	7	8	4	2	1
	6	3	4	4	4
MAST	6	6	6	6	5
	3	2	3	3	3
scVI	9	9	9	9	9
	9	9	9	9	9

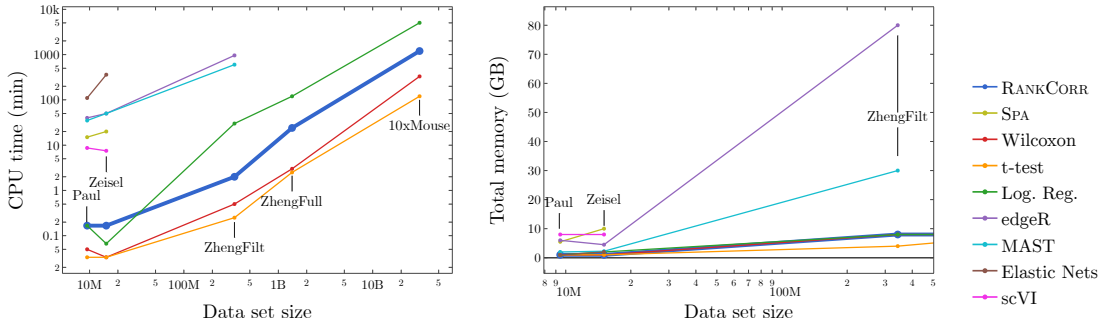
(b) Paul

Markers	1-20	20-50	50-120	120-200	200+
RankCorr	4	3	3	4	5
	3	5	4	5	6
t-test	2	1	2	2	4
	5	2	2	2	2
Wilcoxon	1	4	5	6	6
	2	1	1	1	1
Log. Reg.	6	6	6	5	2
	4	6	6	6	5
edgeR	5	5	4	1	1
	1	3	5	4	4
MAST	3	2	1	3	3
	6	4	3	3	3

(c) ZhengFilt

Figure 2.3: Performance of the marker selection methods on the (a) Zeisel, (b) Paul, and (c) ZhengFilt data sets as the number of selected markers is varied. There are two rows for each method; the first row for each method represents the classification metrics and the second row represents the clustering metrics. Blue indicates better performance than the other methods; orange indicates notably worse performance than the other methods. The marker bins are chosen to emphasize certain features in Figures 2.5-2.11; these figures present the values of the evaluation metrics for the different data sets. The values in the boxes correspond to a ranking of the methods, with 1 being the best method in the marker range. The classification and clustering results are ranked separately. Further notes: (a) All of the methods perform well on the Zeisel data set - an orange box here does not indicate poor performance, but rather that other methods outperformed the orange one. (b) Many of the methods showed nearly identical performance according to the classification metrics; thus, this table contains many yellow boxes.

These results support the idea that RANKCORR is a worthwhile marker selection method to consider (along side other fast methods) when analyzing massive UMI data sets. In the rest of this section, we give performance results on each specific data set.



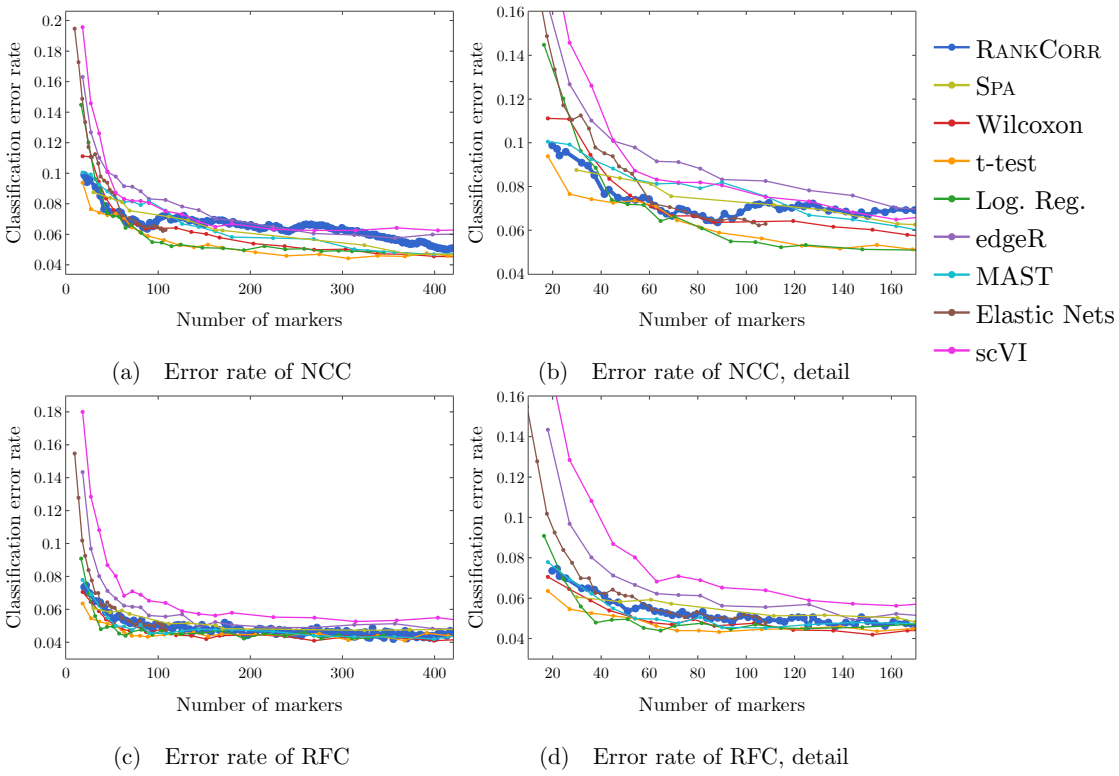
(a) Timing information (b) Required memory

Figure 2.4: Computational resources used by the marker selection methods. In both figures, the data set size is the number of entries in the data matrix X: it is given by $n \times p$, the number of cells times the number of genes. The sizes of the data sets that we consider in this work are indicated in the figures. The total CPU time required to select markers on one fold in the experimental data sets is shown in (a); the total memory required during these trials is shown in (b). Elastic nets scales poorly in (a), so it is only run on PAUL and ZEISEL. Both edgeR and MAST are limited by memory on ZHENGFILT (see (b)); this prevents their application to the larger data sets. scVI also requires a GPU while it is running; this prevents us from testing it on the larger data sets. RANKCORR, the t-test, Wilcoxon, and logistic regression all use 8 GB to run on ZHENGFULL and 80 GB to run on 10XMOUSE. See the Sectionsec:markMeth for more details.

2.5.2.1 The marker selection methods perform well on the ZEISEL data set

The classification error rates of the nearest centroid and random forests classifiers on the ZEISEL data set are presented in Figure 2.5. The error rates are very low: it requires only 100 markers (an average of 11 markers per cluster) to reach an error rate lower than 5% for most methods using the RFC. The ARI, AMI, and FM scores, reported in Figure 2.6, are also high (good) for all methods. Only a small number of markers were selected by elastic nets on the ZEISEL data set; thus, the elastic nets curves end before the others. Figure 2.3(a) contains a summary of the data presented in Figures 2.5 and 2.6. The computational resources required by the methods are presented in Figure 2.4, where it is clear that the RANKCORR, t-test, Wilcoxon, and logistic regression methods all run quickly on the ZEISEL data set and require few resources in comparison to the other methods.

The ground truth clustering that we consider on the ZEISEL data set is biologically motivated and contains nine clusters that are generally well separated (they represent distinct cell types). Most of the methods tested here produce markers that provide a significant amount of information about this ground truth clustering; these results thus represent a biological verification of the marker



(a) Error rate of NCC (b) Error rate of NCC, detail

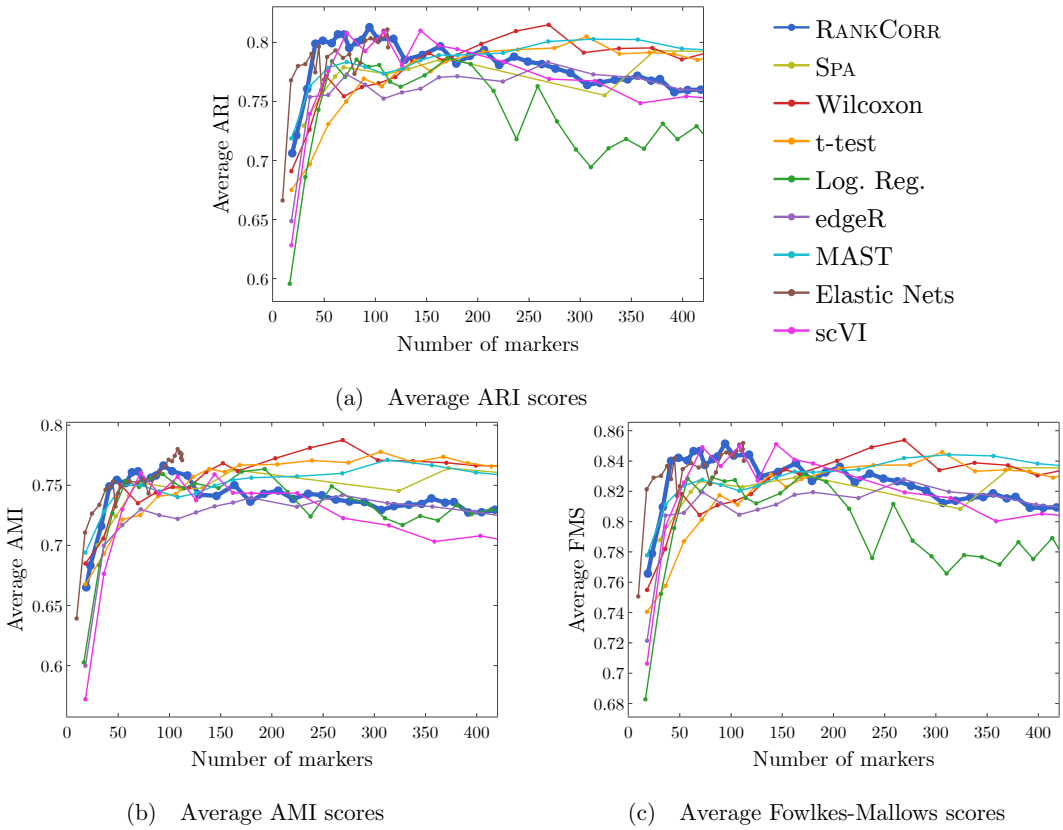
(c) Error rate of RFC (d) Error rate of RFC, detail

Figure 2.5: Error rate of both the nearest centroids classifier (NCC; (a) and (b)) and the random forests classifier (RFC; (c) and (d)) on the Zeisel data set. Figure (b) (respectively (d)) is a detailed image of the error rate of the different methods using the NCC (respectively RFC) when smaller numbers of markers are selected.

selection methods. That is, in this ideal biological scenario (a data set with highly discrete cell types) the (mathematically or statistically defined) markers that are chosen by the methods are biologically informative and can be used as real (biological) markers.

This suggests that, when selecting markers on a data set that is well clustered, it is useful to examine several marker selection algorithms to get different perspectives on which genes are most important. Marker selection algorithms that can run using only small amounts of resources, such as RankCorr (see Figure 2.4), thus have an advantage over the other methods.

In addition to this, RankCorr is the only method that shows high performance when selecting less than 100 markers in both the clustering and classification metrics. Most researchers will be looking for small numbers of markers for their data sets; thus RankCorr stands out as a promising method on the Zeisel data set. Note also that RankCorr generally outperforms Spa in the clustering metrics and is competitive with Spa in the classification metrics: RankCorr is both faster than Spa and selects a generally more informative set of markers than Spa on the Zeisel data set. Therefore, the performance on the Zeisel data set is evidence for the fact that RankCorr is a useful adaptation of Spa [CGC+17] for sparse UMI counts scRNA-seq data.



(a) Average ARI scores

(b) Average AMI scores

(c) Average Fowlkes-Mallows scores

Figure 2.6: Clustering performance metrics vs total number of markers selected for marker selection methods on the Zeisel data set. The ARI score is shown in (a), the AMI score is shown in (b), and the Fowlkes-Mallows score is shown in (c). The clustering is carried out using 5-fold cross validation and scores are averaged across folds.

Finally, these data illustrate how the different evaluation metrics provide different statistical snapshots into the information contained in a set of markers. For example, logistic regression performs significantly worse than all of the other methods when large numbers of markers are selected according to the ARI and FMS plots. In the supervised classification trials, however, the logistic regression method performs competitively with the other methods. When no information about the ground truth clustering is provided, the performance of logistic regression on the Zeisel data sets drops considerably. Despite the good results in the supervised clustering plots (which could be due to quickly selecting a small number of useful markers) it is reasonable to conclude that logistic regression selects many uninformative genes (in comparison to the other methods) as more markers are selected. It is best to think of the metrics as tests that can identify the marker selection methods that don't perform well.

2.5.2.2 Marker selection algorithms struggle with the cell types defined along the cell differentiation trajectory in the PAUL data set

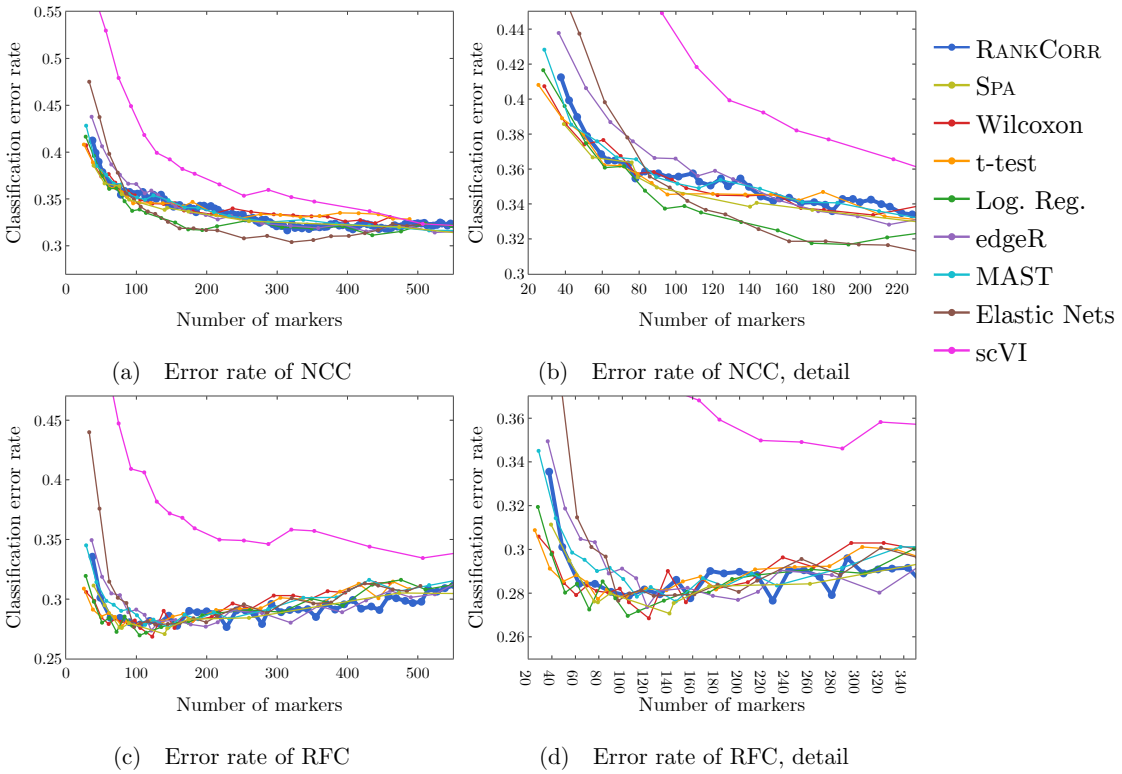
The PAUL data set consists of bone marrow cells and contains 19 clusters (some of which are very small). The clusters lie along a cell differentiation trajectory; therefore, it is reasonable that it would be difficult to separate the clusters or to accurately reproduce the clustering into discrete cell types. The Paul data set thus represents an adversarial example for these marker selection algorithms.

Figure 2.7 shows the performance of marker selection algorithms on the PAUL data set as evaluated by the supervised classification metrics. It is not surprising to see relatively high clustering error rates: the rates are always larger than 30% for the NCC and reach a minimum of around 27% with the RFC. Since there are 19 clusters, this is still much better then classifying the cells at random. The dependence of the ARI, AMI, and FM clustering scores on the number of markers selected is plotted for the different marker selection algorithms in Figure 2.8. The values of the scores are all in low to medium ranges for all marker selection algorithms.

All of the scores produced by all of the methods on the PAUL data set are significantly worse than the metrics on the ZEISEL data set; however, the methods perform considerably better than markers selected uniformly at random. (see Appendix A, Figures A.4-A.6 for this comparison). This is sensible, since it should intuitively be difficult to reproduce a discrete clustering that has been assigned along a continuous path. The notion of discrete cell types does not fit well with a cell differentiation trajectory; the poor score levels reflect the necessity to come up with a better mathematical description of a trajectory for the purposes of marker selection.

The ARI values are especially low on the PAUL data set, and the methods consistently produce lower ARI values than AMI values. This is a change from the ZEISEL data set, where the ARI scores were higher than the AMI scores (and the FMSs were the highest of all). The aspects of the data sets that change the relative ordering of the metrics are unclear; it must be the data sets that influence this change, however, since the change persists across the marker selection algorithms. Designing a metric for benchmarking marker selection algorithms is itself a difficult task, and the optimal metric to consider could depend on the data set in question.

A summary of the relative performance of the marker selection algorithms on the Paul data set is presented in Figure 2.3(b). All of the marker selection methods perform quite similarly on the PAUL data set. The RANKCORR algorithm is one of only three methods that always performs nearly optimally under every metric examined here; the others are the t-test and MAST. In addition, RANKCORR always performs well when selecting small numbers of markers, and shows exceptional performance in this regime under the Fowlkes-Mallows clustering metric. Combined with the facts that RANKCORR is fast to run and requires low computational resources, this shows that RANKCORR is a useful marker selection method to add to computational pipelines.



(a) Error rate of NCC

(b) Error rate of NCC, detail

(c) Error rate of RFC

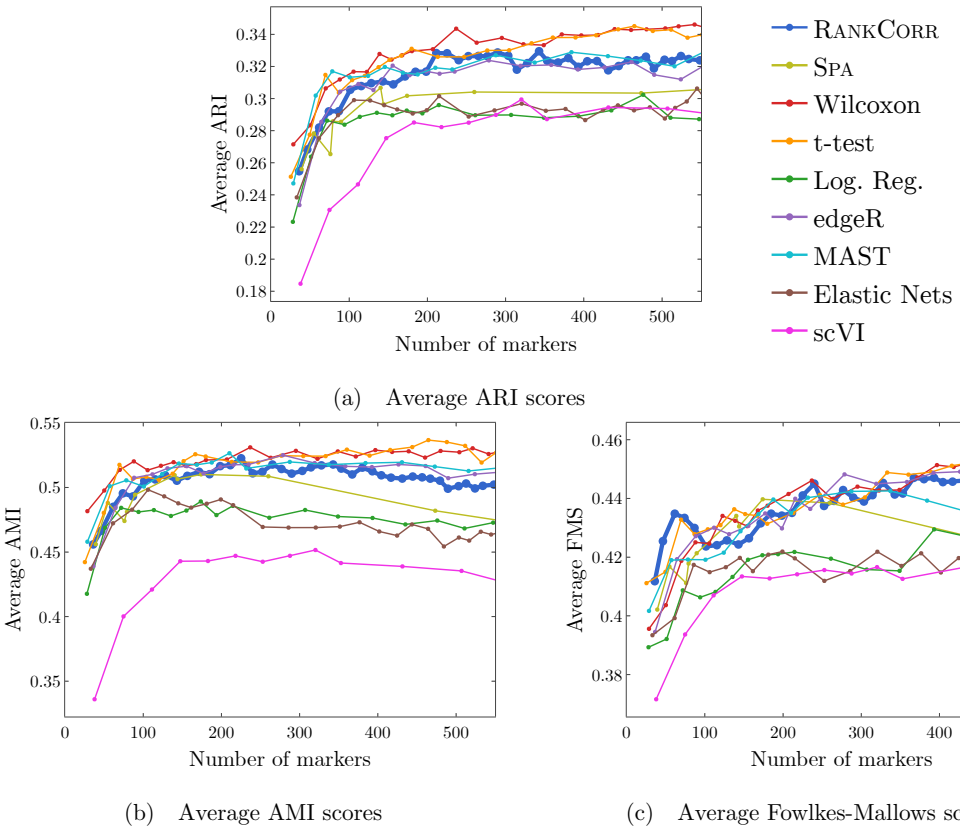
(d) Error rate of RFC, detail

Figure 2.7: Error rates of both the nearest centroids classifier (NCC; (a) and (b)) and the random forests classifier (RFC; (c) and (d)) on the Paul data set. Figure (b) (respectively (d)) is a detailed image of the error rate of the different methods using the NCC (respectively RFC) when smaller numbers of markers are selected. Figure (b) details up to 220 total markers to make clear how similar the methods perform for when small numbers of markers are selected. Figure (d) examines up to 350 total markers to detail the performance of the methods when small numbers of markers are selected as well as get an idea for the increasing behavior and noisy nature of the curves.

2.5.2.3 Results on the ZhengFull and ZhengFilt data sets

Here, we examine the 68k PBMC data set from [ZTB+17]; it contains data from more than 30 times the number of cells in either the Paul or Zeisel data sets. This is more representative of the sizes of the data sets that we are interested in working with. The ground truth clustering that we consider is the labeling obtained in [ZTB+17] by correlation with bulk profiles (biologically motivated "bulk labels"). There are 11 cell types in this clustering. See Section 2.5.1 for more information.

The Zheng data sets contain some distinct clusters (e.g. B cells), as well as some clusters that are highly overlapping (e.g. different types of T cells). There are not any specific cell differentiation trajectories (that we are aware of), but the overlapping clusters provide a challenge for the marker selection methods. Thus, we expect to see performance benchmarks between those of Paul and Zeisel.



(a) Average ARI scores

(b) Average AMI scores

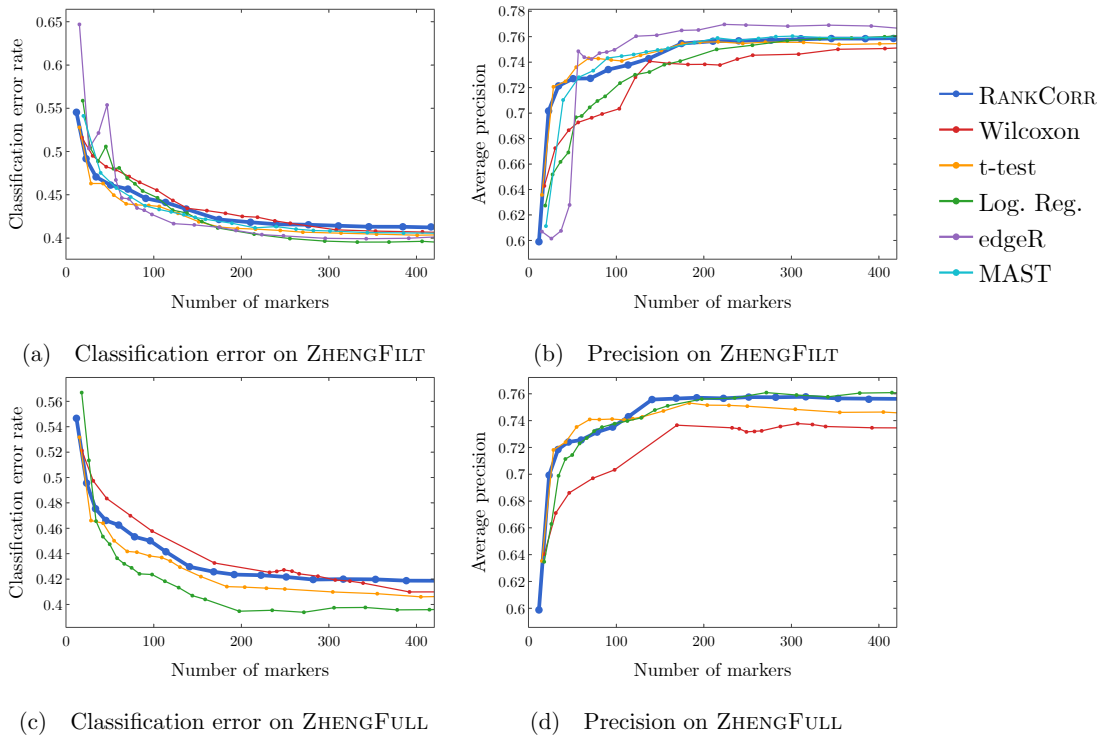
(c) Average Fowlkes-Mallows scores

Figure 2.8: Clustering performance metrics vs total number of markers selected for marker selection methods on the PAUL data set. The ARI score is shown in (a), the AMI score is shown in (b), and the Fowlkes-Mallows score is shown in (c). The clustering is carried out using 5-fold cross validation and scores are averaged across folds.

We mostly focus on ZHENGFILT, a version of the data set that is filtered to only include the information from the top 5000 most variable genes. We also consider the performance of the most efficient algorithms (RANKCORR, logistic regression, Wilcoxon, and the t-test) on ZHENGFULL, the data set containing all of the genes, to check for any differences. Extrapolating from Figure 2.4, it would be infeasible to run any of the other methods on ZHENGFULL. Here we also begin to see that logistic regression scales worse than the other methods: it is already becoming slow and computationally heavy on "only" 68 thousand cells.

Figure 2.9 focuses on the performance of the methods when the NCC is used for classification; corresponding data using the RFC is found in Figure 2.10. Unlike the PAUL and ZEISEL data sets, the precision curves are slightly different in some occasions, and thus they are presented here. In particular, the precision of these methods is significantly higher than their accuracy. Neither the classification accuracy nor the precision changes by very much when we filter from the full gene set (Figures 2.9(c,d) and 2.10(c,d)) to the 5000 most variable genes (Figures 2.9(a,b) and 2.10(a,b)). In general, this filtering very slightly increases both the accuracy and precision of the t-test, Wilcoxon,

and RANKCORR methods, while it worsens the performance of the Logistic Regression method. This suggests that enough marker genes are kept by this variable gene filtering process to maintain accurate marker selection.



(a) Classification error on ZHENGFILT (b) Precision on ZHENGFILT

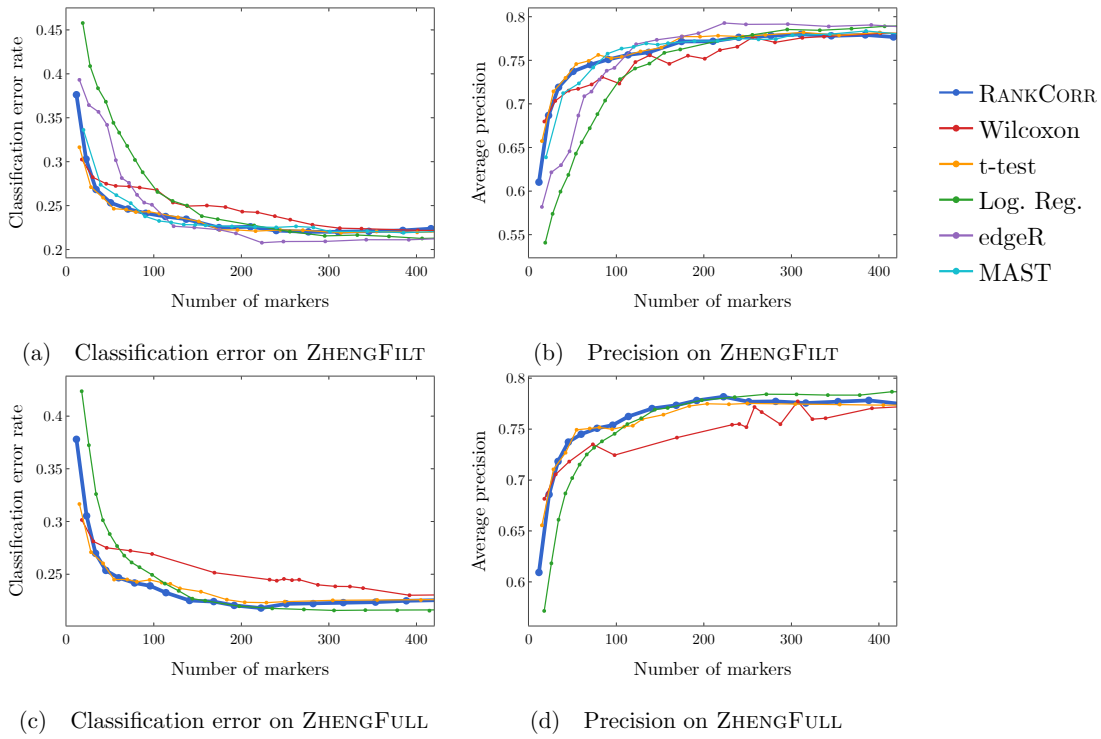
(c) Classification error on ZHENGFULL (d) Precision on ZHENGFULL

Figure 2.9: Accuracy and precision of the nearest centroids classifier on the ZHENG data sets using the bulk labels. The top row correspond to the ZHENGFILT data set and the bottom row corresponds to the ZHENGFULL data set.

Overall, the classification error rates according to the NCC for these data are quite high, and don't level off (to a minimum value of approximately 40%) until around 200 markers are selected; this corresponds to an average of around 18 unique markers per cluster. For very small numbers of markers selected, the classification error rates obtained from the NCC are quite high (around 55%).

The error rates using the RFC are decreased significantly compared to the NCC, and level off to approximately 22% when large numbers of markers are selected. The error again does not completely level off until around 200 total markers are selected, but there is a steeper initial descent. This steep initial descent in error rates could appear as the large groups of cells are separated from each other (e.g. B cells from T cells) and the slower improvement from 100 to 200 of total markers selected could be the methods fine-tuning the more difficult clusters (e.g. Regulatory T from Helper T). The error rates are between those observed in PAUL and ZEISEL. On the other hand, the error rates for the NCC classifier are much higher than expected.

We focus on the ZHENGFILT data set for the clustering metrics. This is due to the fact that the classification metrics are changed only slightly between ZHENGFILT and ZHENGFULL as well as



(a) Classification error on ZhengFilt (b) Precision on ZhengFilt

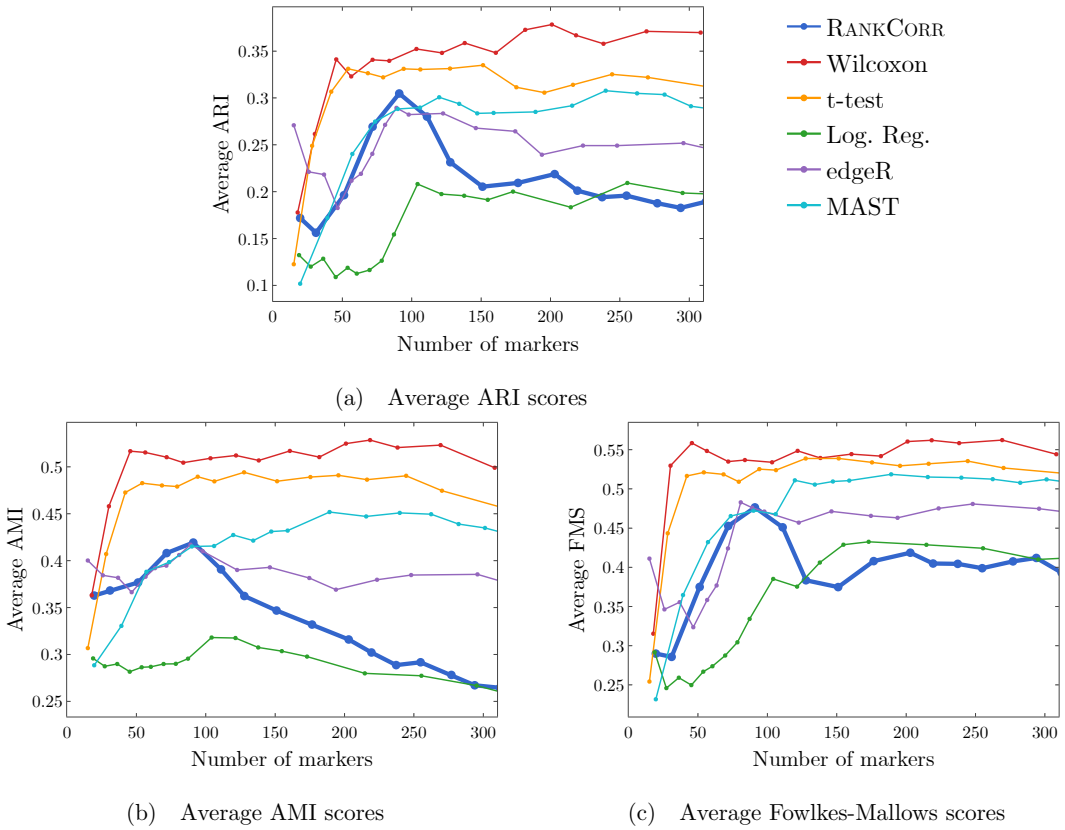
(c) Classification error on ZhengFull (d) Precision on ZhengFull

Figure 2.10: Accuracy and precision of the random forests classifier on the Zheng data sets using the bulk labels. The top row correspond to the ZhengFilt data set and the bottom row corresponds to the ZhengFull data set.

the fact that Louvain clustering on the large Zheng data set is itself time and resource intensive. The clustering metrics on the ZhengFilt data set are presented in Figure 2.11. All three scores are generally quite low, though they are again mostly much higher than random marker selection. The performance of random marker selection can be found in Appendix A, Figure A.11.

A summary of the performance of the marker selection algorithms on the ZhengFilt data set is presented in Figure 2.3(c). Apart from the t-test, the methods show inconsistent performance when comparing the clustering metrics to the classification metrics. For example, the edgeR method exhibits the top performance on the ZhengFilt data set after more than 50-100 unique markers are selected according to the classification metrics. The classification metrics show edgeR as one of the worst methods when choosing less than 50 unique markers, however. This is in direct contradiction to the clustering metrics, where edgeR is the best method for the smallest ($\sim$ 20) total numbers of markers selected, and it then shows performance in the middle of the other methods as larger numbers of markers are selected.

It is possible that changing the number of nearest neighbours considered in the Louvain clustering would produce more consistent data. Although the clustering metrics did not appear to change significantly when altering the number of nearest neighbours on the previous data sets (see the Louvain parameter selection information in Section 2.4.3.3), the ZhengFilt data set is much larger



(a) Average ARI scores

(b) Average AMI scores

(c) Average Fowlkes-Mallows scores

Figure 2.11: Clustering metrics on the ZhengFilt data set.

than those previous data sets; the larger number of cells may necessitate the use of information from more nearest neighbors to recreate the full clustering structure.

It is also possible that the bulk labels that are used for the ground truth are difficult to reproduce through the Louvain algorithm. We generated a clustering that visually looked like the bulk labels via the Louvain algorithm; (it appears in Appendix A, Figure A.16); the ARI, AMI, and FMS values for the generated Louvain clustering compared to the bulk labels are in the ranges produced by the Wilcoxon and t-test methods (not larger than the scores here). In addition, the top ARI and AMI scores (produced by the Wilcoxon and the t-test methods) are comparable to (or only slightly better than) the scores on the Paul data set (Figure 2.8). This runs counter to our expectations: the Paul data set contains a cell differentiation trajectory, with no real clusters that are easy to separate out, while the Zheng data sets contain several clusters that are well separated. It is possible that the bulk labels produce clusters that are more mixed than it appears in a UMAP plot.

In any case, the disparity between the different types of scores further emphasizes the fact that the classification and clustering metrics provide different ways of looking at the information contained in a selected set of markers. Methods that perform well according to both types of metrics should be preferred.

Following this logic, the t-test produces the overall best results on the ZHENGFILT data set. It performs well under the classification metrics, especially for small numbers of total markers selected. In addition, it is consistently competitive with the best method (Wilcoxon) according to the clustering metrics.

Nonetheless, on the whole, RANKCORR performs approximately as well as the t-test, especially when selecting smaller numbers of markers. In particular, RANKCORR shows nearly optimal performance on the ZHENGFILT data set under the classification metrics. It performs poorly according to the clustering metrics when selecting more than 120 total markers, however, though it is still competitive with logistic regression in this domain. Still, the good performance when selecting less than 120 markers supports the notion that RANKCORR is a useful analytical resource for researchers to consider.

2.5.2.4 Marker selection on the 1 million cell 10XMOUSE data set

We consider the 10XMOUSE data set: it consists of 1.3 million mouse neurons generated using 10x protocols [xG]. The "ground truth" clustering that we examine in this case was algorithmically generated without any biological verification or interpretation (see the section about data sets). We include this data set as a stress test for the methods and therefore we do not perform any variable gene selection before running the marker selection algorithms (to keep the data set as large as possible). We also only consider the four fastest and lightest methods (RANKCORR, the t-test, Wilcoxon, and logistic regression) as these are the only methods considered in this work that could possibly produce results in a reasonable amount of time on this data set.

Figure 2.12 shows the classification error of the four methods on the 10XMOUSE data set using the NCC. There are 39 clusters in the "ground truth" clustering that we examine here - thus, the error rates produced by all of the methods are much lower than the error rate expected from random classification. In Figure 2.12, we see that the logistic regression method performs the best overall, and that RANKCORR consistently shows the highest error rate. The largest difference between the RANKCORR curve and the logistic regression curve is only around 3%, however. In addition, as mentioned above, logistic regression is the slowest method by far on this data set - extra accuracy is not worth much if the method is not able to finish running.

Because a biologically motivated or interpreted clustering may be quite different from the clustering used here and because the classification error rate does not capture the full information in a set of markers (and thus similar error rates are not necessarily an accurate indication of the relative performance of methods), it is only possible to conclude that all four methods examined here show similar performance on the 10XMOUSE dataset. The RANKCORR method produces useful markers, runs in a competitive amount of time, and takes a step towards selecting a smart set of markers for each cluster (rather than the same number of markers per cluster). It is impressive

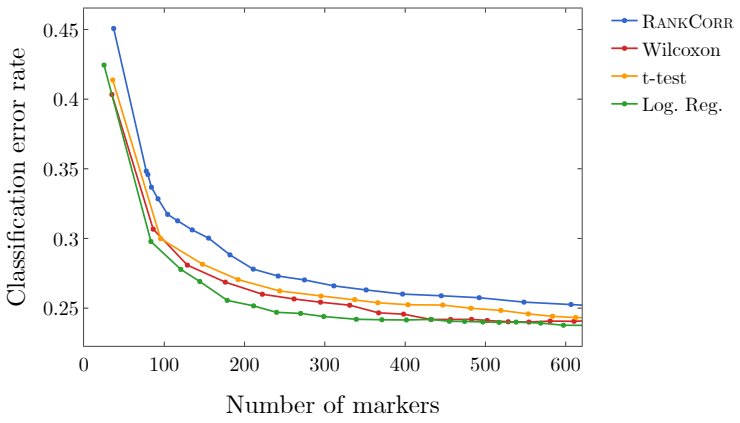


Figure 2.12: Classification error rate under the NCC vs number of markers on the full 10xMouse data set.

that these methods are able to run on such a massive data set.

The implementation of the RFC in `scikit-learn` was quite slow on the large 10xMouse data set, and thus we do not compare the methods via the RFC. From the smaller data sets, we might expect that the random forest classifier produces curves that are shaped similarly to the ones in Figure 2.12 but are shifted down to a lower error rate. This is indeed what we see for the RankCorr method: a comparison of the RFC and the NCC is shown in Figure 2.13. Each point on the RFC curve in Figure 2.13 took over 3 hours on 10 CPU cores to generate; the largest point took over 15 hours. The Louvain clustering method was also too slow to compute any clustering error rates for the markers selected here. This is a situation where the marker selection algorithms are faster than almost all of the evaluation metrics (emphasizing the continued need for good marker set evaluation metrics).

2.5.3 Generating synthetic data based on scRNA-seq data

In order to generate synthetic data that is made to look like an experimental droplet-based scRNA-seq data set, we use the `Splat` method from the R Splatter package (version 1.6.1) [ZPO17] in R version 3.5.0.

We use the data set consisting of purified (CD19+) B cells from [ZTB+17] in order to estimate the `Splat` simulation parameters. In [ZTB+17], the authors analyzed this dataset and saw only one cluster, suggesting that it consists mostly of one cell type. We have also combined it with the full ZhengFull dataset from [ZTB+17] (see the descriptions of the experimental data sets in Section 2.5.1) and observed good overlap with the cluster that the authors identified as B cells in ZhengFull when looking at a two dimensional UMAP visualization. This overlap appears in Appendix A, Figure A.16.

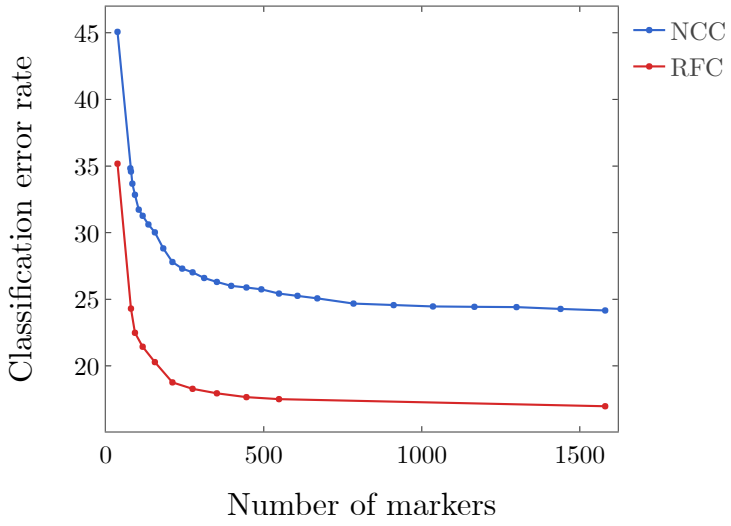


Figure 2.13: A comparison of the nearest centroid classifier (NCC) and the random forest classifier (RFC) using the RANKCORR method on the 10XMOUSE data set

Testing with Splatter showed that including dropout in the `Splat` simulation resulted in a simulated data set with a higher fraction of entries that are 0 than the original dataset. On the other hand, not including dropout resulted in similar fractions of entries that are 0 in the simulated and original datasets. Taking into account the fact that the Splat dropout randomly sets entries to 0 regardless of the size of those entries (a practice that we would argue is an unrealistic representation of actual dropout), we do not include additional dropout in our `Splat` simulations.

In the `Splat` method, differential expression is simulated by generating a multiplicative factor for each gene that is applied to the gene mean before cell counts are created - a factor of 1 means that the gene is not differentially expressed. These multiplicative factors come from a lognormal distribution with location 0.1 and scale 0.4 - the default values in the Splatter package. We have not attempted to tweak these default parameters in this work. Using the default parameters, many of the "differentially expressed" genes have a differential expression multiplier that is between 0.9 and 1.1; for these genes, the gene mean is barely different between the two clusters. This creates a significant number of differentially expressed genes that are difficult to detect.

In our synthetic data sets, we ask for Spatter to simulate two groups: 10% of the genes in the first group are differentially expressed (i.e. have a differential expression multiplier not equal to 1) and none of the genes in the second group are differentially expressed. In this way, all differentially expressed genes can be considered to be marker genes for the first group - there are no overlaps between markers for the first and second groups. The direction of differential expression is randomly determined for each gene. Since the differentially expressed genes are chosen at random, this means that many of the genes that are labeled as differentially expressed in the output data show low

expression levels (often they are expressed in less than 10 cells).

We create 20 different simulated data sets from the CD19+ B cells dataset; see Figure 2.14 for a diagramme of the set-up. For all 20 simulated data sets, we simulate 5000 cells and the same number of genes that we input. The first 10 data sets are created by using the full (unfiltered) information from 10 random samples of 5000 cells from the B cell data set. This procedure results in synthetic data sets that consist of 5000 cells and about 12000 genes (the number of nonzero genes depends on the subsample). We label our results on these data sets under the heading "all genes used for simulation."

To attempt to mitigate the issue of extremely similar gene means between the two clusters in some of the "differentially expressed" genes, we filter the genes of the simulated data via the method introduced in [MBS+15]: namely, place the genes in 20 bins based on their mean expression levels and select the genes with the highest dispersion from each bin. Using this method, we select the top 5000 most variable genes from the simulated data and we then use only these genes for marker selection. In the figures, we report these data under the heading "filtering after simulation."

This type of gene filtering is also common in the literature, and we thought the affect of filtering on marker selection deserved further consideration. Thus, the second 10 simulated data sets are created by using only the top 5000 most variable genes in the original data as the input to Splatter. In this way, we are forcing the differentially expressed genes to look like genes that were originally highly variable. The results on these data are labeled "filtering before simulation."

We again use the `cell_ranger` flavor of the `filter_genes_dispersion` function in the `scanpy` python package for all variable gene selection. Occasionally, this results in only 4999 genes selected; in those cases, we consider 4999 genes (rather than 5000) in the filtered data sets. See the scripts and notebooks on our GitHub repository (link in the data availability statement, Section 2.7.4) for precise information about when this occurred.

2.5.4 Comparison of marker selection methods on synthetic data

We have evaluated RANKCORR on synthetic data sets that are designed to look like experimental scRNA-seq data. In each synthetic data set that we consider, there is a known ground truth set of markers, and all genes that are not markers are statistically identical across the cell populations. Thus, we can present the actual precision of the marker selection methods as well as ROC curves. Precision is an especially important metric for marker selection - it is desirable for an algorithm to select genes that truly separate the two data sets (rather than genes that are statistically identical across the two populations). The values of precision, TPR, and FPR are computed without cross-validation, since the entire set (of genes) in each data set is test data - there is no training to be done. We additionally examine the classification error metric that was introduced in Table 2.1. We still use 5-fold cross-validation to compute this metric.

Since the speed of a marker selection algorithm has been observed as an important factor for use on experimental data, we compare RANKCORR only to the fastest methods: the t-test, Wilcoxon, and logistic regression.

See the Section 2.5.3 for a full description of the data generation process. See also Figure 2.14 for an outline of the design. In short, we generate 20 different synthetic data sets; each simulated data set consists of 5000 cells that are split into two groups, and 10% of the genes are differentially expressed between these groups.

For the purposes of computational efficiency, many data analysis pipelines reduce input data to a subset of the most variable genes before selecting markers. Thus, we examine synthetic data sets that are filtered down to the 5000 most variable genes in addition to unfiltered data sets. In 10 samples, we filter before simulating (and simulate 5000 genes); in the other 10 samples, we simulate without filtering (and simulate as many nonzero genes as there were in the input data, usually around 12000 genes). From each data set that was simulated without filtering, we produce another data set by filtering down to the 5000 most variable genes. This results in three simulation conditions (all genes used for simulation, filtering before simulation, and filtering after simulation) and a total of 30 data sets. See Figure 2.14.

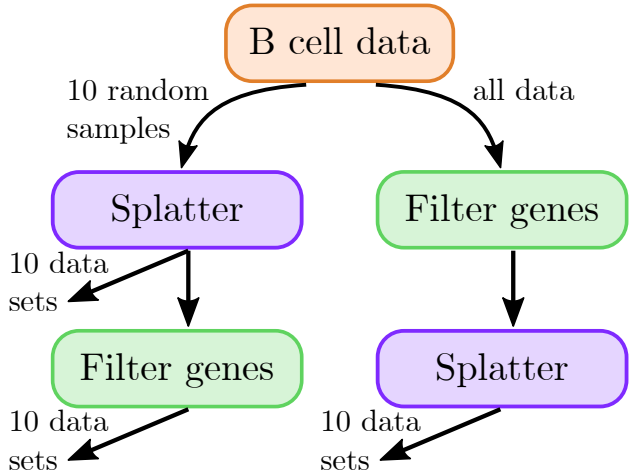


Figure 2.14: Set up of the simulated data. We consider 3 conditions: all genes used for simulation, filtering after simulation, and filtering before simulation. On the left side of this diagramme, we produce 10 data sets by using all genes in simulation, and 10 more by filtering down to the 5000 most variable genes after simulation. These "filtering after simulation" data sets contain a subset of the information from the "all genes used for simulation" data sets. On the right hand side, we produce 10 data sets by filtering down to the 5000 most variable before simulation.

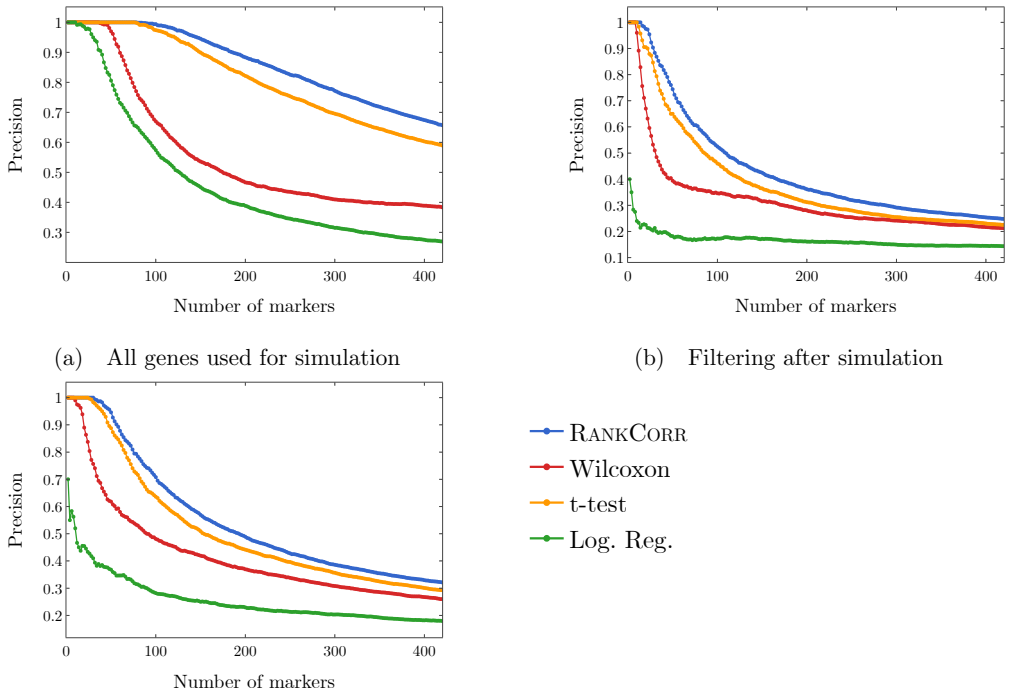
Apart from the t-test data, each curve presented in this section represents the average across all 10 simulated data sets that are relevant to the curve. For the t-test, one of the trials in each simulation condition produced genes with tied p-values. This resulted in situations where it was impossible to

select the top k genes in a stable manner; thus, these data sets were ignored and the t-test precision, TPR, and FPR curves each represent the average of the 9 data sets that are relevant to the curve.

The differentially expressed genes are chosen randomly; thus many of them show low expression levels (often expressed in less than 10 cells) and are difficult to detect. In general, marker selection methods should not select genes with very low expression levels (since these genes are not particularly useful as markers when all cell types have large enough populations). Thus, we do not present information about the recall here.

2.5.4.1 Simulated data illuminates the precise performance characteristics of marker selection methods

In Figure 2.15, we examine the precision of the marker selection algorithms for the first 400 unique genes selected. It is promising to see that RANKCORR produces the highest precision in marker selection across all of the simulation methods. The t-test is second, the Wilcoxon method is third, and logistic regression consistently exhibits the lowest precision.



(a) All genes used for simulation

(b) Filtering after simulation

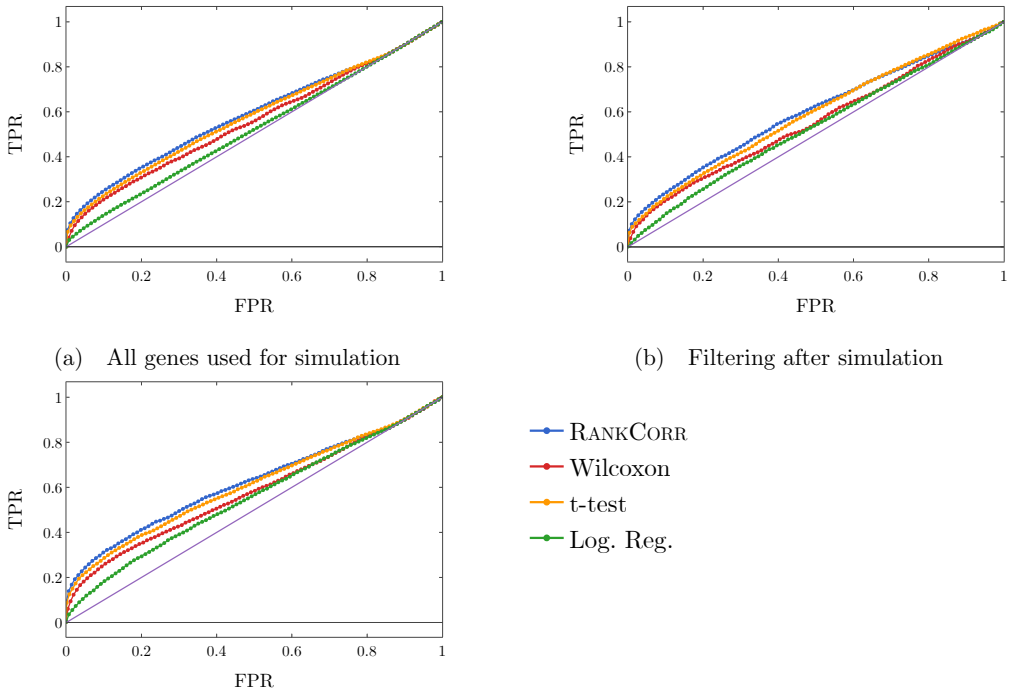
(c) Filtered genes used for simulation

Figure 2.15: Precision of the marker selection methods versus the number of markers selected for the first 400 markers selected. Each sub-figure corresponds to a simulation method and the four lines correspond to the different marker selection algorithms. The RANKCORR method consistently shows the highest precision across all three simulation methods.

Examining Figure 2.15 more closely, we see that the methods generally start off with high

precision that decreases as more markers are selected (each data set contains more than 400 differentially expressed genes). In both of the filtered simulation conditions, all of the methods get close to a precision of 0.1 or 0.2 when 400 markers are selected, and all of the curves are still decreasing at this point (a precision of 0.1 corresponds to random gene selection on these data sets). There are around 2000 differentially expressed genes in the un-filtered simulation condition, so the fact that the precision drops significantly when selecting up to 400 markers indicates proportionally similar behavior to the filtered data sets.

The ROC curves in Figure 2.16 also reflect this behavior: the curves increase (above the diagonal) quite rapidly for a short period of time, but then remain close to the diagonal overall.



(a) All genes used for simulation

(b) Filtering after simulation

(c) Filtered genes used for simulation

Figure 2.16: ROC curves. Each sub-figure corresponds to a simulation method and the four lines correspond to the different marker selection algorithms. The solid (purple) line is the diagonal TPR = FPR.

These data are somewhat expected: in the simulations that come from all of the genes, many of the "differentially expressed" genes show low levels of expression. Thus, we would expect that the ROC curves should end up close to the diagonal as intermediate to large numbers of total markers are selected (since finding these low expression markers should be close to random selection). The filtered data sets could have solved this problem; however, the filtering method used here (see the full synthetic data description in Section 2.5.3) preserves the relative proportions of low- and high-expression genes and (possibly for this reason) do not affect the ROC curves very much.

Another explanation for these difficulties could be the differential expression parameters used in the Splat simulation. With these default parameters, the gene mean for some of the "differentially expressed" genes are only slightly different between the two clusters (for specifics, see Section 2.5.3). Thus, although the simulation may label these genes as differentially expressed, detecting the differential expression by any method will be very difficult. This underscores the differences between biological markers and differentially expressed genes: these differentially expressed genes would not be good practical biological markers, as it would be very difficult to tell two clusters apart based on the expression levels of these types of genes without collecting a lot of data.

Regardless, both of these plots support the notion that the methods are able to easily identify a small set of differentially expressed genes from the synthetic data but then rapidly start to have difficulties as more genes are selected. In addition, the RANKCORR method consistently shows the highest value of precision and TPR.

2.5.4.2 Inconsistent results are obtained when these simulated genes are filtered by dispersion

Comparing Figures 2.15(a-c) across the simulation conditions, we see that the highest precision for each of the marker selection methods is obtained by using all genes for simulation, without any filtering. It is tough to explain why filtering genes by dispersion (the filtering method that we use here; see the synthetic data generation procedure in Section 2.5.4.1) after simulating produces lower precision scores than not filtering. Since the t-test (for example) works by choosing genes based on a p-value score, and the genetic information is not changed by the filtering process (p-values would be the same in both the unfiltered and filtered after simulation data sets), it must be the case that many of the differentially expressed genes are removed from the data set when we filter after simulation. The highly variable genes selected by the filtering method used here are not required to have high expression; thus, there is no obvious reason that many differentially expressed genes should be filtered out.

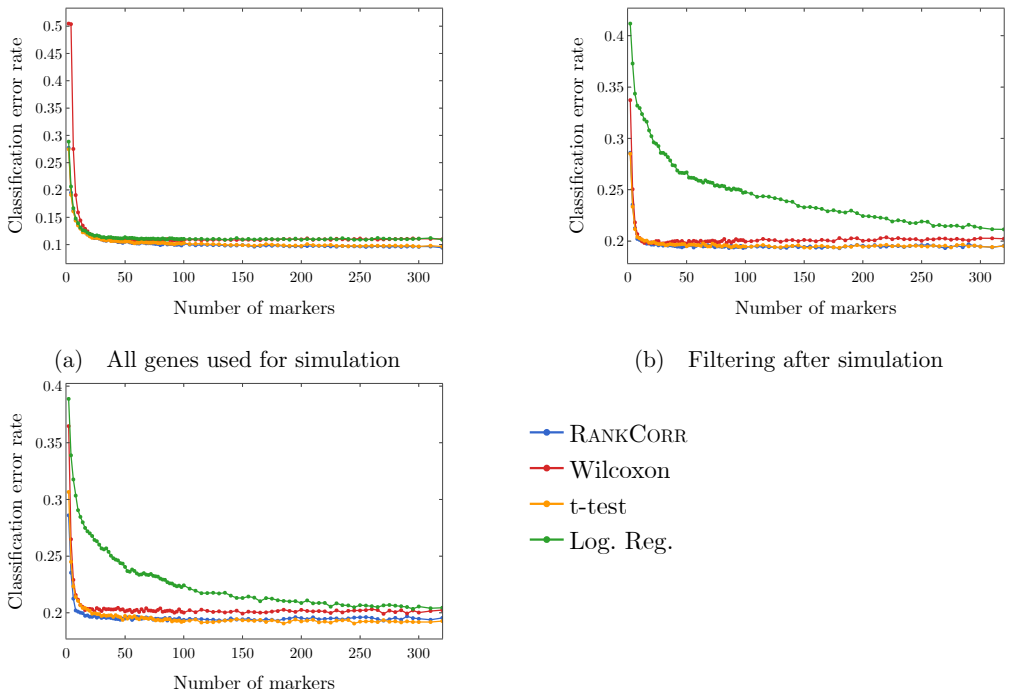
Note that a similar effect is not observed in the ZHENG data sets (see Figures 2.9 and 2.10), suggesting that this inconsistency is an artifact of the simulation methods used here. Simulating scRNA-seq data is itself a difficult task; see also [SR17] for a further discussion of the difficulties involved in simulating scRNA-seq data (and a tool that can help to expose these types of issues). Nonetheless, filtering genes is quite a heuristic process, and there is still more work to be done in fully understanding how this filtering impacts real scRNA-seq data.[5] At the very least, it is clear

[5]It may be possible to filter so as to retain only the interesting genes while reducing the size of the data set (this is the goal, for example, of the different filtering and preprocessing schemes for marker selection in the Seurat package; see the description of Seurat in Section 2.7.1); it is important to carefully consider the effects of these preprocessing choices on the final results.

that the process of filtering genes by dispersion does not commute with the simulation methods used here, since filtering before simulation shows higher precision than filtering after simulation.

2.5.4.3 The classification error rate is an informative but coarse metric

Finally, we examine the classification error rate of the methods applied to the synthetic data in Figure 2.17. It is interesting to note that, with only two clusters, we still misclassify a minimum of around 10% of cells. This suggests that the simulated data are not well separated - the differential expression introduced in the synthetic data is not strong enough to easily separate the two clusters. Moreover, apart from the curves corresponding to the logistic regression method, all of the curves look to be fairly constant after a small number of markers have been selected (approximately 50 for the simulations based on all genes and approximately 30 for the simulations based on filtered data). This further supports the discussion from above - the methods start by quickly choosing a small number of good markers; after this, the genes that are selected do not provide significantly more information about the clustering.



(a) All genes used for simulation

(b) Filtering after simulation

(c) Filtered genes used for simulation

Figure 2.17: Clustering error rates using the Random Forest classifier for the first 500 markers chosen by each method. The sub-figures correspond to different simulation conditions. The RankCorr algorithm consistently produces the smallest values of the clustering error rate.

Note that the methods that show higher precision in Figure 2.15 also show a lower classification error in Figure 2.17. On the other hand, logistic regression shows poor precision levels on the

filtered data sets and also appears significantly worse than the other methods in the classification error rate curves. Thus, according to these experiments, the classification error rate seems to be a coarse but reasonable measure of how well a set of markers describes the data set. In this example, if one methods performs worse than another method according to the classification error rate curves (Figure 2.17), then the same relationship holds in the precision curves (Figure 2.15). Some large differences in precision are eliminated in the classification error rate curves, however, and thus the classification error rate should be considered with a grain of salt. That is, the classification error rate is informative, but it does not provide a full statistical picture of how the methods are actually performing.

2.6 Conclusions

Across a wide variety of data sets (large and small; data sets containing cell differentiation trajectories; datasets with well separated clusters; biologically defined clusters; algorithmically defined clusters) and looking at many different performance metrics, it is impossible (and even inappropriate) to say that any of the methods tested selects better markers than all of the others. Indeed, the marker selection method that was "best" depended on the data set as well as the evaluation metric in question, and the difference in performance between the "best" marker selection algorithm and the "worst" was often quite small.

Thus, the major factors that differentiate the methods examined in this work are the computational resources (both physical and temporal) that the methods require. Since the algorithms show similar overall quality, researchers should prefer marker selection methods that are fast and light.

In addition to this, as technology advances, the trend is towards the generation of larger and larger data sets. High throughput sequencing protocols are becoming more efficient and cheaper, and other statistical and computational methods are improved when many samples are collected. Through imputation and smoothing methods (see e.g. [LL18, vDSN$^+$18, WYY18]), a detailed description of the transcriptome space can be revealed even when low numbers of reads are collected in individual cells. Thus, the speed of a marker selection algorithm will only become more important.

The RANKCORR, Wilcoxon, t-test, and logistic regression methods run the fastest of all of the methods considered in this work. They run considerably faster and/or lighter than any of the complex statistical methods that have been designed specifically for scRNA-seq data. Logistic regression does not scale particularly well with the data set size, however, and it requires an amount of resources that is not competitive with the other three methods on the largest data sets. Moreover, logistic regression exhibits poor performance on several of the data sets considered in this work, especially when selecting small numbers of markers. Thus, as a general guideline, RANKCORR, Wilcoxon, and the t-test are the optimal marker selection algorithms to consider for the analysis of

large, sparse UMI counts data. This recommendation is further bolstered by the fact that these three algorithms tend to perform well in the experiments that we have considered here, especially when selecting lower numbers of markers.

The RANKCORR algorithm, introduced in this work, is the slowest of the three recommended algorithms. Nonetheless, RANKCORR outperformed the other fast algorithms in our synthetic tests. In addition, it provides some interpretability in the multi-class marker selection scenario. Specifically, RANKCORR attempts to select an informative number of markers for each cluster (rather than just a fixed number for each cluster), generally selecting more markers for clusters that we are less certain about. The work of properly selecting sets of markers in a multi-class scenario has not been completed, however, and RANKCORR only proposes one step. Overall, as a fast and efficient marker selection algorithm, RANKCORR is a valuable addition to the set of scRNA-seq analysis tools.

RANKCORR also involves taking a rank transform of scRNA-seq counts data. The rank transformation has other uses in scRNA-seq; it is thus useful to understand the further properties of the rank transformation. These properties are further explored in Chapter 3, but there is further work still to be done as well.

2.6.1 The difficulties of benchmarking and the importance of simulated data

Benchmarking marker selection algorithms on scRNA-seq data is inherently a difficult task. The lack of a ground truth set of markers requires for us to devise performance evaluation metrics that will illuminate the information contained in a selected set of genes. We have examined several natural evaluation metrics in this work; these metrics sometimes produce conflicting results, however. Our experiments herein make it clear that these metrics provide different ways to view the information contained in a set of genes rather than capturing the full picture provided by of a set of markers.

Having a ground truth set of markers available makes the evaluation of marker selection algorithms much more explicit. In the analysis on synthetic data here, for example, it becomes apparent that the methods rapidly select a set of markers that provide a lot of information about the clustering, then essentially start picking things by chance. This type of behavior can only be revealed by a study with a known ground truth.

On the other hand, simulating scRNA-seq data is itself a difficult problem. The simulated data that we consider in this work behaves strangely when we filter it by selecting highly variable genes. In particular, the filtering process considered here seems to remove many of the useful differentially expressed genes in the simulated data. This type of behavior was not observed in the ZHENGFILT experimental data set, where working only with high variance genes had little impact on the marker set evaluation metrics. Better simulation methods, and mathematical results formalizing the quality

of simulated data, are extremely important future projects. See [SR17], [CSG+19] for some work towards these goals.

Finally, in the way that data processing pipelines are currently set up, researchers will often be forced to select markers without the knowledge of a ground truth set of markers. Thus, it may be valuable to consider metrics such as the ones discussed in this work when performing marker selection. Combining the values of several of the metrics may help to aid researchers in deciding when they have selected enough markers to adequately describe their cell types (so that they are not considering genes that were chosen at random), for example. The question of how to stop selecting markers is another important consideration for future work.

2.6.2 The relationship between marker selection and the process of defining cell types

The marker selection framework considered in this work is quite narrow. It is focused on discrete cell types, and (as shown in the PAUL data set) does not handle cell differentiation trajectory patterns very well. Moreover, we assume that the genetic information that we supply to a marker selection algorithm consists of cells that are already partitioned into cell types. This is consistent the data processing pipeline that many researchers currently follow (cluster the scRNA-seq data with an algorithm, then find markers for the clusters that are produced [GWP+15, ZWT+17]); it seems more reasonable to allow for marker selection to help guide the process of finding and defining cell types, however.

For example, future marker selection methods could find markers that are useful for identifying certain regions of the transcriptome space (in an unsupervised or semi-supervised manner). This would allow for clarity along a cell differentiation pathway - at any point on the trajectory, a researcher could view the markers that identify the nearby area, and to what degree each marker identifies the area. Thus, cell types (or differentiation pathways) could be suggested based on marker genes. These cell types might themselves reveal more informative markers, creating an iterative process: let the markers guide the clustering and vice versa. Such a method is known as an *embedded* feature selection method in the computer science literature; adapting an embedded feature selection method to scRNA-seq data is left for future consideration.

2.6.3 Further research directions

Multi-class marker selection with multinomial logistic regression. As alluded to in the introduction to this chapter, only a few true multi-class machine learning methods are currently known. One fairly simple multi-class classifier can be constructed through multinomial logistic regression (MLR; see Section 4.4 in [HTF09]). MLR produces different regression coefficients from one-vs-rest logistic regression (which we considered in this work); these coefficients can then be used to

determine the genes that are significant for each cluster. In addition, methods exist for training a MLR model with ℓ_1 regularization (and neural networks could be used to consider other types of regularization); this would help to create sparse coefficient vectors (that is, ℓ_1 regression could be used to select only a small number of important features).

Like the logistic regression model considered in this work, the MLR model has the advantage that it considers the dependence structure of the genes. The rank correlation between a gene g_1 and a cluster indicator vector τ as calculated by Spearman's ρ is a only a measure of the marginal correlation between g_1 and τ. Even when the marginal correlation between g_1 and τ is 0, the expression of g_1 can give information about τ, however, and some of this dependence is captured by the MLR model. Unfortunately, MLR still does not solve the problem of merging lists of marker genes for different clusters, however.

Note that other multi-class methods could also be applied to the problem of finding marker genes, such as (sparse) multi-class support vector machines. The MLR method is a simple extension of the existing logistic regression marker selection, and it is thus a simple and natural next step for further consideration.

Considering generative models. In this chapter, we make no assumptions about the distribution of mRNA counts and we treat the rank correlation as a purely parametric technique. In particular, we do not make assumptions about the distribution of the rank correlation values.

On the other hand, consider a copula $\mathcal{C}$ for the joint distribution of the mRNA counts and the cluster indicator vector (see [Nel06] for relevant definitions); $\mathcal{C}$ is not necessarily unique since the mRNA counts are discrete (and thus the distribution functions for the counts are discontinuous). In this case, [Neš07] proves that there is a copula $\mathcal{C}_s$ such that the rank correlation between a gene and the cluster indicator vector can be expressed entirely in terms of $\mathcal{C}_s$. Essentially, the rank correlation does not depend on the marginal distributions of the genes; it is only a function of the dependence structure between a gene and the cluster indicator vector. For some intuition, see the discussion in Section 3.2.3: essentially, rank correlation is a measure of the monotonicity of the relationship between two variables. It is reasonable that this monotonicity depends only on the dependence structure between the two variables (and not on the distributions obtained by marginalizing out one of the variables).

Some recent work (e.g. [FSA19]) considers copula models for scRNA-seq data; it would be useful to examine the distribution of rank correlation under these copula models in order to prove precise performance guarantees for the RANKCORR method when it is applied to realistic biological data. This would probably be a difficult task, however. In Chapter 3, we provide a basic analysis of the RANKCORR algorithm by determining some characteristics of the distribution of rank correlation when sparse data is ranked and all ranks are equally likely to occur. Assumptions about

the distribution of rank correlation will force the copula for the joint distribution to satisfy certain characteristics, however. It would thus also be worthwhile to examine what kinds of assumptions can be made about the distribution of rank correlation that will result in a biologically relevant copulas in order to further justify the analysis in Chapter 3.

2.7 Marker selection method implementation details and data availability

2.7.1 Marker selection methods

A summary of the marker selection methods that we consider in this work is found in Table 2.3. In addition, we also implemented SCDE and D^{3}E, but found these two methods to be too slow. We discuss our precise implementation details below. We use Python version 3.7 and R version 3.5.0 unless otherwise noted.

Wilcoxon and the t-test. The t-test and the Wilcoxon rank sum are general statistical methods that aren't specifically designed for RNA-seq data, but they are still often used for the purposes of differential expression testing in the scRNA-seq literature. We use the Python `scanpy` package (version 1.3.7) implementation to find Wilcoxon rank sum and t-test p-values with some editing to the file `_rank_genes_groups.py` to fix several bugs (that are now fixed in the main release). See Section 2.7.4 for how to find this edited file.

Both of these methods produce a score for each gene: when choosing the markers for the clusters, we use the absolute value of this score (so we would chose markers that have a large negative score as well). This is for more direct comparison to the RANKCORR method, where we choose markers by the absolute value of their coefficients. In addition, both of these methods correct the p-values that they produce using Benjamini-Hochberg correction. Finally, we use the version of the t-test in `scanpy` that overestimates the variance of the data.

edgeR and MAST. The methods edgeR and MAST were originally implemented in R. In order to run them with our existing framework, we use the `rpy2` (version 2.9.4) package to access the methods through Python.

Based on the results and scripts from [SR18], edgeR (version 3.24.1) was run using the quasi-likelihood approach (QLF method) on the un-normalized scRNA-seq counts matrix X. For MAST (version 1.8.1), the data matrix X was normalized: the rows of X were scaled so that each row summed to 1 million (to approximate something that looks like "transcripts per million") to create a scaled matrix X^s and then each entry X^s_{ij} of X^s was replaced by $\log(X^s_{ij} + 1)$.

Again following [SR18], we ran both edgeR and MAST in two ways on the PAUL and ZEISEL data sets. In the first way, we only consider the cluster label when fitting the statistical model; these

results are presented in Section 2.5 above. For the second way, we additionally include the fraction of genes that are detected in each cell ("detection rate") as a covariate. We refer to edgeR and MAST run the second way by edgeRdet and MASTdet respectively. According to the marker set evaluation metrics (see Section 2.4), edgeRdet and MASTdet perform similarly to the other methods considered in this work. In addition, edgeRdet (MASTdet) requires slightly more computational resources than edgeR (MAST). For this reason, we choose not to include the edgeRdet or MASTdet results in this manuscript; see Appendix A, Figures A.1-A.6 for that information.

scVI. scVI (version 0.2.4) is implemented in python and utilizes GPUs for faster training. Although the authors provide evidence that their code can handle a data set with one million cells (scVI is tested on the 10XMOUSE data set in [LRC+18]), scVI requires steep computational resources - around 75 GB of RAM to go with one core and one GPU. We were unable to obtain this large amount of memory and a GPU at the same time, so we have been unable to reproduce their results here. One issue is that scVI does not work with sparse data structures (or it makes them dense after loading them); thus, it has been computationally infeasible for us to run scVI on the larger data sets like 10XMOUSE.

Another issue with scVI is that the differential expression methods included in the package are themselves computationally demanding (even after the model has been trained). As far as we can tell, requesting information about differentially expressed genes from a trained scVI instance produces a matrix of size larger than $(10 \cdot n) \times p$, where n is number of cells and p is the number of genes in the original data set. Even restricting to the top 3000 variable genes in the 10XMOUSE data set, this matrix would require around 250 GB of memory to load into storage - in addition to the storage required for the (dense) 10XMOUSE dataset itself. Thus, although it may be possible to train the model on the 10XMOUSE data set, it will be nearly impossible with our computational resources to actually acquire the differential expression information from the trained model.

(An example of the extreme memory used by scVI: the ZEISEL dataset takes approximately 5 MB to store in a dense format. The matrix produced during the differential expression computation method requires 4.1 GB. The actual computation of the Bayes factors - the generalization of a p-value produced by scVI - uses a peak of 15-16GB of memory during processing. This high memory usage does not appear in Figure 2.4 (in Section 2.5) since it is only required for post processing - actually training the scVI model does not require this memory.)

SPA. We examine the performance of the method SPA introduced in [CGC+17] and analyzed further in [Gen15]. As discussed in Section 2.3.2, SPA was the inspiration for this work, and selects markers based on a sparsity parameter s. The features of SPA that are important for the rest of this work are the following: it is a relatively complex algorithm that has two hyperparameters (α, λ, see

Equation (2.6)) that need to be optimized in order to select the best features. In addition, it involves an application/domain specific normalization step that cannot easily be generalized. Finally, the picture to keep in mind is that (for $\alpha = 0$) the method works by solving the optimization problem from [PV13] (Equation (2.3)) on standardized data - thus, essentially, we are finding an optimal separating hyperplane with a sparse normal vector (a "sparse separating hyperplane").

Algorithm 7 Marker selection according to [CGC+17]

1: **procedure** SPA(X, an $n \times p$ matrix of counts; $\tau \in \{\pm 1\}^n$; s, the sparsity parameter; $\alpha \in [0, 1]$, an interpolation parameter; $\lambda > 0$, a scaling parameter)
2: Let X' be a normalized version of X. Let x'_i denote the i-th row of X'.
3: $\bar{x} \leftarrow \frac{1}{n} \sum_{i=1}^{n} x'_i$
4: For all $1 \leq i \leq n$, let $x^c_i = x'_i - \bar{x}$ ▷ Center the data
5: For all $1 \leq \ell \leq p$, let $\sigma_\ell \leftarrow \sigma(X_\ell)$.
6: For all $1 \leq \ell \leq p$, let $\rho_\ell \leftarrow$ empirical correlation of X_ℓ with τ.
7: For all $1 \leq \ell \leq p$, let $\alpha_\ell \leftarrow \alpha^{2(1-|\rho_\ell|)}$.
8: Let $x^{\text{std}}_{i,j} \leftarrow \alpha_j \lambda x^c_{i,j} + (1 - \alpha_j) \frac{x^c_{i,j}}{\sigma_j}$ ▷ Standardize the data by weighted interpolation
9: **return** SELECT(X^{std}, τ, s)
10: **end procedure**

The full SPA is method presented in Algorithm 7. Line 2 is specific to the empirical data in question - in [CGC+17], the authors consider mass spec data, and they must ensure that it all has a common baseline (see section 3.2.1 of [Gen15]). In the example of scRNA-seq data, the total number of reads per cell from a group of cells of the same type (i.e. from the same cluster) will vary widely due to technical issues in the experimental procedure and thus this information is removed during the normalization step. Any normalization method could be used in Line 2.

Line 8 in Algorithm 7 creates a convex combination (different for each feature) of the standardized data and the centered data (see also Equation (2.6)). This is an attempt to take advantage of the correlation of the features with the vector τ. If $|\rho_\ell|$ is close to 1 (i.e. feature ℓ is strongly correlated with τ), then α_ℓ will be close to 1 (from line 7). Following line 8, this will make things so that the data are centered (we ignore the effects of standardizing) and we will be able to take full advantage of the structure of the data. On the other hand, if $\alpha_\ell = 0$, then the data are fully standardized. This can help to get rid of large peaks in the data that are not particularly informative (i.e. the feature with the large peak has low correlation with the vector τ) while preserving smaller peaks that are informative (i.e. the feature with the low peak has high correlation with the vector τ).

The choice of the interpolation parameter α controls the "speed" to which the exponents α_ℓ move towards 1 as ρ_ℓ increases. Smaller values of α make it so that α_ℓ stays closer to 0 even when ρ_ℓ is fairly large (so we mostly standardize the data); larger values of α emphasize the centered data. The choice of the scaling parameter λ changes the emphasis on the centered data and is very

important in keeping the "balance" between the two terms. If the data are large overall, the centered data could overpower the standardized data even when α_ℓ is small; on the other hand, if the data are quite small, then the standardized data could overpower the centered data. See Section 3.2.3 of [Gen15] for further discussion of these parameters.

The fact that SPA has two hyperparameters that we are required to tune causes SPA to take considerably longer than RANKCORR to run for a fixed value of s. Moreover, in a situation with no known ground truth, it is unclear what metric we would like to optimize when selecting these hyperparameters. For the current evaluation, we have minimized the classification error rate using the NCC (see information about the marker set evaluation metrics in Section 2.4), but it is not clear that this would be the best metric to optimize in general. We choose the NCC classifier since the RFC exhibits some variance (see Appendix A, Figure A.13) - thus, optimizing the classification error rate according to the RFC classifier would produce an unstable set of markers (performing the optimization again would result in a different set of markers). We choose to optimize the supervised classification error (rather than one of the unsupervised clustering metrics) for the sake of speed - optimizing a slower evaluation metric would increase the computation time required for the SPA marker selection method.

Another inconvenience of the SPA method is that the hyperparameters affect the number of markers that are selected for a fixed value of s. This makes the number of markers selected by SPA method more inconsistent and unpredictable. For example, it has occurred that the "optimal" (in terms of minimizing the classification error rate using the NCC, as discussed above) choice of hyperparameters for sparsity parameters $s_1 > s_2$ has resulted in a smaller number of markers selected for s_1 than the "optimal" choice of hyperparameters for s_2. That is, increasing s can lead to selecting smaller numbers of markers.

Elastic nets. The SPA method introduced in [PV13] and discussed above is similar to an L_1- and L_2- regularized SVM without an offset (i.e. it finds a sparse separating hyperplane that is assumed to pass though the origin, the instinct for this is given near Equation (2.5) in the setup of the rank correlation). Thus, we also compare the performance of RANKCORR to that of the Elastic Nets version of LASSO: a least squares method with both L_1 and L_2 regularization [ZH05]. Elastic Nets has two regularization parameters that need to be tweaked in order to find the optimal set of features; this requires extra cross-validation and therefore we are only able to run on the smaller PAUL and ZEISEL data sets. Although the `scikit-learn` (version 0.20.0) package contains a method for finding the regularization parameters by cross-validation, it still takes a significant manual effort in order to find a range of the regularization parameters that capture the full possible behavior of the system but will also allow for the objective function to converge (in a reasonable number of iterations) the majority of the time. The timing information presented in Figure 2.4 only

represents the run time of the method, and does not take into account this (time consuming) process of manipulating the data.

Another feature to note about the cross-validated elastic nets method is that it is (intentionally) a sparse method. Thus, scores are only generated for a small number of genes in each cluster - the genes that are specifically deemed "markers" for that cluster. It is not possible to compare the relative utilities of the genes that are not considered markers - each of those genes are given a score of 0. Thus, beyond a certain number of genes, it is not possible to get any more information from the markers selected by the elastic nets method. (You cannot, for example, request a "bad" marker in order to combine it with the information from other "good" markers).

Logistic regression. Logistic regression is proposed as a method for marker selection in [NYMP18]. Specifically, a regression is performed on each gene using the cluster label as the response variable. This is translated into a p-value via a likelihood ratio (comparing to the null model of logistic regression). The `scanpy` (version 1.3.7) package includes this method, and thus we are able to run it on sparse data. We again have made some updates to the file `_rank_genes_groups.py` in the `scanpy` package to fix some slight errors (that are now fixed in the main release); see Section 2.7.4 for where to find this edited file.

RANKCORR. The RANKCORR algorithm is introduced in this paper; the precise procedure is discussed in Section 2.3. It is important to note that the implementation of RANKCORR that we use here has not been fully optimized. Note that the major step (2) of the SELECT algorithm (Algorithm 1) essentially consists of computing the dot product of each column of a data matrix with the cluster labels τ. The only other time consuming portion of SELECT is computing the ℓ_2 norm of a vector. These types of linear algebraic computation have fast implementations that are accessible from python (e.g. `numba` or `TensorFlow` using GPUs). We have not yet optimized the method to take advantage of all possible speed ups since RANKCORR runs quickly enough in our trials.

Random marker selection. As a sanity check, we select markers by choosing genes uniformly at random (the same number of markers for each cluster). All of the other methods presented in this work outperform random marker selection by significant margins. For the sake of clarity, the performance of random marker selection is relegated to Appendix A (Figures A.1-A.11).

Seurat. The commonly-used Seurat data analysis package [BHS+18] contains implementations of several methods that we consider here, including the Wilcoxon method, the t-test, logistic regression, and MAST. The default method for selecting markers in Seurat is the Wilcoxon method combined with some gene pre-filtering that is designed to speed up the computations. Potentially,

the Seurat method could perform better than the standard Wilcoxon rank-sum test (considered in the manuscript) if the researcher is careful to manually tune the different gene filtering thresholds to eliminate uninteresting genes that have high Wilcoxon scores. The Seurat documentation warns that this type of filtering may also eliminate marker genes with weaker differential expression signals, however. We thus do not examine tuning these parameters in this manuscript. See [BHS+18] or the Seurat website (`https://satijalab.org/seurat/`) for further information.

SCDE. The SCDE package (we examined version 2.6.0) is implemented in R. Our testing found that it was too slow to be used on real scRNA-seq data sets: it was taking approximately one minute per cell to fit the model (on one core). Since we are performing 5-fold cross-validation, we would need to fit the model approximately 5 times. On one of the smaller data sets (PAUL or ZEISEL), this would require approximately 250 hours of computer time; it would be infeasible to train on the larger data sets. Since we are specifically developing methods for use with the large data sets that are appearing more often, we have excluded SCDE from our final analysis.

D^{3}E. D^{3}E is also implemented in Python (version 2.7), but it has no support for sparse data structures; thus, running on the 10XMOUSE data set would require a very large amount of memory. Although the method allows for splitting the data into smaller segments (to allow for parallel computation), the full data set needs to be loaded into memory when initializing the process. In addition, when running on the PAUL data set using the faster method-of-moments mode, D^{3}E took about 25 minutes running on 10 cores (about 4 hours and 10 minutes total CPU time) to find markers for one cluster (vs the rest of the population). Since we need the p-values for all (19) clusters for all 5 folds, this method would require approximately 40 hours on 10 cores. Although this is faster than SCDE, this would still be too slow to run on the larger data sets, and thus we exclude D^{3}E from our final analysis as well. All of our testing was carried out using the D^{3}E source on GitHub (commit `efe21d1`).

COMET. The COMET method [DSC+19] is a (streamlined) brute force approach for examining the predictive power of "gene panels" (sets containing up to four genes). COMET inherently has different goals than the majority of the marker selection methods discussed here, and it can not (currently) select more than four genes for a given cell type. Restricting to a very small number of selected markers is useful when the cell types are well-known and a researcher wishes to perform further experimental analysis; COMET is indeed used to guide FACS sorting in [DSC+19]. The brute force computational costs do not make COMET a useful tool for data exploration or for providing feedback towards the veracity of a potential clustering, however. In fact, when tested on the PAUL data set, the COMET method took an average of eight minutes to rank pairs of markers

for a fixed cluster (resulting in around 2.5 hours of total run time for all 19 clusters; this would be approximately 12.5 hours to run on all five folds). In addition to this, as of this writing, COMET requires all data to be input as text files or 10x expression files - these are not (especially) sparse formats, and this further limits the data sets that can be studied with this tool (for example, the 10x data files for the full ZHENG dataset require around 600MB of disk space, while the sparse version requires about 100 MB). We thus do not include the full COMET method in our comparisons in this manuscript.

As mentioned in Section 2.1, however, COMET is based on a statistical test (the XL-mHG test, see [Wag17]) that has some desirable properties for scRNA-seq data. In addition to considering combinations of genes, COMET also ranks individual genes by the average of their XL-mHG p-values and (the logarithm of) their fold-changes. These ranks (or the related p-values) could be used in a similar fashion to the other differential expression methods considered in this manuscript to select markers for the data sets (e.g. select the top five genes for each cluster). We have elected not to examine this method, however, since we are working with several other differential expression methods based on statistical tests (e.g. Wilcoxon, the t-test).

scGeneFit. The scGeneFit method is introduced in the preprint [DVME19]. As mentioned in Section 2.1, this method attempts to discover markers that are informative about a clustering as a whole rather than determining markers for individual cell types. Specifically, it finds a set of genes M such that the projection of the data onto M is "optimal." In this case, optimality means that the distance (after projecting) between different clusters is larger than an input parameter (as much as possible). This optimal projection is defined by a linear program: the variables correspond to genes, and the constraints enforce separation between clusters (there is one constraint for each pair of cells in different clusters). For the sake of efficiency, the current implementation of scGeneFit (`https://github.com/solevillar/scGeneFit-python`, commit `32dd6a1`) considers a random subsample of the constraints; thus, scGeneFit can be applied to data sets of arbitrary size if selecting lower quality markers is acceptable. The trade-offs between efficiency and marker quality are not yet fully explored in the preprint [DVME19], however (for example, the number of constraints required for quality should probably be related to the number of clusters; this method of random sampling also seems to deemphasize rare cell types).

We ran scGeneFit using parameters suggested on the scGeneFit GitHub page and altering the number of constraints so that the method ran in a reasonable amount of time on the PAUL and Zeisel data sets (approximately 10 minutes to run on one fold). The results that we obtained were somewhat suboptimal, however, especially on the PAUL data set. Additionally, the supercomputer architecture used for our analysis changed before we collected data using scGeneFit; thus, comparisons of the timing of scGeneFit to the other methods would not be valid. Therefore, further considering that

the scGeneFit [DVME19] is still presented in a preprint (and is thus subject to significant future change), we report our scGeneFit results in Appendix A rather than the main manuscript.

2.7.2 Generating marker sets of different sizes from algorithms other than RANKCORR

For a fixed data set, we need to select markers for a given clustering - not just markers for a single cluster. Here we describe how we select a specific number of markers and how we merge lists of markers for individual clusters to make a marker list for the entire clustering.

For a differential expression method, we proceed in a one-vs-all fashion: letting C denote the number of clusters in the given clustering, we use the differential expression methods to find C vectors of p-values; the i-th vector corresponds to the comparison between cluster i and all of the other cells. For the sake of simplicity, we then include an equal number of markers for each cluster to create a set of markers for the clustering.

For example, we consider the classification error rate (see Section 2.4.2) when the marker list consists of the three genes with the smallest p-values from each cluster (with duplicates removed). As previously mentioned, this is a vast oversimplification of a tough problem - how to merge these lists of p-values in an optimal way, making sure that we have good representation of each cluster - but it allows for us to quickly and easily compare the methods that we present here. (Note that we would probably want to choose more markers for a cluster in which all p-values were large - we probably need more coordinates to distinguish this cluster from all of the others, even if those coordinates are not particularly informative. Thus, setting a p-value threshold could potentially perform worse than the method outlined here, as we may not select any markers for a certain cluster with a thresholding method.) The process of merging lists of p-values is left for future work.

For the elastic nets method, which selects one list of markers as optimal (without giving a score for all of the markers), we apply a similar strategy to approximate selecting a small number of markers. In particular, we choose an equal number of markers with the highest score for each cluster until we run out of markers to select. For example, when attempting to select 20 markers per cluster, we may include the top 20 markers for one cluster and all 18 of the markers that are selected for a different cluster.

2.7.3 Ranking the performance of the methods in Figure 2.3

The numbers in the cells of the tables in Figure 2.3 are meant to provide an approximate ranking of the marker selection methods as different numbers of markers are selected. The supervised classification metrics are ranked separately from the unsupervised clustering metrics; that is, each column in a table in Figure 2.3 contains *two* rankings of the marker selection methods. A rank of 1 corresponds to the best method; larger rank values indicate worse performance. Since these

numbers are ranks, they do not capture the magnitude of the gaps in performance between methods. For example, if two methods differ by one rank (e.g. the method ranked two vs the method ranked three), there could be a large gap in performance between the two methods. The colors of the cells are meant to capture the larger differences between (tiers of) methods.

Each rank is meant to capture the results of multiple evaluation metrics when the methods select a range of markers (six metrics for the classification table cells and three metrics for the clustering table cells - see Table 2.1; the ranges of numbers of markers appear in the first row of each table in Figure 2.3). We thus first calculate a score summarizing the metrics in a range of selected markers. Here, will call these the AoM scores, so called because they are based on an Average of Medians. Briefly, for a fixed method and a fixed range of markers, the AoM score is computed as an average across the relevant evaluation metrics. Each quantity in the average is the median (computed over the fixed range of markers) of the performance of the method according to one of the relevant evaluation metrics. That is, for a fixed method and a fixed range $[a, b]$ of markers, we define

$$\mathrm{AoM} = \mathrm{Average}_{m \in \mathrm{metrics}} \left\{ \mathrm{median}_{[a,b]} \{ m(\mathrm{method}) \} \right\} \tag{2.10}$$

Some considerations about the definitions of the AoM scores is discussed in more detail in the following paragraphs. The ranks are then determined from the relative AoM scores of the different methods.

We calculate the AoM scores from the data points that appear in Figure 2.5–Figure 2.11 without any extrapolation (that is, the curves themselves are not considered in these calculations, only the points). For this reason, the AoM scores are quite sensitive to the marker ranges (the bins in the top row of the tables in Figure 2.3): since the performance of the methods improve as more markers are selected, methods that have a data point towards the right edge of one of the marker ranges will generally be rated more favorably by summary calculations. This effect will be especially pronounced when analyzing the leftmost portions of Figures 2.5–2.11 (that is, when considering small sets of markers), where the curves are generally improving rapidly. To partially account for this issue, the AoM scores are based on the median performance value produced by the methods in a range of markers.

We choose to average these medians as a way to summarize them in our computation of the AoM scores. This is mainly due to the fact that we want each evaluation metric to contribute equally to the summary score. For example, for a fixed method, if one evaluation metric produces poor results while the other metrics are okay, we still want to include information about this outlier in our AoM score. As discussed previously, the evaluation metrics all produce different ways of examining the information contained in a set of markers; it is thus a warning sign if even one metric is poor. The evaluation metrics are on the same scale (all produce values from 0 to 1, with larger values

indicating better performance), so standardization is not needed.

We report ranks in Figure 2.3 (rather than the raw AoM scores) since the values of these AoM scores are relatively uninformative and difficult to read (for example, a difference of 0.02 between two AoM scores is quite large - it represents a 2% difference between the two methods). The actual AoM values can be found in the data in the GitHub repository related to this paper (`https://github.com/ahsv/marker-selection-code`).

2.7.4 Data availability

The experimental data sets analysed during the current study are publicly available. They can be found in the following locations:

- ZEISEL is found on the website of the authors of [ZMMC+15]: `http://linnarssonlab.org/cortex/`. The data are also available on the GEO (GSE60361).
- PAUL is found in the `scanpy` python package - we consider the version obtained by calling the `scanpy.api.datasets.paul15()` function. The clustering is included in the resulting `Anndata` object under the heading `paul15_clusters`. The data are also available on the GEO (GSE72857).
- ZHENGFULL and ZHENGFILT are (subsets) of the data sets introduced in [ZTB+17]. The full data set can be found on the 10x website (`https://support.10xgenomics.com/single-cell-gene-expression/datasets/1.1.0/fresh_68k_pbmc_donor_a`) as well as on the SRA (SRP073767). The biologically motivated bulk labels can be found on the `scanpy_usage` GitHub repository at `https://github.com/theislab/scanpy_usage/blob/master/170503_zheng17/data/zheng17_bulk_lables.txt` (we use commit `54607f0`).
- 10XMOUSE is available for download on the 10x website (`https://support.10xgenomics.com/single-cell-gene-expression/datasets/1.3.0/1M_neurons`. The clustering analysed in this manuscript can be found on the `scanpy_usage` GitHub repository (`https://github.com/theislab/scanpy_usage/tree/master/170522_visualizing_one_million_cells`; we consider commit `ba6eb85`)

The synthetic data analysed in this manuscript is based on the CD19+ B cell data set from [ZTB+17]. This B cell data set can be found on the 10x website at `https://support.10xgenomics.com/single-cell-gene-expression/datasets/1.1.0/b_cells`. The synthetic data sets themselves are available from the author on request.

All scripts that were used for marker selection and data processing can be found at the GitHub repository located at `https://github.com/ahsv/marker-selection-code`. This includes the implementations of SPA and RANKCORR.

CHAPTER 3

Theoretical Analysis of the Rank Transformation and the RANKCORR Algorithm

In this chapter, we present a theoretical analysis of the RANKCORR algorithm. We split our analysis into three separate parts.

First, in Section 3.1, we provide further analysis of the correctness and complexity of the RANKCORR algorithm. Specifically, we prove that the SELECT method (Algorithm 1) obtains the correct solution to the optimization (2.3). We then show that SELECT requires time $\mathcal{O}(n^2)$ on an scRNA-seq data set containing genetic information from n cells. Assuming that the data are sparse, we also provide a method for ranking a vector of gene counts in time $\mathcal{O}(n)$ on average; using this method, we show that RANKCORR also requires $\mathcal{O}(n^2)$ time on average.

In Section 3.2, we establish some general properties of the rank transformation and provide an elementary statistical exploration of rank space. Sparse integer count data are found in a myriad of applications including voting data, traffic patterns, online behavior statistics, as well single cell RNA-sequencing. Ranking these data provides a non-parametric way to consider the counts, and exploits the sparsity to make certain types of analysis more simple. The empirical evidence presented in Chapter 2 suggests that ranking scRNA-seq data is an informative technique for analyzing this type of sparse data. Our goal in this section is to lay a theoretical foundation for understanding how the rank transformation acts on sparse data and to propose several properties of scRNA-seq data that make it amenable to analysis via the rank transformation.

Finally, in Section 3.3, we build upon the framework presented in Section 3.2 in order to more accurately characterize the types of markers that are selected by RANKCORR. We accomplish this by constructing a distance measure on rank space that corresponds to the Spearman rank correlation between two points. Under this distance, the optimal marker selected by RANKCORR is the point in rank space closest to a vector that is determined by the group of cells in question. We finish this section by providing bounds on the minimum distance the optimal gene will be located from the group indicator vector with the assumption that the collected genetic data is sparse.

3.1 Analysis of RANKCORR

The RANKCORR algorithm is presented in Algorithm 3. As discussed above, the method starts by ranking (and standardizing) the input data. The heavy lifting behind marker selection is then carried out in the SELECT algorithm (Algorithm 1). Thus, in this section, we prove results about the correctness and the time complexity of the SELECT algorithm. These results easily extend to bounds on the asymptotic time complexity of the RANKCORR algorithm.

3.1.1 The correctness of SELECT

Recall that SELECT, presented in Algorithm 1, finds the optimal solution $\hat{\omega}$ the optimization 2.3, reproduced below:

$$\begin{aligned} \hat{\omega} &= \arg\max_{\omega} \sum_{i=1}^{n} \overline{\tau}_i \langle \overline{x}_i, \omega \rangle \\ &\text{subject to } \|\omega\|_2 \leq 1, \|\omega\|_1 \leq \sqrt{s} \end{aligned} \tag{3.1}$$

where $\overline{\tau}$ is (related to) a measured signal (a cluster indicator vector), $\overline{X}$ is (related to) a sparse measurement matrix (in Chapter 2, X contained scRNA-seq counts), $\overline{x}_i$ is the i-th row of $\overline{X}$, and s is an input sparsity parameter. We require that $s \geq 1$ for the constraint set to be nontrivial.

In section 2.3.2.2, we state the fact that the solution $\hat{\omega}$ to (2.3) is obtained by a normalized soft thresholding of the vector $v = \sum_{i=1}^{n} \bar{\tau}_i \bar{x}_i$ (see Equation 2.4). This is why SELECT works with the vector $Y(v, \beta) = T_\beta(v)/\|T_\beta(v)\|_2$ where T_β is the soft-thresholding operator defined in Equation (2.4). Select starts with the vector $Y(v, \beta_m)$, where β_m is chosen so that $Y(v, \beta_m)$ is not the zero vector but has as few nonzero entries as possible. All entries of $Y(v, \beta_m)$ must be equal; if $\beta_g > \beta_m$, then either $Y(v, \beta_g) = 0$ is the zero vector or $Y(v, \beta_g) = Y(v, \beta_m)$. For example, if the entry of v that is largest in magnitude is unique, then β_m can be chosen to be slightly less than $\|v\|_\infty$ (so that β_m is still larger in magnitude than all of the other entries of v). SELECT proceeds by iteratively decreasing β (that is, soft thresholding v by smaller and smaller values) following the magnitudes of the elements of v until it finds the first coordinate value β^* where $\|Y(v, \beta^*)\| \geq \sqrt{s}$. Since only the support of $\hat{\omega}$ is required for feature selection, SELECT returns as soon as β^* is found. The exact solution $\hat{\omega}$ can be quickly determined from the knowledge of β^*, though this calculation is not presented here.

To show that SELECT is correct, it is sufficient to show that the 1-norm of the vector $Y(v, \beta)$ increases monotonically as β decreases. This implies that $\|Y(v, \beta_m)\|_1$ is the minimum possible nonzero value of $\|Y(v, \beta)\|_1$; moreover, if[1] this minimum value is smaller than $\sqrt{s}$, then SELECT will appropriately decrease β (increasing the ℓ_1 norm) until it finds the normalized soft-thresholding

[1] We handle the corner cases after the proof of Proposition 3.1.1.

of v with ℓ_1 norm equal to $\sqrt{s}$. That is, SELECT correctly finds the support of $\hat{\omega}$.

Proposition 3.1.1, below, tells us that we decrease the ℓ_1 norm of a vector when we soft threshold and standardize that vector. Note that if $\beta_2 < \beta_1$ then there is a value $\gamma > 0$ such that $Y(v, \beta_1) = Y(Y(v, \beta_2), \gamma)$. Thus, Proposition 3.1.1 implies that

$$\|Y(v, \beta_2)\|_1 \geq \|Y(Y(v, \beta_2), \gamma)\|_1 = \|Y(v, \beta_1)\|_1, \tag{3.2}$$

which is the full monotonicity result we desire.

Let $B_n = \{x \in \mathbb{R}^n \colon \|x\|_2 = 1\}$.

Proposition 3.1.1. *For all $x \in B_n$ and for any $0 \leq \beta \leq \|x\|_\infty$ we have that*

$$\|Y(x, \beta)\|_1 \leq \|x\|_1.$$

Proof. Fix an arbitrary $x \in B_n$; and assume that x is labeled in such a way that $x_i \geq x_{i+1}$ for all i. For a given value β, let s_β be the number of nonzero entries in $Y(x, \beta)$. We will prove the claim for x by considering $\|Y(x, \beta)\|_1$ as a function of β for $0 \leq \beta \leq \|x\|_\infty$. Note that

$$\|Y(x, \beta)\|_1 = \frac{\|T_\beta(x)\|_1}{\|T_\beta(x)\|_2} \quad \text{and so the claim is equivalent to} \quad \frac{\|T_\beta(x)\|_1}{\|x\|_1} \leq \|T_\beta(x)\|_2.$$

Apply the Cauchy-Schwarz inequality to x and $T_\beta(x)$ to obtain that

$$\|T_\beta(x)\|_2 \geq \sum_{i \in [s_\beta]} |x_i|(|x_i| - \beta) \tag{3.3}$$

and thus it suffices to show that

$$\sum_{i \in [s_\beta]} |x_i|(|x_i| - \beta) \geq \frac{1}{\|x\|_1} \sum_{i \in [s_\beta]} (|x_i| - \beta). \tag{3.4}$$

Note that both sides of (3.4) are piecewise linear functions of β. So let $f(\beta) = \sum_{i \in [s_\beta]} |x_i|(|x_i| - \beta)$ and $g(\beta) = \frac{1}{\|x\|_1} \sum_{i \in [s_\beta]} (|x_i| - \beta)$. In addition, let m denote the smallest nonzero value of x and $p = \|x\|_0$ be the number of nonzero elements in x. Note the following:

- $f(0) = \sum_i x_i^2 = 1 = \|x\|_1 / \|x\|_1 = g(0)$.
- $f(\|x\|_\infty) = 0 = g(\|x\|_\infty)$.
- Both f and g have corners whenever $\beta = x_i$ for some i and are differentiable otherwise. In addition, f and g are both continuous.

- When $\beta \neq x_i$ for any i, we have that $f'(\beta) = -\sum_{i \in [s_\beta]} |x_i|$ and $g'(\beta) = -s_\beta / \|x\|_1$. These slopes are monotonically increasing as β increases (since they are all negative and get smaller in magnitude).

- Assume that $\beta_1 < \beta_2$ with $\beta_1, \beta_2 \neq x_i$ for any i. If $|f'(\beta_1)| \geq |g'(\beta_1)|$, then $|f'(\beta_2)| \geq |g'(\beta_2)|$. That is, if f is ever decreasing more quickly than g, then it will continue to decrease more quickly than g for larger values of β.

  To see this, note that the quantity

  $$\frac{1}{s_\beta} \sum_{i \in [s_\beta]} |x_i| \tag{3.5}$$

  monotonically increases with increasing β. This is due to the facr that 3.5 is the average of the s_β largest values of x. As β increases, s_β decreases, so we will look at the average of increasingly larger terms.

  Thus, if $\sum_{i \in [s_{\beta_1}]} |x_i| \geq s_{\beta_1} / \|x\|_1$, then we have that

  $$\frac{1}{s_{\beta_2}} \sum_{i \in [s_{\beta_2}]} \geq \frac{1}{s_{\beta_1}} \sum_{i \in [s_{\beta_1}]} \geq \frac{1}{\|x\|_1} \tag{3.6}$$

  and from this, it follows that $\sum i \in s_{\beta_2} |x_i| \geq s_{\beta_2} / \|x\|_1$. That is, $|f'(\beta_2)| \geq |g'(\beta_2)|$ as desired.

- When $0 < \beta < m$, we have that $f'(\beta) = -\|x\|_1$ and $g'(\beta) = -p/\|x\|_1$. Since $x \in B_p$, we have that $\|x\|_1 \leq \sqrt{p}$ and thus $-g'(\beta) \geq p/\sqrt{p} = \sqrt{p} \geq -f'(\beta)$. Thus, g decreases more quickly than f when $0 < \beta < m$, which means that $f(\beta) \geq g(\beta)$ on this domain.

From all of the above, we can conclude that $f(\beta) \geq g(\beta)$ for all $0 \leq \beta \leq \|x\|_\infty$. We know that $f(\beta) \geq g(\beta)$ for $0 \leq \beta \leq m$. Thus, if there is a value β_a with $f(\beta_a) < g(\beta_a)$, then we would need for there to be a value $\beta_b \leq \beta_a$ with $|f'(\beta_b)| > |g'(\beta_b)|$ since both f and g are piecewise linear. That is, f would need to decrease more quickly than g on one of the linear segments for f to become smaller than g. But, as we showed above, in this case we would have that $|f'(\beta)| \geq |g'(\beta)|$ for all $\beta > \beta_b$ with $\beta \neq x_i$ for all i. That is, f would be decreasing at least as quickly as g for all values of $\beta > \beta_a$. Thus, g could never catch up to f, and we could not have that $f(\|x\|_\infty) = g(\|x\|_\infty)$. This means that β_a cannot exist, which means that $f(\beta) \geq g(\beta)$ for all $0 \leq \beta \leq \|x\|_\infty$. This concludes the proof of the statement (3.4), which means that the Proposition is proven. □

To address some corner cases, let β_m be defined as in the discussion preceding Proposition 3.1.1. If $\|Y(v, \beta_m)\|_1 \geq \sqrt{s}$, then the optimal solution to $\hat{\omega}$ is given by $\sqrt{s} \cdot Y(v, \beta_m) / \|Y(v, \beta_m)\|_1$. That is, in this case SELECT should return all of the features that are nonzero in $Y(v, \beta_m)$. This is

accomplished of in line 14 of Algorithm 1. On the other hand, if $\|Y(v,0)\|_1 < \sqrt{s}$, then SELECT should return all features. This case is also found in line 16 of Algorithm 1. Both of these cases will be extremely rare in the case of scRNA-seq data, where the vector v is constructed in terms of the rows of X which are fairly random gene expression profiles (and thus v is likely to have a unique largest entry).

3.1.2 Algorithm run times

Here, we determine the asymptotic runtime of the RANKCORR algorithm. As previously discussed, the algorithm SELECT is at the heart of RANKCORR. The following proposition provides the precise asymptotic runtime of SELECT:

Proposition 3.1.2. *Given an input matrix $X \in \mathbb{R}^{n\times p}$,* SELECT *(defined in Algorithm 1) has runtime $\mathcal{O}((p+n)n)$*

Proof. For an input matrix $A \in n \times p$, the addition step (Line 2 in Algorithm 1) will require $\mathcal{O}(np)$ operations: np multiplications to weight the entries of X by the elements of τ and np additions to add the weighted rows together. The weighted rows of X are added together into a vector $v \in \mathbb{R}^n$, which is then sorted in Line 3, taking time $\mathcal{O}(n \log n)$ in the worst case. Lines 8-12 of SELECT then loop over the entries of v; in each step of the loop, we compute the 1-norm and the 2-norm of a soft-thresholded version of v and thus each step takes time $\mathcal{O}(n)$. Thus, (worst case) asymptotic time complexity of SELECT is $\mathcal{O}(pn + n^2 + n\log n) = \mathcal{O}((p+n)n)$. □

For the purposes of scRNA-seq, it is reasonable to assume that $p = o(n)$ - we are collecting information about a fixed number p of genes (usually around $20,000$ to $30,000$ genes) from many (n) cells (currently, up to 10^6 cells). Moreover, as sequencing technology improves, the number n of cells sequenced in one experiment will increase (essentially without bound), whereas p will never increase by very much (since there are only a limited number of genes in an organism's genome). Thus, for the purposes of scRNA-seq, this time complexity bound can be simplified: SELECT runs in time $\mathcal{O}(n^2)$.

In RANKCORR (Algorithm 3), the extra work comes from applying the rank transform Φ to the data. Given a vector $w \in \mathbb{R}^n$, performing the rank transformation on w (computing $\Phi(w)$) will require worst case time $\mathcal{O}(n \log n)$. This is due to the fact that, when the entries of w are all unique, computing $\Phi(w)$ is equivalent to sorting w. Thus, performing the rank transform on the p columns of X will require time $\mathcal{O}(pn \log n)$. The other calculations done to standardize the ranked data matrix require time $\mathcal{O}(pn)$. Thus, including the time requires in the call to SELECT, RANKCORR requires time $\mathcal{O}((p+n)n + pn\log n)$. Again applying the assumption that $p = o(n)$, we obtain that $\mathcal{O}((p+n)n + pn\log n) = \mathcal{O}(n^2 \log n)$; that is, performing the rank transformation is the most costly step of the algorithm.

There are sorting algorithms that require average time $\mathcal{O}(n)$ on inputs of length n as well as non-comparative sorting algorithms that require worst case time $\mathcal{O}(n)$ when certain parameters can be chosen correctly (e.g. radix sort) - using one of these sorting algorithms would reduce the time required by RANKCORR to $\mathcal{O}(n^2)$. Here, we utilize the sparsity of X, along with hashed dictionaries (that have average lookup time $\mathcal{O}(1)$) to provide an implementation of the rank transformation on a (sparse) vector $w \in \mathbb{R}^n$ with average time complexity $\mathcal{O}(n)$. This method is outlined in Algorithm 8, resulting in an average time complexity of $\mathcal{O}((p+n)n)$ for both the SELECT and RANKCORR algorithms. We have found that our implementation of Algorithm 8 run faster than using the sorting functions built in to Python or the ranking functions in the `numpy` package.

Algorithm 8 Computing Φ using a hashed dictionary

1: **procedure** FASTRANK($w \in \mathbb{R}^n$, a vector with all non-negative entries.)
2: Set $C \leftarrow \{\}$ to be an empty dictionary.
3: Let U be a list of the unique entries in w.
4: $U \leftarrow \text{sort}(U)$ so that $U_0 \leq U_i$ for all i.
5: $C[U_i] \leftarrow 0$ for all $0 \leq i \leq n$.
6: **for** each entry $w_i \in w$ **do**
7: $C[w_i] \leftarrow C[w_i] + 1$ ▷ Count the number of times each entry appears
8: **end for**
9: Set $R \leftarrow \{\}$ to be an empty dictionary.
10: $r \leftarrow 0$
11: **for** $i = 0$ to $i = n$ **do**
12: $R[U_i] \leftarrow r + \frac{C[U_i]+1}{2}$ ▷ Use the counts to compute the ranks of the unique values
13: $r \leftarrow r + C[U_i]$
14: **end for**
15: Set $v \leftarrow 0 \in \mathbb{R}^n$ be the zero vector.
16: **for** $i = 0$ to $i = n$ **do**
17: $v_i \leftarrow R[w_i]$ ▷ Populate the vector of ranks
18: **end for**
19: **return** v, the ranked version of w.
20: **end procedure**

Proposition 3.1.3. *Let $W \subset \mathbb{R}^n$ be a collection of vectors, and assume that there is value ϵ such that, for all $w \in W$, we have that $\|w\|_0 \leq n^{1-\epsilon}$. Using hashed dictionaries that have $\mathcal{O}(1)$ expected time complexity for lookups, the* FASTRANK *algorithm (defined in Algorithm 8) has expected time complexity of $\mathcal{O}(n)$ on the elements of W.*

Proof. Algorithm 8 starts by letting U be a list of the unique entries of the input w in line 3; this requires a loop through the entries of w, taking time $\mathcal{O}(n)$. Note that, by assumption, the length of U is smaller that $n^{1-\epsilon}+1$. U is sorted in Line 4, requiring worst case time $\mathcal{O}(n^{1-\epsilon}\log(n^{1-\epsilon})) = \mathcal{O}(n)$. Next, we count the number of times each unique entry of w occurs by populating a dictionary C.

This subroutine appears in lines 5-8 and requires a loop through the entries of w. At each step in the loop, we make one lookup in C. Thus, this loop has an average time complexity of $\mathcal{O}(n)$. We then populate a dictionary R of the ranks taken by the values of the unique entries of w in $\Phi(w)$. This similarly requires a loop that has an average time complexity of $\mathcal{O}(n)$ - at each step in this loop, we perform a constant number of arithmetic operations and lookups to the dictionary C. Finally, we define the vector $v = \Phi(w)$ in a loop that requires n lookups to the dictionary R. Thus, the total average time complexity of Algorithm 8 is given by $\mathcal{O}(n)$. □

3.2 Properties of rank space

In this section, we switch the focus from the RANKCORR algorithm to the rank transformation. In particular, we establish some general properties of the rank transformation, provide a geometric framework for understanding the space of ranked points, and establish some fundamental statistical properties of ranked points. We use these statistical properties in Section 3.3 to determine some foundational results about the markers that are preferred by RANKCORR.

For convenience, we reproduce Definition 2.2.1 here. Consider a vector $x \in \mathbb{R}^n$. For a given index i with $1 \leq i \leq n$, let $S_i(x) = \{\ell \in [n] \colon x_\ell < x_i\}$ and $E_i(x) = \{\ell \in [n] \colon x_\ell = x_i\}$ (note that $i \in E_i(x)$).

Definition. *The* rank transformation *is the transformation* $\Phi \colon \mathbb{R}^n \to \mathbb{R}^n$ *defined by*

$$\Phi(x)_i = |S_i(x)| + \frac{|E_i(x)| + 1}{2}. \tag{3.7}$$

We refer to Φ as the rank transformation due to the fact that $\phi(x)_i$ is the index of x_i in an ordered version of x (i.e. it is the *rank* of x_i in x). If multiple elements in x are equal, we assign their ranks to be the average of the ranks that would be assigned to those elements (that is, for fixed i, all elements x_j for $j \in E_i(x)$ will be assigned the same rank).

In the definition above, the codomain of Φ is given as $\mathbb{R}^n$. From the defining equation (3.7), however, we see that each entry of $\Phi(x)$ ($x \in \mathbb{R}^n$) is either an integer or a half-integer. In the following, we refer to the image of $\mathbb{R}^n$ under Φ as *rank space* and denote it by RS^n.

3.2.1 Alternate characterizations of rank space

In this section, we show that the points in RS^n are in bijective correspondence with both ordered set partitions of $[n]$ as well as weak orders on $[n]$. A good portion of the existing literature on the rank transform approaches the material from the ordered set partition or weak order point of view (for example, determining consensus in elections with rank choice voting [EM02, ACN08, Ail08, GLPR11, MF19]), and thus we have chosen to introduce these points of view here. In the remainder

of this chapter, we will inherently use the ideas behind weak orders and ordered set partitions to help describe concepts, statistics, and distances on rank space.

Definition 3.2.1. *Consider a set T. A (finite)* ordered set partition *P of T is a (finite) partition $\{T_i\}_{i\in I}$ (that is, $|I| < \infty, \cup_{i\in I} T_i = T$, and $T_j \cap T_i = \emptyset$ for all $i \neq j$) and a linear order $<$ on the sets in the partition (that is, for all $i, j \in I$, we have either that $T_i < T_j$ or $T_j < T_i$). We will write $P = (\{T_i\}_{i\in I}, <)$.*

Definition 3.2.2. *Consider a set T. An* weak order *on T is a binary relation $<$ that is transitive and irreflexive (i.e. for all $v \in V$ we do not have $v < v$). Moreover, for all $v, w, z \in T$, if v is incomparable with w and w is incomparable with z, then v is incomparable with z.*

Proposition 3.2.3. *RS^n is in bijective correspondence with the set of ordered set partitions on $[n]$ as well as the set of weak orders on $[n]$.*

Proof. Let P denote the set of ordered set partitions of $[n]$ and let W denote the set of weak orders on $[n]$. We will start by providing a correspondence between P and W.

To do this, consider an relation $<\in W$. Define a partition $T = \{T_i\}_{i\in I}$ of $[n]$ given by the "incompatibility" equivalence classes of $<$: for a fixed $i \in I$, two elements x and y are in T_i if neither $x < y$ nor $y < x$ (that is, x and y are incompatible). From Definition 3.2.2, we know that this "incompatibility relation" is reflexive and transitive; it is also symmetric. Thus, $\{T_i\}_{i\in I}$ is a partition of $[n]$.

Define the order $<_p$ on T by $T_i <_p T_j$ if there is $x \in T_i$ and $y \in T_j$ such that $x < y$. This relation $<_p$ is a linear order on T: if there is $x \in T_i$ and $y \in T_j$ such that x and y are incompatible under $<$, then $i = j$ by the definition of T.

Let $\phi\colon W \to P$ denote the map defined above that sends $<$ to $(\{T_i\}_{i\in I}, <_p)$. By walking through the definitions, we see that ϕ is injective. Specifically, given two relations $<^1 \neq <^2 \in W$, without loss of generality there must be a pair $x, y \in [n]$ such that $x <^1 y$ and either $y <^2 x$ or x and y are incompatible under $<^2$. Let $\{M_i\}_{i\in I}$ (respectively $\{N_j\}_{j\in J}$) denote the equivalence classes of $<^1$ (respectively $<^2$) under the incompatibility relation, as discussed above. Consider i_1 and i_2 such that $x \in M_{i_1}$, $y \in M_{i_2}$. Note that $M_{i_1} <^1_p M_{i_2}$. Likewise, consider j_1, j_2 such that $x \in N_{j_1}$ and $y \in N_{j_2}$. We either have that $j_1 = j_2$ or that $N_{j_2} <^2_p N_{j_1}$. Thus, even if $\{M_i\}_{i\in I}$ and $\{N_j\}_{j\in J}$ are the same partition of N, we will have that $<^1_p \neq <^2_p$: that is, $(\{M_i\}_{i\in I}, <^1_p)$ and $(\{N_j\}_{j\in J}, <^2_p)$ cannot be the same elements of P.

A mapping $\psi\colon P \hookrightarrow W$ can be obtained in a similar fashion. Specifically, given an ordered set partition $(T_{ii\in I}, <_p)$ of $[n]$, define the weak order $<$ on $[n]$ by $x < y$ if there are i, j such that $x \in T_i$, $y \in T_j$ and $T_i <_p T_j$. Since $<_p$ is a linear order, we have that $<$ is transitive and irreflexive. Moreover, since $\{T_i\}_{i\in I}$ is a partition of $[n]$, the incompatibility relation for $<$ is an equivalence

relation. Thus, $<$ is a weak order on $[n]$. Similar calculations to those above show that ψ is also injective. This establishes a correspondence between P and W.

We now provide a correspondence between RS^n and P. Given a point $z \in RS^n$, consider the partition of $[n]$ given by $\{E_i(z)\}_{i\in[n]}$ (E is defined in Definition 2.2.1). Then define the (linear) order $<_p$ on $\{E_i(z)\}_{i\in[n]}$ by $E_i <_p E_j$ if and only if $z_i < z_j$. This is a surjective map. To see that it is also injective, note that the set $S_j(x)$ (given in Definition 2.2.1) can be written in terms of the sets $E_i(z)$: specifically,

$$S_j(z) = \bigcup_{i:\, z_i<z_j} E_i(z). \tag{3.8}$$

Thus, following Definition 2.2.1, the point $z \in RS^n$ depends entirely on the sets $E_i(z)$ and their relative ordering. □

These characterizations yield the size of rank space. Indeed, it is known (see, for example, [Goo75]) that the exponential generating function for $|P|$ and $|W|$ is given by

$$\sum_{n=0}^{\infty} |RS^n| \frac{x^n}{n!} = \frac{1}{2 - e^x}. \tag{3.9}$$

A specific contour integration of that generating function can be used to show ([Goo75, Bai98]) that

$$|RS^n| = \frac{n!}{2(\ln 2)^{n+1}} + \mathcal{O}\left(\frac{n!}{(2\pi)^n}\right). \tag{3.10}$$

which is to say that, for large values of n, $|RS^n|$ is approximately the exponential factor of $\frac{1}{(\ln 2)^{n+1}}$ larger than $n!$. The points in RS^n can also be represented by labeled young diagrams with n boxes.

Example: If $n = 5$, an ordered set partition of $[5]$ is given by $\{3\} < \{1, 2, 5\} < \{4\}$. This corresponds to any point in $x \in \mathbb{R}^5$ with $x_3 < x_1 = x_2 = x_5 < x_4$ and, given such an x, $\Phi(x) = (3, 3, 1, 5, 3)$. We could represent $\Phi(x)$ by either of the following ordered Young tableaux:

$$\begin{array}{|c|}\hline 3 \\ \hline \end{array}\!\!\begin{array}{|c|c|c|}\hline 1 & 2 & 5 \\ \hline \end{array}\!\!\begin{array}{|c|}\hline 4 \\ \hline \end{array} \quad \text{represents the same point as} \quad \begin{array}{|c|}\hline 3 \\ \hline \end{array}\!\!\begin{array}{|c|c|c|}\hline 5 & 1 & 2 \\ \hline \end{array}\!\!\begin{array}{|c|}\hline 4 \\ \hline \end{array} \tag{3.11}$$

Note that, when referring to a point in RS^n, the order of the numbers that fill a row in the ordered Young tableau does not matter. In addition, note that a permutation of the labels of the Young tableau corresponds to a permutation of the elements of $\Phi(x)$.

Definition 3.2.4. *Given $z \in RS^n$, let D be a ordered Young tableau that corresponds to z. Define the* shape *of z (or the shape of D), denoted by $sh(z)$, as the composition of the integer n associated*

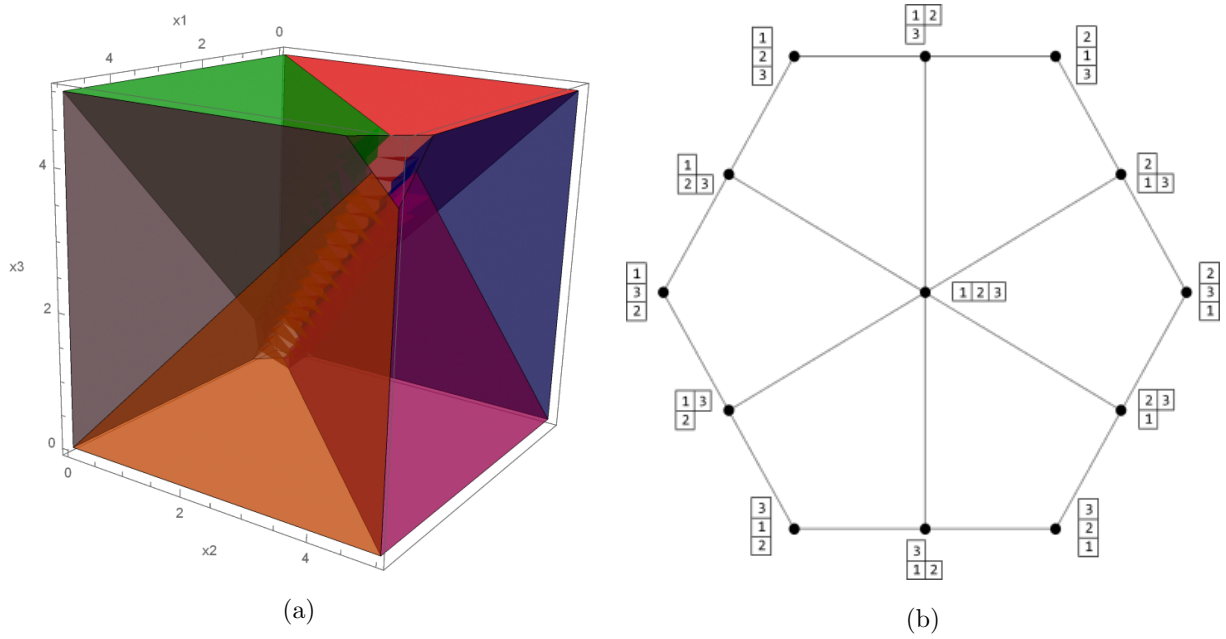
with D and define the partition *of z, denoted by $\Lambda(z)$, as the partition of n associated with D. Note that the partition of z is the shape of z listed from largest to smallest.*

Example: As in the previous example, let $z = (3, 3, 1, 5, 3) \in RS^5$ and let

$$D = \begin{array}{|c|c|c|c|}\hline 3 & & & \\ \hline 1 & 2 & 5 & \\ \hline 4 & & & \\ \hline \end{array}. \tag{3.12}$$

Then we have that $sh(z) = (1, 3, 1)$ and $\Lambda(z) = (3, 1, 1)$. We will often write $sh(z) = 131$ and $\Lambda(z) = 311$.

3.2.2 Further observations and motivation for sparse data



(a) (b)

Figure 3.1: Rank space in 3 dimensions. See the text for further description

Figure 3.1 provides two depictions of RS^3. Figure 3.1(a) shows $\mathbb{R}^3$ sliced into six 3-dimensional segments by the planes $x_1 = x_2$, $x_1 = x_3$, and $x_2 = x_3$. Each point in any of the six 3-dimensional regions is on the same side of all three of the planes. For example, one of the regions is $R = \{x \in \mathbb{R}^3 \colon x_1 > x_3, x_1 < x_2, x_2 > x_3\}$. Given a point $x \in R$, we see that $x_2 > x_1 > x_3$. Thus, each point in R will have the same image under Φ. That is, each 3-dimensional region is mapped to one point by the rank-transformation.

The half-planes between each of the 3-dimensional regions represent areas where two coordinates are equal and the third is different (for example, the half-plane with $x_2 = x_1 > x_3$ is adjacent to the region R, defined above). Thus, each of these half-planes is also mapped to a point by Φ.

Finally, the line through the origin (with $x_1 = x_2 = x_3$) is also mapped to a point by Φ. That is, $|\Phi(\mathbb{R}^3)| = 13$.

Figure 3.1(b) shows RS^3 with corresponding diagrams for each point. The corners, with the diagrammes of shape 111, correspond to the three dimensional regions in Figure 3.1(a), the points with diagrammes of shape 21 or 12 correspond to the half-planes between the three dimensional regions in Figure 3.1(a), and the point with shape 3 corresponds to the line through the center in Figure 3.1(a). For example, the region R (discussed above) is mapped to the point with diagramme $\begin{array}{|c|}\hline 3 \\ \hline 1 \\ \hline 2 \\ \hline \end{array}$ in located in the bottom left of Figure 3.1(b).

This figure helps to illustrate several other properties of the rank transformation. In particular, Φ is non-linear and highly distorts the underlying geometry. Moreover, Φ acts non-locally. For example, the point $x = (10, 11, 1) \in R$ is closer to the point $y = (10, 10, 1) \notin R$ than the point $v = (300, 4000, 60) \in R$ under the Euclidean distance. However, $\Phi(v) = \Phi(x) \neq \Phi(y)$. In fact, two points can be arbitrarily close in Euclidean distance and be mapped to different points by Φ. Likewise, two points can be arbitrarily far away under Euclidean distance and mapped to the same point by Φ.

Note that the rank transform also serves to attenuate outliers. For example, if $x = (1, 2, 3, 4, 5)$ and $y = (1, 2, 3, 4, 100)$, then $\Phi(x) = \Phi(y) = (1, 2, 3, 4, 5)$. That is, if x_m is the largest value of x (i.e. $x_m \geq x_i$ for all i) and x_ℓ is the second largest element of x (assuming that $x_m > x_\ell \geq x_i$ for all i such that $x_i \neq x_m$) then $\Phi(x)_m - \Phi(x)_\ell$ is independent of the difference $x_m - x_\ell$.

Furthermore, the rank transform will tend to expand tightly grouped data. Certainly, we see that $|\Phi(x)_i - \Phi(x)_j| \geq 1$ for all i and j with $x_i \neq x_j$ (so if there are any elements x_i and x_j of x with $|x_i - x_j| < 1$, then these two elements will be pushed apart by Φ). Going further, given $x \in \mathbb{R}^n$, assume that $|E_i| = m$ for some i; in this case, we will have that $|\Phi(x)_i - \Phi(x)_j| \geq \frac{m+1}{2}$ for $j \notin E_i$. Thus, when many entries of x are equal to one fixed value (i.e. when m is large), $\Phi(x)$ will contain a significant gap between the entries that were equal and the surrounding entries (the gap will be $\frac{m+1}{2}$).

This type of behavior intuitively seems desirable when dealing with some types of sparse data. Consider a sparse integer data matrix X, and assume that the data contain a significant number of large outliers - some features have been detected many times in a small number of samples. These outliers appear in many real examples (such as scRNA-seq data) and often have a strong effect when working in Euclidean space (i.e. using $\|\cdot\|_2$; this is a known problem in least squares regression). In this context, a given feature X_i will tend to have many 0 counts (it will not be observed in many of the samples), and then some nonzero entries.

Following the preceding discussion (see also Definition 2.1), this means that $\Phi(X_i)$ will contain a large gap between the entries of X_i that were 0 and the entries of X_i that were 1, and (generally) a

smaller gap between the entries of X_i that were 1 and the entries of X_i that were 2. Then, the larger counts in X_i - the outliers, which will tend not to repeat very often - will all be close to each other. See Figure 2.1 for an example with scRNA-seq data.

Thus, by applying the rank transformation, we put a strong emphasis on the difference between no expression of a feature and some expression of that feature. Once a feature is observed, however, we are more interested in high expression versus low expression: the actual difference in counts (say between a count of 500 and a count of 1000) will be de-emphasized in the ranked data. It is important to note that here we are inherently assuming that the entries of X that are 0 are informative; that is, $X_{ij} = 0$ indicates that the actual count of feature j in sample i should be close to 0 (it is not often something large that hasn't been examined). If the counts of 0 are not informative, then the large gap between a count of 0 and a count of 1 will not be accurately capturing true patterns in the data.

Another way to think about RS^3 and Figure 3.1 is in the context of the RANKCORR algorithm: if only 3 samples are taken, and we apply the rank transform to each feature, then we are assuming that each feature is one of only 13 types. We are calling two features equivalent if they fall into the same region in Figure 3.1; that is, if the respective ranks of the elements in the samples are the same. According to Equation 3.10, this will not be a limitation on high dimensional data sets: we are choosing to discretize the features based on their relative expression patterns.

3.2.3 The geometry of rank space under rank correlation

Based on Figure 3.1, we see that RS^3 appears to have an appealing geometric structure. Therefore, in this section, we discuss some attempts to define distance metrics on RS^n and create a geometric framework on rank space. A notion of distance is a measure of dissimilarity; if two points are a large distance apart, this means that they are quite dissimilar. Thus, it is reasonable that we would be able to obtain a distance from a measure of *similarity*.

Measuring the similarity between two rankings is a problem that has been extensively studied under the guise of rank correlation [EM02, Car09, GLPR11, vDE12]. In particular, a rank correlation is a statistic calculated for a pair of rankings (points in RS^n) for which a value of $+1$ indicates perfect alignment and -1 indicates complete disagreement. Any rank correlation coefficient ρ can thus be converted into a measure of dissimilarity via the transformation $d(z, y) = 1 - \rho(z, y)$ (note that $d \geq 0$ by construction), though this will not usually produce a metric. In particular, many correlation coefficients in the literature are symmetric and satisfy $\rho(z, y) = 1$ if and only if $z = y$ (perfect alignment is only obtained when the rankings are the same), but the triangle inequality is not respected.

Relaxing the triangle inequality requirement allows us to consider Spearman's ρ, which is given

by

$$\rho(z, y) = \frac{\langle z - \mu(z), y - \mu(y) \rangle}{\|z - \mu(z)\|_2 \|y - \mu(y)\|_2} \tag{3.13}$$

for $z, y \in RS^n$. This is symmetric and efficient to compute. Moreover, $\rho(z, y)$ is the cosine of the angle between $(z - \mu(z))/\|z - \mu(z)\|_2$ and $(y - \mu(y))/\|y - \mu(y)\|_2$; thus, comparing points in RS^n based on ρ is equivalent to comparing the angles between the points with respect to the center of RS^n (see Proposition 3.2.5). From this, we see that the corresponding similarity measure $d_\rho = 1 - \rho$ will be given by

$$d_\rho(z, y) = \left\| \frac{z - \mu(z)}{\|z - \mu(z)\|_2} - \frac{y - \mu(y)}{\|y - \mu(y)\|_2} \right\|_2^2 \tag{3.14}$$

which is also easily interpretable as (half of) the squared Euclidean distance between z and y projected onto a sphere of radius 1 around the center of RS^n.

Another appealing property of d_ρ is that it is *neutral*: that is, given two points $y, z \in RS^n$, and a permutation $\tau \in \mathfrak{S}_n$, we have that $d(y, z) = d(\tau(y), \tau(z))$. That is, each element of $\mathfrak{S}_n$ acts on RS^n by an isometry according to d_ρ. This is appealing due to the fact that we are looking at the distance between two rankings without any regard to the objects being ranked; if the underlying objects are rearranged but their rankings are kept the same, then the distance between the rankings should not change. For example, in the case of scRNA-seq data, we should be able to permute the cells without changing the markers that we select.

For the rest of this chapter, we refer to Spearman's ρ simply as the rank correlation. As discussed in Section 2.3.2.3, RANKCORR selects the markers that exhibit the largest (in magnitude) rank correlation with a cluster indicator vector τ. Following the preceding discussion, consider a copy of RS^n that has been centered (i.e. shifted to the origin) and then projected onto the sphere of radius 1. In this shifted version of RS^n, RANKCORR will select the feature that is nearest to the point corresponding to $\Phi(\tau)$ (or it's antipode) according to the (squared) Euclidean distance.

Spearman's ρ has received much attention in the literature, and much is known about the statistical properties and behavior of ρ in the large sample limit ([HP36, KKS39, Ken48]). Unfortunately, there has been limited work on the properties of ρ in the presence of ties. In our case, we are assuming that the data are sparse integer counts; therefore, we expect many ties. In Section 3.3, we examine the properties of Spearman's ρ when ties are allowed, specializing to the high sparsity case whenever we can.

3.2.3.1 Digression: true metrics on RS^n

It is reasonable to consider metrics (defined in the usual mathematical sense) on RS^n that could be used for other applications. We will restrict ourselves to neutral metrics; that is, metrics that

satisfy the neutral condition discussed above.

In our presentation, an element $z \in RS^n$ is a point in $\mathbb{R}^n$. The ℓ_p metrics are neutral metrics on $\mathbb{R}^n$; thus, they can be restricted to RS^n to provide distances on RS^n. In particular, the ℓ_1 metric restricted to RS^n has received significant study under the name of *Spearman's footrule* [DG77]. Though these derived distances are quite natural, they do not attempt to capture the structure of a ranking.

A neutral metric based on a rank correlation is presented in [EM02]; this metric counts the number of "half-flips" (creating ties from adjacent elements or breaking ties into two adjacent ranks) needed to move from one ranking to another in an appropriate fashion. Computing this metric involves an $n \times n$ matrix product, however; this is costly as the sample size n gets large. In our examples, this would be nearly impossible when considering one million cells ($n = 10^6$), both from the perspective of time and memory. Thus, in this work, we focus on the distance derived from Spearman's ρ for the sake of computation speed, even though this distance d_ρ is not a metric.

3.2.4 Basic statistical properties of points in rank space

In this section, we examine some of the statistical properties of the rank transformation. We use the framework established in this section to prove bounds on Spearman's ρ; this helps to further understand the distance introduced in Section 3.2.3 and to characterize the types of features that are preferred by the RANKCORR method in Section 3.3.

Recall that $\mu(z)$ denotes the mean of z and $\sigma(z)$ denotes the standard deviation of z. Here we show that $\mu(z)$ is constant for all $z \in RS^n$ and we develop an understanding for how $\sigma(z)$ behaves on RS^n. These types of results have been previously considered for rankings that do not allow ties (i.e. when considering linear orders rather than weak orders) [HP36, KKS39, Ken48]. The addition of ties adds complexity that requires new techniques for dealing with the points in rank space.

Proposition 3.2.5. *For all $x \in \mathbb{R}^n$ we have that $\mu(\Phi(x)) = \frac{n+1}{2}$ is a constant.*

Proof. The idea: if all of the elements of x are distinct, then $\mu(\Phi(x)) = \frac{1}{n}\sum_{i=1}^{n} i = \frac{n+1}{2}$. Next, suppose that x_i is equal to j other elements of x (i.e. $|E_i| = j + 1$) and that there are m elements of x which are smaller than x_i (i.e. $|S_i| = m$). Then $\phi(x)_i = m + \frac{j+1+1}{2}$ and so

$$\sum_{k \in E_i} \Phi(x)_k = (j+1)\left(m + \frac{(j+1)+1}{2}\right) = (j+1)m + \sum_{k=1}^{j+1} k = \sum_{k=1}^{j+1} m + k \tag{3.15}$$

Thus, $\{\phi(x)_k \colon k \in E_i\}$ contribute the same amount to the sum in $\mu(\Phi(x))$ as they did in the case where all elements of x were distinct. □

Proposition 3.2.5 shows that, by applying the rank transformation to a data set, we are essentially centering the coordinates, since all points have the same mean. Thus, the convex hull of RS^n has dimension at most $n-1$ (the vector of all 1s is orthogonal to this convex hull, once it is centered).

Proposition 3.2.6. *Given two points y and z in RS^n with $\Lambda(y) = \Lambda(z)$, we have that $\sigma(y) = \sigma(z)$.*

Proof. Let $\lambda = sh(y)$ be the shape of y. We may assume that $sh(z) = \tau(\lambda)$ where τ is an adjacent transposition in $\mathfrak{S}_n$, the symmetric group on n items. Assume that $\tau = (i, i+1)$ swaps i and $(i+1)$. Let $\lambda_i = a$ and $\lambda_{i+1} = b$. Let $\sum_{j=1}^{i-1} \lambda_j = m$.

We have that $\mu(y) = \mu(z) = \frac{n+1}{2} = \mu$ by Proposition 1. In the computation of $\sigma(y)$, under the square root, we have that there are

$$\begin{array}{lll} a & \text{terms equal to} & (m + \frac{a+1}{2} - \mu)^2 \\ b & \text{terms equal to} & (m + a + \frac{b+1}{2} - \mu)^2 \end{array}$$

Likewise, in the computation of $\sigma(z)$, under the square root, we have that there are

$$\begin{array}{lll} b & \text{terms equal to} & (m + \frac{b+1}{2} - \mu)^2 \\ a & \text{terms equal to} & (m + b + \frac{a+1}{2} - \mu)^2 \end{array}$$

These are the only terms that differ in the computations of $\sigma(y)$ and $\sigma(z)$. A computer algebra system can be used to verify that

$$a(m + \frac{a+1}{2} - \mu)^2 + b(m + a + \tfrac{b+1}{2} - \mu)^2 = b(m + \tfrac{b+1}{2} - \mu)^2 + a(m + b + \tfrac{a+1}{2} - \mu)^2. \quad (3.16)$$

and thus $\sigma(z) = \sigma(y)$. □

It remains to understand how the standard deviations of points y and z in rank space are related when $\Lambda(y) \neq \Lambda(z)$. Intuitively, we would expect for $\sigma(y)$ to be less than $\sigma(z)$ when many elements of y are equal to each other: that is, when $\Lambda(y)$ starts with several large elements compared to $\Lambda(z)$. To formalize this, consider the majorization order on partitions: given two partitions λ and ρ of the integer n, we say that λ *majorizes* ρ, written $\lambda \succeq \rho$, if $\sum_{i=1}^{k} \lambda_i \geq \sum_{i=1}^{k} \rho_i$ for all $1 \leq k \leq n$.

Proposition 3.2.7. *Consider two points y and z in RS^n. If $\Lambda(y) \succeq \Lambda(z)$ then $\sigma(y) \leq \sigma(z)$.*

Proof. Note that $\Lambda(y) \succeq \Lambda(z)$ if and only if the Young diagram D_y of $\Lambda(y)$ can be obtained from the Young diagram D_z of $\Lambda(z)$ by a sequence of box moves in D_z, only moving boxes upwards (from lower rows to higher rows). Thus, without loss of generality, we may assume that $\Lambda(z) = (\ldots, a, \ldots, b, \ldots)$ and that $\Lambda(y) = (\ldots, a+1, \ldots, b-1, \ldots)$. Due to Proposition 3.2.6, above, we

may further assume that $sh(y) = (\ldots, a+1, b-1, \ldots)$ and $sh(z) = \Lambda(z) = (\ldots, a, b, \ldots)$; that is, we move a box from one row in D_z to the row directly above it in D_y. ($sh(y)$ is not necessarily a partition, however).

We now calculate $n \cdot (\sigma(z)^2 - \sigma(y)^2)$ in a similar fashion to the proof of Proposition 2. Let i be the index such that $sh(z)_i = b$ and let $m = \sum_{k=1}^{i-1} sh(z)_k$.

In this case, in the computation of $n\sigma(y)^2$, we have that there are

$$\begin{array}{lll} b-1 & \text{terms equal to} & (m + \frac{b-1+1}{2} - \mu)^2 = (m + \frac{b}{2} - \mu)^2. \\ a+1 & \text{terms equal to} & (m + b - 1 + \frac{a+1+1}{2} - \mu)^2 = (m + b + \frac{a}{2} - \mu)^2 \end{array}$$

and in the computation of $n\sigma(z)^2$, we have that there are

$$\begin{array}{lll} b & \text{terms equal to} & (m + \frac{b+1}{2} - \mu)^2 \\ a & \text{terms equal to} & (m + b + \frac{a+1}{2} - \mu)^2 \end{array}$$

As before, these are the only terms that differ in the computations of $n\sigma(y)^2$ and $n\sigma(z)^2$. A computer algebra system can thus be used to verify that

$$\begin{aligned} n\sigma(z)^2 - n\sigma(y)^2 =& b(m + \tfrac{b+1}{2} - \mu)^2 + a(m + b + \tfrac{a+1}{2} - \mu)^2 \\ & - (b-1)(m + \tfrac{b}{2} - \mu)^2 - (a+1)(m + b + \tfrac{a}{2} - \mu)^2 \\ =& \tfrac{1}{4}(a + a^2 + b - b^2) \\ \geq& 0 \end{aligned}$$

where the final inequality above is due to the fact that $a \geq b > 0$ (by the definition of $\Lambda(z)$). This means that $\sigma(z) \geq \sigma(y)$. □

3.2.5 Some further combinatorial 'statistics' on RS^n

Now that we have a basic understanding of how the standard deviation behaves on rank space, we establish some further ways of considering the points in RS^n. The quantities that we define here will be useful in our discussions in Section 3.3.

Proposition 3.2.8. *Let $\lambda = sh(z)$ and let $\mu = \mu(z) = \frac{n+1}{2}$ for a point $z \in RS^n$*

(i) If $\sum_{i=1}^{k} \lambda_i < \mu$, then $\sum_{i=1}^{k-1} \lambda_i + \frac{\lambda_k+1}{2} < \mu$. That is, if you have seen fewer than μ boxes in λ, then the rank of the elements in the row you are looking at is less than μ.

(ii) If $\sum_{i=1}^{k} \lambda_i \geq \mu$ and $\sum_{i=1}^{k-1} \lambda_i + \frac{\lambda_k+1}{2} \leq \mu$ then $\sum_{i=1}^{k} \lambda_i + \frac{\lambda_{k+1}+1}{2} > \mu$. That is, if you have seen more than μ boxes but the rank of the row you are looking at is less than μ, then the rank

of the next row is larger than μ.

Proof. (i) follows from the fact that $\frac{t+1}{2} \leq t$ as long as $t \geq 1$ and $\lambda_i \geq 1$ for all i. In (ii), we have that $\sum_{i=1}^{k} \lambda_i + \frac{\lambda_{k+1}+1}{2} > \sum_{i=1}^{k} \lambda_i \geq \mu$. □

Proposition 3.2.8 gives us the following picture: when examining the rows of $\lambda = sh(z)$ from the top to the bottom, the ranks of the elements in the rows will be smaller than μ (i.e. $z_i - \mu$ will be negative for all of the boxes in these rows) until you have counted more total boxes than μ. We call the row in which your total box count meets or exceeds μ the *transition row*. The boxes in the transition row can have a rank less than μ, greater than μ, or equal to μ. Then the ranks of all the boxes in all the rows below the transition will be larger than μ (so $x_i - \mu$ is positive for all of these boxes).

Let z_i be an element of z where i is a label in the transition row of a labeled Young diagram D_z for z. We say that λ is *balanced* if $z_i = \mu$. We say that λ is *unbalanced above* (respectively *below*) if $z_i > \mu$ (respectively $z_i < \mu$). We say that λ is *unbalanced* if it is either unbalanced above or unbalanced below.

Let λ_j be the part of λ that corresponds to the transition row. Let

$$B(\lambda) = \begin{cases} \sum_{i=1}^{j} \lambda_i & : \lambda \text{ unbalanced below} \\ \sum_{i=1}^{j-1} \lambda_i & : \text{otherwise} \end{cases}, \; A(\lambda) = \begin{cases} \sum_{i=j}^{n} \lambda_i & : \lambda \text{ unbalanced above} \\ \sum_{i=j+1}^{n} \lambda_i & : \text{otherwise} \end{cases}. \tag{3.17}$$

That is, $B(\lambda)$ is the number of elements of z with ranks below μ and $A(\lambda)$ is the number of elements of z with ranks above μ. Let $k(\lambda) = \min\{A(\lambda), B(\lambda)\}$ and define the *unbalancing factor* $f(\lambda) = n - 2k(\lambda)$. Note that, if λ is unbalanced, then $f(\lambda) = |B(\lambda) - A(\lambda)|$ - it is the number of extra boxes on the unbalanced side of λ. A balanced shape λ can also have $f(\lambda) \neq 0$: in this case, $f(\lambda)$ corresponds to the number of boxes that have ranks equal to μ and $k(\lambda)$ corresponds to the number of boxes below μ (which is the same as the number of boxes above μ in this case). Note that by definition $0 \leq k(\lambda) \leq \frac{n}{2}$. The following lemma tells us that the length of the transition row is an upper bound for the unbalancing factor.

Lemma 3.2.9. *Given $z \in RS^n$, let $\lambda = sh(z)$, and assume that λ_j is part of λ that corresponds to the transition row. Then $\lambda_j \geq f(\lambda)$. If $\lambda_j = f(\lambda)$, then λ is balanced.*

Proof. First assume that λ is unbalanced below; thus there are $k(\lambda) + f(\lambda)$ boxes below the mean and $k(\lambda)$ boxes above the mean. We know that $\sum_{i=1}^{j-1} \lambda_i + \frac{\lambda_j+1}{2} < \frac{n+1}{2}$ since λ is unbalanced below.

Note that

$$\begin{aligned}\sum_{i=1}^{j-1}\lambda_i &= k(\lambda) + f(\lambda) - \lambda_j \\ &= \frac{n + f(\lambda)}{2} - \lambda_j\end{aligned}$$

using the definition of $f(\lambda)$ to write $k(\lambda) = \frac{n-f(\lambda)}{2}$. Thus, we obtain that

$$\begin{aligned} & \tfrac{n+f(\lambda)}{2} - \lambda_j + \tfrac{\lambda_j+1}{2} < \tfrac{n+1}{2} \\ \Rightarrow \quad & n + f(\lambda) - 2\lambda_j + \lambda_j + 1 < n+1 \\ \Rightarrow \quad & f(\lambda) < \lambda_j \end{aligned} \tag{3.18}$$

as desired. That is, if λ is unbalanced below, then $\lambda_j > f(\lambda)$.

Now assume that λ is balanced, which means that $\sum_{i=1}^{j-1}\lambda_i + \frac{\lambda_j+1}{2} = \frac{n+1}{2}$. In this case, we get that $\sum_{i=1}^{j-1}\lambda_i = k(\lambda) = \frac{n-f(\lambda)}{2}$. Then, following similar calculations to those in Equation (3.18), we will obtain that $f(\lambda) = \lambda_j$.

Finally, if λ is unbalanced above, then $\sum_{i=1}^{j-1}\lambda_i + \frac{\lambda_j+1}{2} > \frac{n+1}{2}$ and $\sum_{i=1}^{j-1} = k(\lambda) = \frac{n-f(\lambda)}{2}$; similar calculations to those in Equation (3.18) show that $f(\lambda) < \lambda_j$ again in this case. □

The concept of the transition row as well as the quantities k and f are heavily leaned upon in the proofs of bounds on the rank correlation in the following section.

3.3 Understanding the features selected by RANKCORR by establishing bounds on distances in rank space

We desire for RANKCORR to select good biological markers; as discussed in Section 1.1.1, however, there is no concrete biological definition of a marker gene. Thus, we would like to characterize the types of markers that are selected by the RANKCORR algorithm. Previously, in Section 2.3.2.3, we showed that RANKCORR will select the features that show the highest rank correlation (according to Spearman's ρ) with the cluster indicator vector $\tau \in \{\pm 1\}^n$. The rank correlation is often described as measuring the degree of monotonicity between two variable. This is not completely accurate, however, and we explore this relationship in this section. We also establish some lower bounds on the rank correlation distance that was introduced in 3.2.3 when dealing with sparse vectors in the context of feature selection. These bounds reveal the ideal features according to the RANKCORR algorithm and could give researchers an idea as to the quality of the (non-ideal) markers that they select.

Consider a data set $\{x_i \colon i \in [n]\}$ that is separated into two clusters indexed by $A \subset [n]$ and

$B \subset [n]$ ($A \cap B = \emptyset$, $A \cup B = [n]$), we create a vector of cluster indices τ (an *cluster indicator vector*) with $\tau_i = 1$ if $i \in A$ and $\tau_i = -1$ if $i \in B$. Then, given a set of points $S \subset RS^n$, our goal in feature selection with RANKCORR is to find the point in S closest to $\Phi(\tau)$ where $\tau \in \{\pm 1\}^n$. The following proposition shows us that we can work with the original indicator τ when working with the rank correlation - we do not need to work with $\Phi(\tau)$. This allows for us to simplify the later analysis.

Proposition 3.3.1. *Fix a vector of cluster labels $\tau \in \{\pm 1\}^n$. There is a constant c such that the rank correlation between τ and any vector $z \in RS^n$ is equal to $c \cdot \dfrac{\langle \tau, \overline{z} \rangle}{\sigma(z)}$.*

Proof. Let $A = \{i \colon \tau_i = +1\}$ and $B = \{i \colon \tau_i = -1\}$. Note that there are two numbers a and b in $\frac{1}{2}\mathbb{Z}$ such that $\overline{\Phi(\tau)}_i = a$ if $\tau_i = +1$ (i.e. if $i \in A$) and $\overline{\Phi(\tau)}_i = -b$ if $\tau_i = -1$ (we are assuming that both a and b are greater than 0). We can relate a and b in the following manner:

$$\begin{aligned} \sum_{i=1}^{n} \overline{\Phi(\tau)} = \sum_{i \in A} a - \sum_{i \in B} b = 0 \\ \Rightarrow |A|a = |B|b \\ \Rightarrow \quad a = \frac{|B|}{|A|} b \end{aligned}$$

In a similar fashion, we have that

$$\begin{aligned} \sum_{i=1}^{n} (z_i - \overline{z}) = \sum_{i \in A} (z_i - \overline{z}) + \sum_{i \in B} (z_i - \overline{z}) = 0 \\ \Rightarrow \qquad \sum_{i \in A} (z_i - \overline{z}) = -\sum_{i \in B} (z_i - \overline{z}) \end{aligned}$$

so that

$$\langle \tau, \overline{z} \rangle = \sum_{i \in A} (z_i - \overline{z}) - \sum_{i \in B} (z_i - \overline{z}) = 2 \sum_{i \in A} (z_i - \overline{z}) \tag{3.19}$$

Then we have that the rank correlation between τ and z is given by

$$\begin{aligned} \frac{\sum_{i=1}^{n} \overline{\Phi(\tau)}_i (z_i - \overline{z})}{\sigma(\Phi(\tau))\sigma(z)} &= \frac{1}{\sigma(\Phi(\tau))\sigma(z)} \left(\sum_{i \in A} a(z_i - \overline{z}) - \sum_{i \in B} b(z_i - \overline{z}) \right) \\ &= \frac{1}{\sigma(\Phi(\tau))\sigma(z)} \left(\frac{|B|}{|A|} b \sum_{i \in A} (z_i - \overline{z}) - b \sum_{i \in B} (z_i - \overline{z}) \right) \\ &= \frac{1}{\sigma(\Phi(\tau))\sigma(z)} \left(\frac{|B|}{|A|} b \sum_{i \in A} (z_i - \overline{z}) + b \sum_{i \in A} (z_i - \overline{z}) \right) \end{aligned}$$

$$= \frac{1}{\sigma(\Phi(\tau))\sigma(z)} \left(\frac{|B| + |A|}{|A|} b \sum_{i \in A} (z_i - \overline{z}) \right)$$

and we finally see that

$$\frac{\langle \tau, \overline{z} \rangle}{\sigma(z)} = \frac{2\sigma(\Phi(\tau))|A|}{nb} \rho(z, \tau). \tag{3.20}$$

Note that the expression $\frac{2\sigma(\Phi(\tau))|A|}{nb}$ does not depend on z. Thus, we have the desired result. □

In the proof above, we heavily rely on the fact that $\Lambda(\tau)$ has exactly two parts (by assumption). If $\Lambda(\tau)$ had more than two parts, then the result would not hold. Nonetheless, in case or binary feature selection, we obtain the following picture: the points in standardized rank space lie on the intersection of S^{n-1} and the plane normal to $(1, \ldots, 1)$; this is actually a copy of S^{n-2}. The effect of dotting with τ is to rotate this space with an isometry. Thus, the rank correlation between a feature and τ can be obtained by adding the coordinates of a point in rotated standardized rank space.

In order to establish a lower bound on the rank correlation distance introduced in Section 3.2.3 in the context of feature selection with RANKCORR, the above proposition allows us to start by focusing on sums of the form $\sum_{i=1}^{n} \tau_i (z_i - \mu)$ where $\tau_i \in \{\pm 1\}$ for all i, $z \in RS^n$ is a point in rank space and $\mu = \mu(z) = \frac{n+1}{2}$. This restriction will further reveal that rank correlation does not treat all monotonic relationships in the same way. Consider a point $z \in RS^n$ and an vector $\tau \in \{\pm 1\}^n$, and let $\lambda = sh(z)$. The optimal monotonic relationship between z and τ (i.e. $z_i \leq z_j$ if and only if $\tau_i \leq \tau_j$) will have the result that all values in the dot product are positive; thus, we will first consider $\sum_{i=1}^{n} |z_i - \mu|$. This is the largest rank correlation (corresponding to the smallest distance) that could possibly be obtained between a coordinate z and a vector $\tau \in \{\pm 1\}^n$.

Proposition 3.3.2. *Given a point x in rank space, let $k = k(sh(x))$. We have that $\sum_{i=1}^{n} |x_i - \mu| = k \cdot (n - k)$.*

Proof. First note that $\sum_{i=1}^{n} x_i - \mu = -n\mu + \sum_{i=1}^{n} x_i = -\sum_{i=1}^{n} x_i + \sum_{i=1}^{n} x_i = 0$. This means that the terms of x that are below μ exactly cancel out with the terms about μ in $\sum_{i=1}^{n} x_i - \mu$. In addition, this means that $\sum_{i=1}^{n} |x_i - \mu| = 2 \sum_{i:\, x_i > \mu} x_i - \mu$.

Now, given a labeled Young diagram D corresponding to a point $z \in RS^n$, let $\lambda = sh(z)$, let λ_j correspond to the transition row, and assume that $i, k < j$ with $i \neq k$. Construct a new labeled young diagram D' from D by moving the box $D_{i,m}$ to $D_{k,\ell}$, where $m \leq \lambda_i$ and $\ell \in [\lambda_j + 1]$. Let $z' \in RS^n$ correspond to D'. Since the box shift from D to D' does not change the transition row, we have that $\sum_{i:\, z_i > \mu} z_i - \mu = \sum_{i:\, z'_i > \mu} z_i - \mu$. Thus, by the discussion in the preceding paragraph, we see that $\sum_{i=1}^{n} |z_i - \mu| = \sum_{i=1}^{n} |z'_i - \mu|$. That is, as long as we don't change the boxes that are

above the mean, we can move any of the boxes below the mean around arbitrarily without changing the sum $\sum_{i=1}^{n} |z_i - \mu|$. By avoiding moving the boxes in the transition row, we ensure that the boxes that are above the mean don't change. Similar corresponding facts are true about the boxes above the mean. This means that, in order to understand the possible values of $\sum_{i=1}^{n} |z_i - \mu|$, we will use Lemma 3.2.9 to tell us more about the transition row in λ.

Next, assume that $\lambda = sh(z)$ is unbalanced below and let λ_j be the transition row of λ. We know that $\lambda_j > f(\lambda)$, so let $\lambda_j = f(\lambda) + b$ (Lemma 3.2.9). We will consider the effect of moving one box from λ_j to λ_i, where $i < j$ (or creating a new row above λ_j and reindexing if needed). Towards this end, let D be a labeled Young diagram for z and let D' be obtained from D by setting $D'_i = D_i$ for all $1 \leq i < j$ and $D'_{i+1} = D_i$ for all $j \leq i \leq n$. Then let D'_j be one box with $D'_{j,1} = D_{j,m}$ for some $1 \leq m \leq \lambda_j$. Finally, remove one box $D'_{j+1,m}$, so that D_j has one more box in it than D'_{j+1}. We now have that the ranks of the boxes in D'_{j+1} are given by $\sum_{i=1}^{j-1} \lambda_i + 1 + \frac{\lambda_j}{2}$. We would like to know when this shift does not change the sum $\sum_{i=1}^{n} |z_i - \mu|$; as discussed above, this sum will not change if $\sum_{i=1}^{j-1} \lambda_i + 1 + \frac{\lambda_j}{2} \leq \frac{n+1}{2}$ (since in this case the boxes below the transition row have not been changed at all). This implies that $b \geq 1$, as the following calculation shows:

$$\begin{array}{rrcll} & \sum_{i=1}^{j-1} \lambda_i + 1 + \frac{\lambda_j}{2} & \leq & \frac{n+1}{2} & \\ \Rightarrow & \frac{n+f(\lambda)}{2} - \lambda_j + 1 + \frac{\lambda_j}{2} & \leq & \frac{n+1}{2} & \text{(see proof of Lemma 3.2.9)} \\ \Rightarrow & n + f(\lambda) + 2 - \lambda_j & \leq & n+1 & \\ \Rightarrow & f(\lambda) + 1 - f(\lambda) - b & \leq & 0 & (\lambda_j = f(\lambda) + b) \\ \Rightarrow & 1 & \leq & b & \end{array} \tag{3.21}$$

Thus, we may remove b boxes from the transition row of D, creating a sequence of b labeled Young diagrams $D_1, D_2, \ldots, D_b$ with corresponding points in rank space $z^1, \ldots, z^b$ such that $\sum_{i=1}^{n} |z_i - \mu| = \sum_{i=1}^{n} |z_i^j - \mu|$ for all $1 \leq i \leq b$. In addition, we have that the transition row of $\rho = sh(D_b)$ has length $\rho_{j+b} = f(\lambda)$. According to Lemma 3.2.9, this means that ρ is a balanced shape.

To summarize: if $\lambda = sh(z)$ is unbalanced below, then there is another balanced shape $\rho = sh(z^b)$ with $f(\lambda) = f(\rho)$ (so that $k(\lambda) = k(\rho)$) and $\sum_{i=1}^{n} |z_i - \mu| = \sum_{i=1}^{n} |z_i^b - \mu|$. A similar argument shows that the same is true if $\lambda = sh(z)$ is unbalanced above.

Thus, all that remains is to calculate $\sum_{i=1}^{n} |x_i - \mu|$ when $x \in RS^n$ is such that $\lambda = sh(x)$ is

balanced with $k(\lambda) = k$. In this case, we know that there are k boxes below the mean. Thus,

$$\begin{aligned}\sum_{i=1}^{n} |x_i - \mu| &= 2\sum_{i=1}^{k} \mu - i \\ &= 2\left(k\mu - \frac{k(k+1)}{2}\right) \\ &= k(n+1-(k+1)) \\ &= k(n-k)\end{aligned}$$

which completes the proof of the desired result. □

Note that, for a fixed $\tau \in \{\pm 1\}^n$, Proposition 3.3.2 shows that two elements $z, w \in RS^n$ can be perfectly monotonically related to τ (e.g $z_i \leq z_j$ exactly when $\tau_i \leq \tau_j$; similar for w) but the rank correlation between z and τ will be different from the rank correlation between w and τ. Specifically, this situation will occur when $k(sh(z)) = k(sh(w))$ and $\sigma(z) \neq \sigma(w)$ (for example when $\Lambda(z) \neq \Lambda(w)$ - move some boxes around on the unbalanced side, far away from the transition row). Moreover, this result implies that the rank correlation will be larger for monotonic relationships when z has many constant values; it is best to have as many large elements of $sh(z)$ as possible, since this will decrease $\sigma(z)$ (see Proposition 3.2.7). We can also use this result to obtain the following bounds.

Corollary 3.3.3. *Let $\mu = \frac{n+1}{2}$ and $\tau \in \{\pm 1\}^n$. Given $z \in RS^n$, let $\lambda = sh(z)$. Then we have that $\sum_{i=1}^{n} \tau_i(z_i - \mu) \leq k(\lambda) \cdot (n - k(\lambda))$.*

Proof. Note that $\sum_{i=1}^{n} \tau_i(z_i - \mu) \leq \sum_{i=1}^{n} |z_i - \mu| = k(\lambda) \cdot (n - k(\lambda))$ by Proposition 3.3.2. □

It is also possible to show the reverse bound: that is, any of these simple dot products are also bounded above by $k(\lambda)(n - k(\lambda))$, where $\lambda = sh(\Phi(\tau))$. Although the actual result is not the most useful, the proof introduces techniques that we will use in proofs of more interesting results.

Proposition 3.3.4. *Let $\mu = \frac{n+1}{2}$ and $\tau \in \{\pm 1\}^n$. Let $\lambda = sh(\Phi(\tau))$. Given $z \in RS^n$, let $\overline{z}$ be the centered version of z ($\overline{z}_i = z_i - \mu$). Then we have that $\langle \tau, \overline{z} \rangle \leq k(\lambda) \cdot (n - k(\lambda))$.*

Proof. Let $\rho = sh(z)$. First note that, if $k(\rho) = k(\lambda)$, then

$$\langle \overline{z}, \tau \rangle \leq k(\rho)(n - k(\rho)) = k(\lambda)(n - k(\lambda)) \tag{3.22}$$

by Corollary 3.3.3.

Now suppose the $k(\rho) \leq k(\lambda)$. In this case, also applying Corollary 3.3.3, we see that

$$\langle \overline{z}, \tau \rangle \leq \|\overline{z}\|_1 = k(\rho)(n - k(\rho)) \leq k(\lambda)(n - k(\lambda)) \tag{3.23}$$

where we also use the fact that $k(\lambda) \leq \frac{n}{2}$ by the definition of $k(\lambda)$.

It only remains to consider the case that $k(\rho) > k(\lambda)$. Without loss of generality, we will assume that λ is unbalanced below; that is, that the number of entries of τ equal to -1 is larger than the number of entries equal to $+1$. (Note that τ must be unbalanced: if $k(\lambda) = \frac{n}{2}$ then we cannot have that $k(\rho) > k(\lambda)$).

Now note that, in order to maximize $\langle \tau, \bar{z} \rangle$, we must have that the largest entries of z correspond to the entries of τ that are $+1$. That is, letting $P(\tau) = \{i \in [n] \colon \tau_i = +1\}$ and $M(z) = \{z_i \colon i \in [n] \colon \tau_i = -1\}$, we may assume that $z_i > z_j$ for all $i \in P(\tau)$ and $j \in M(\tau)$.

Otherwise, if there is $i \in P(\tau)$ and $j \in M(\tau)$ with $z_i < z_j$, then define z' with $z'_k = z_k$ for all k except that $z'_i = z_j$ and $z'_j = z_i$. Note that $z' \in RS^n$. Then $\langle \overline{z'}, \tau \rangle - \langle \overline{z}, \tau \rangle = (z_j - \mu - z_i + \mu) - (z_i - \mu - z_j + \mu) = 2(z_j - z_i) > 0$ by the assumption that $z_i < z_j$.

If there is an $i \in P(\tau)$ and $j \in M(\tau)$ with $z_i = z_j$, then let $P_i(z) = \{\ell \in P(\tau) \colon z_\ell = z_i\}$ and $M_i(z) = \{\ell \in M(\tau) \colon z_\ell = z_i\}$. Consider the element $z' \in RS^n$ with $z'_\ell = z_\ell$ if $\ell \notin E_i(z)$. If $\ell \in M_i(z)$, then $z'_\ell = z_i - \frac{1}{2}|P_i(z)|$ and if $\ell \in P_i(Z)$ then $z'_\ell = z_i + \frac{1}{2}|M_i(z)|$. Note that $z' \in RS^n$; we have broken the ties in the row of $sh(z)$ containing the element z_i but have not changed any of the other ranks. Now we have that $\langle \overline{z'}, \tau \rangle - \langle \overline{z}, \tau \rangle = |P_i(z)||M_i(z)| > 0$.

Now we can consider two cases: either z is unbalanced above or z is unbalanced below. If z is unbalanced above, then $k(\rho)$ is the number of entries of z that are above the mean. By the preceding discussion, every entry of z that is below the mean corresponds to a -1 in τ; that is, every negative entry of $\bar{z}$ contributes a positive amount to $\langle \bar{z}, \tau \rangle$ Now, considering the entries of z that are larger than μ, let $A^+(z) = \{\ell \colon z_\ell > \mu, \tau_\ell = +1\}$ and $A^-(z) = \{\ell \colon z_\ell > \mu, \tau_\ell = -1\}$. For i in $A^+(z)$ we have that the entries $\tau_i(z_i - \mu) = |z_i - \mu|$, and for the entries $j \in A^-(z)$ we have that $\tau_j(z_j - \mu) = -|z_i - \mu|$. Thus, $\langle \bar{z}, \tau \rangle = \|\bar{z}\|_1 - 2\sum_{j \in A^-(z)} |z_i - \mu|$.

To compute $\sum_{j \in A^-(z)} |z_i - \mu|$, note that for $i \in A^+(z)$ and $j \in A^-(z)$, we have that $z_i > z_j$. Note also that $|A^-(z)| = k(\rho) - k(\lambda)$. This means that

$$\begin{aligned}
\sum_{i \in A^-(z)} |z_i - \mu| &= \sum_{i=1}^{|A^-(z)|} n - k(\rho) + i - \mu \\
&= |A^-(z)|(n - k(\rho)) + \frac{|A^-(z)|(|A^-(z)| + 1)}{2} - |A^-(z)|\mu \\
&= |A^-(z)| \left(n - k(\rho) + \frac{(|A^-(z)| - n)}{2} \right).
\end{aligned} \tag{3.24}$$

Thus we have that

$$\begin{aligned}\langle \bar{z}, \tau \rangle &= k(\rho)(n - k(\rho)) - |A^-(z)| \left(2n - 2k(\rho) + |A^-(z)| - n\right) \\ &= k(\rho)(n - k(\rho)) - (k(\rho) - k(\lambda))\,(n - k(\rho) - k(\lambda)) \\ &= k(\lambda)(n - k(\lambda))\end{aligned}$$

as desired.

If z is unbalanced above, we may follow logic similar to that above to find that $\langle \bar{z}, \tau \rangle = \|z\|_1 - 2\sum_{j \in A^-(z)} |z_i - \mu|$ with $|A^-(z)| = n - k(\rho) - k(\lambda)$. Moreover,

$$\begin{aligned}\sum_{i \in A^-(z)} |z_i - \mu| &= \sum_{i=1}^{|A^-(z)|} k(\rho) + i - \mu \\ &= |A^-(z)| \left(k(\rho) + \frac{(|A^-(z)| - n)}{2}\right)\end{aligned}$$

similar to above. This means that

$$\begin{aligned}\langle \bar{z}, \tau \rangle &= k(\rho)(n - k(\rho)) - |A^-(z)| \left(2k(\rho) + |A^-(z)| - n\right) \\ &= k(\rho)(n - k(\rho)) - (n - k(\rho) - k(\lambda))(k(\rho) - k(\lambda)) \\ &= k(\lambda)(n - k(\lambda)).\end{aligned}$$

again, as desired. If z is balanced, then either of the proofs for z unbalanced above or unbalanced below will still follow. □

Now, to fully bound the rank correlation, we must also understand how the standard deviation of z behaves (i.e. we need to understand the denominator of $\frac{\langle \tau, \bar{z} \rangle}{\sigma(z)}$ from Proposition 3.3.1). The following two results provide us with the extra information needed for a general upper bound on the rank correlation when using sparse data in the context of feature selection.

Proposition 3.3.5. *Fix a value k with $1 \leq k \leq \frac{n}{2}$. Then $\rho = (n - k, k) \in \arg\min_{\lambda:\, k(\lambda) = k} \sigma(\lambda)$.*

Proof. Consider λ such that $k(\lambda) = k$. Note that, if $\rho_1 \geq \lambda_1$, then $\rho \succeq \lambda$ (since $\rho_1 + \rho_2 = n = \sum_{i=1}^n \lambda_i$). On the other hand, it cannot be that $\lambda_1 > \rho_1$: if $\lambda_1 = n - k + j$ for some $j > 0$, then $k(\lambda) = k - j \neq k$ (note that $n - k + j > \frac{n}{2}$). Thus, if $k(\lambda) = k$, we have that $\rho \succeq \lambda$ and thus (following Proposition 3.2.7) $\sigma(\rho) \leq \sigma(\lambda)$. □

Lemma 3.3.6. *Consider a point $z \in RS^n$ such that $\lambda(z) = (n - k, k)$ (recall that $k \leq n/2$). Then, we have that $\sigma(\bar{z}) = \frac{1}{2}\sqrt{k(\lambda)(n - k(\lambda)}$*

Proof. Let $k = k(\lambda)$. We may calculate

$$\begin{aligned}\sigma(\lambda) &= \sqrt{\frac{(n-k)\left((n-k+1)/2-(n+1)/2\right)^2 + k(n-k+(k+1)/2-(n+1)/2)^2}{n}} \\ &= \sqrt{\frac{(n-k)(-k/2)^2 + k((n-k)/2)^2}{n}} \\ &= \frac{1}{2}\sqrt{\frac{k(n-k)(k+n-k)}{n}} \\ &= \frac{1}{2}\sqrt{k(n-k)}.\end{aligned}$$

□

We would now like to apply these results to data x that are sparse. In particular, let $K_{n,s} = \{x \in \mathbb{R}^n \colon \|x\|_0 \leq s\}$. Note that when $s < \frac{n+1}{2}$, we have that $k(\lambda(\Phi(x))) \leq s$ for $x \in K_{n,s}$ (since the row in $\lambda(\Phi(x))$ corresponding to the entries of x that are equal to 0 will be the transition row). The following proposition uses the framework established so far in this section to provide an upper bound on the rank correlation between a sparse vector x and a vector $\tau \in \{\pm 1\}^n$. That is, for a cluster indicator vector τ, Proposition 3.3.7 tells us how close a sparse vector of gene counts x can appear under the rank correlation distance that is used by RANKCORR to select markers. It may be possible to use these bounds to determine the quality of the markers selected (when the calculated rank correlation scores are close to saturating the bounds).

Proposition 3.3.7. *Fix a dimension n and assume that $s < \frac{n+1}{2}$.*

1. *Fix $x \in K_{n,s}$. Then $\max_{\tau \in \{\pm 1\}^n} \left\langle \overline{\Phi(x)}, \tau \right\rangle \leq s(n-s)$.*

2. *Fix $\tau \in \{\pm 1\}^n$, and let a be the number of entries of τ that are equal to +1. Then,*

$$\max_{x \in K_{n,s}} \frac{\left\langle \overline{\Phi(x)}, \tau \right\rangle}{\sigma(\Phi(x))} = \begin{cases} 2\sqrt{a(n-a)} & : \quad a \leq s \text{ or } a \geq n-s \\ \frac{2s(n-a)}{\sqrt{s(n-s)}} & : \quad s < a \leq \frac{n}{2} \\ \frac{2sa}{\sqrt{s(n-s)}} & : \quad \frac{n}{2} \leq a < n-s \end{cases}. \tag{3.25}$$

Proof. **1.** This statement follows from Proposition 3.3.2 and Corollary 3.3.3: if $x \in K_{n,s}$, then $k(\lambda(\Phi(x))) \leq s$ since $s < \frac{n+1}{2}$.

2. There are three cases to consider. In the first case, $a \leq s$. Define $w \in K_{n,s}$ with $\Phi(\tau) = \Phi(w)$ by $w_i = 0$ if $\tau_i = -1$ and $w_i = 1$ if $\tau_i = 1$. Following the previous discussions, we know that

$\Phi(w) \in \arg\max_{z \in RS^n} \frac{\langle \overline{z}, \tau \rangle}{\sigma(z)}$; in this case, we obtain the maximum in $K_{n,s}$. Applying Lemma 3.3.6, we calculate that $\frac{\langle \overline{\Phi(w)}, \tau \rangle}{\sigma(\Phi(w))} = 2\sqrt{a(n-a)}$ since $a = k(\lambda(\tau))$.

In the second case, $a \geq n - s$, which means that $s \geq n - a$. Thus, we may define $w \in K_{n,s}$ with $\Phi(\tau) = \Phi(w)$ by $w_i = -1$ if $\tau_i = -1$ and $w_i = 0$ if $\tau_i = 0$. Note that $\|w\|_0 = n - a \leq s$. Thus, again we have that $\frac{\langle \overline{\Phi(w)}, \tau \rangle}{\sigma(\Phi(w))} = 2\sqrt{a(n-a)}$ is the maximum possible value.

In the third case, $s < a < n - s$. In this case, we will be able to maximize $\langle \tau, \overline{\Phi(x)} \rangle$ and minimize $\sigma(\Phi(x))$ independently.

First, note that $\sigma(\Phi(x))$ only depends on $sh(\Phi(x))$ (Proposition 3.2.6). Specifically, consider $x \in K_{n,s}$ with $\|x\|_0 = s_x \leq s$. Then, let $\lambda = \lambda(\Phi(x))$ and let $\rho = (n - s_x, s_x)$. Note that $\lambda_1 \leq n - s_x$ since $s < \frac{n+1}{2}$ (and therefore $\frac{n}{2} \geq s \geq s_x$). Thus, we have that $\rho \succeq \lambda$, which means that $\sigma(\Phi(x)) \geq \sigma(\rho) = \frac{1}{2}\sqrt{s_x(n - s_x)}$ (Proposition 3.3.5 and Lemma 3.3.6). In addition, by defining x such that $x_i = 1$ for $1 \leq i \leq s_x$ and $x_i = 0$ for $i > s_x$, we see that $\sigma(\Phi(x)) = \frac{1}{2}\sqrt{s_x(n - s_x)}$; that is, we are able to reach this bound. This establishes the minimum of $\sigma(\Phi(x))$.

Now, we will maximize $\langle \tau, \overline{\Phi(x)} \rangle$. This essentially comes down to computation. As in the proof of Proposition 3.3.4, we again observe that we must have that the largest entries of x correspond to the entries of τ that are $+1$. That is, letting $P(\tau) = \{i \in [n] \colon \tau_i = +1\}$ and $M(\tau) = \{i \in [n] \colon \tau_i = -1\}$, we may assume that $x_i \geq x_j$ for all $i \in P(\tau)$ and $j \in M(\tau)$. Otherwise, if there is $i \in P(\tau)$ and $j \in M(\tau)$ with $x_i < x_j$, then define x' with $x'_k = x_k$ for all k except that $x'_i = x_j$ and $x'_j = x_i$. Then $\langle \overline{\Phi(x')}, \tau \rangle - \langle \overline{\Phi(x)}, \tau \rangle = 2(\Phi(x)_j - \Phi(x)_i) > 0$ by the assumption that $x_i < x_j$.

Now consider $x \in K_{n,s}$ and let $s_x = \|x\|_0$; we have that $s_x \leq s < a$. Following the preceding discussion, let $B(x) = \{i \colon x_i < 0\}$ and $P(x) = \{i \colon x_i > 0\}$. Since $a > s$ and $|P(x)| \leq s_x$, we have that $P(x) \subset P(\tau)$; in a similar fashion, since $|B(x)| \leq s_x$ and $a < n - s$, we have that $B(x) \subset M(\tau)$. Define $b_x = |B(x)|$ and $p_x = |P(x)|$. Note that $b_x + p_x = s_x$ and that $k(sh(\Phi(x))) = \max\{b_x, p_x\}$. Since $s_x \leq s < \frac{n+1}{2}$ we have that $\Lambda(\Phi(x))_1 = n - s_x$ corresponds to the transition row of $\Phi(x)$.

First let us assume that $b_x \geq p_x$ (so that $k(sh(\Phi(x))) = b_x$). In this case, we have that $\Phi(x)$ is unbalanced above; that is, the ranks of the entries in the transition row are larger than $\mu = \frac{n+1}{2}$.

Thus, letting $A^-(x) = \{i\colon \tau_i = -1 \text{ and } x_i = 0\}$, we see that

$$\begin{aligned}
\langle \overline{\Phi(x)}, \tau \rangle &= b_x(n - b_x) - 2 \sum_{i \in A^-(x)} |\Phi(x_i) - \mu| \\
&= b_x(n - b_x) - 2 \sum_{i \in A^-(x)} b_x + \frac{n - s_x + 1}{2} - \frac{n+1}{2} \\
&= b_x(n - b_x) - |A^-(x)|(2b_x - s_x) \\
&= b_x(n - b_x) - (n - a - b_x)(2b_x - s_x) \qquad (3.26)
\end{aligned}$$

where the last line is due to the fact that there are $a - p_x$ indices i with x_i in the transition row (that is, with $x_i = 0$) that correspond to positive entries of τ (that is, with $\tau_i = +1$). Since the transition row has length $n - s_x$, this means that $A^-(X) = n - s_x - (a - p_x) = n - a - (b_x + p_x) + p_x = n - a - b_x$.

Note that Equation 3.26 is a quadratic function in b_x with a positive leading coefficient. Thus, the maximum value of this function will occur at an extreme point. Since $\frac{s_x}{2} \leq b_x \leq s_x$, the values at the extreme points are given by

$$\begin{array}{rcl}
s_x(n - s_x) - (n - a - s_x)(2s_x - s_x) & = & s_x a \\
\frac{s_x}{2}\left(n - \frac{s_x}{2}\right) - \left(n - a - \frac{s_x}{2}\right)(s_x - s_x) & = & \frac{s_x}{2}\left(n - \frac{s_x}{2}\right)
\end{array} \qquad (3.27)$$

That is, when $b_x \geq p_x$, we have that $\langle \overline{\Phi(x)}, \tau \rangle \leq \max\{as_x, \frac{s_x}{2}(n - \frac{s_x}{2})\}$.

Now consider the case that $b_x < p_x$ so that $k(sh(\Phi(x))) = p_x$. Following a similar analysis to the above, we will see that

$$\langle \overline{\Phi(x)}, \tau \rangle = p_x(n - p_x) - (a - p_x)(2p_x - s_x) \qquad (3.28)$$

(where we use the facts that $\Phi(x)$ is unbalanced below and $|\{i\colon x_i = 0 \text{ and } \tau_i = +1\}| = a - p_x$). Equation 3.28 is also quadratic in p_x with a positive leading coefficient. Thus, the maximum value will occur when $p_x = s_x$ or $p_x = \frac{s_x}{2}$. In this case, we obtain the local maximum values of :

$$\begin{array}{rcl}
s_x(n - s_x) - (a - s_x)(2s_x - s_x) & = & s_x(n - a) \\
\frac{s_x}{2}\left(n - \frac{s_x}{2}\right) - \left(a - \frac{s_x}{2}\right)(s_x - s_x) & = & \frac{s_x}{2}\left(n - \frac{s_x}{2}\right).
\end{array} \qquad (3.29)$$

Thus, overall, we obtain that

$$\langle \overline{\Phi(x)}, \tau \rangle \leq \max\{s_x a, s_x(n - a), \tfrac{s_x}{2}(n - \tfrac{s_x}{2})\} \qquad (3.30)$$

for any x with $\|x\|_0 = s_x \leq s \leq \frac{n}{2}$ when $s < a < n - s$.

Now consider the case that $s < a \leq \frac{n}{2}$. In this case, we will have that $n - a \geq \frac{n}{2}$ and thus we

see that $s_x a \leq s_x \frac{n}{2} \leq s_x(n-a)$. Moreover, we have that $n - a \geq \frac{n}{2} > \frac{n}{2} - \frac{s_x}{4} = \frac{1}{2}(n - \frac{s_x}{2})$ and thus (multiplying by s_x) we obtain that $s_x(n-a) > \frac{s_x}{2}(n - \frac{s_x}{2})$ as well. That is, when $s < a \leq \frac{n}{2}$, we have that

$$s_x(n-a) = \max_{y \in \mathbb{R}^n : \, \|y\|_0 = s_x} \langle \overline{\Phi(y)}, \tau \rangle. \tag{3.31}$$

There are many vectors y which obtain this maximum value; in particular, this maximum value occurs when $\Phi(y)$ has shape $(n - s_x, s_x)$ (e.g. y has $n - s_x$ entries equal to 0 and s_x entries equal to 1). Note also that $s_x(n-a)$ is an increasing function of s_x.

Finally, consider the case that $\frac{n}{2} \leq a < n - s$. In this case, we obtain that $s_x a \geq s_x \frac{n}{2} \geq s_x(n-a)$. Similar to the above, we also see that $s_x a \geq s_x \frac{n}{2} > s_x \frac{n}{2} - \frac{s^2}{4} = \frac{s}{2}(n - \frac{s}{2})$. Thus, for $\frac{n}{2} \leq a < n - s$, we get that

$$s_x a = \max_{y \in \mathbb{R}^n : \, \|y\|_0 = s_x} \langle \overline{\Phi(y)}, \tau \rangle. \tag{3.32}$$

This maximum is obtained with $sh(\Phi(y)) = (s_x, n - s_x)$ (i.e. when $b_x = s_x$).

In summary, for x with $\|x\|_0 = s_x \leq s$, we have currently shown that

- $\sigma(\Phi(x)) \leq \frac{1}{2}\sqrt{s_x(n-s_x)}$. This minimum occurs for any x with $\lambda(\Phi(x)) = (n - s_x, s_x)$.
- When $s < a \leq \frac{n}{2}$, we get that $\langle \overline{\Phi(x)}, \tau \rangle \leq s_x(n-a)$. This bound is saturated when $sh(\Phi(x)) = (n - s_x, s_x)$.
- When $\frac{n}{2} \leq a < n - s$, we get that $\langle \overline{\Phi(x)}, \tau \rangle \leq s_x a$. This bound is saturated when $sh(\Phi(x)) = (s_x, n - s_x)$.

Note that the points x which maximize $\langle \overline{\Phi(x)}, \tau \rangle$ in both conditions on a also minimize $\sigma(\Phi(x))$. Thus, we see that

$$\max_{y : \, \|x\|_0 = s_x} \frac{\langle \overline{\Phi(x)}, \tau \rangle}{\sigma(\Phi(x))} = \begin{cases} 2\frac{s_x(n-a)}{\sqrt{s_x(n-s_x)}} & : \quad s < a \leq \frac{n}{2} \\ 2\frac{s_x a}{\sqrt{s_x(n-s_x)}} & : \quad \frac{n}{2} \leq a < n - s \end{cases}. \tag{3.33}$$

And these are increasing functions of s_x (since $s_x / \sqrt{s_x(n-s_x)}$ is an increasing function of s_x, which can be obtained by examining the derivative of that function). Thus, we obtained the desired result. □

In scRNA-seq data, the vectors of gene counts x are always positive. Thus, Propositions 3.3.4 and 3.3.7.2 give us the following description of the algorithm RANKCORR: if the columns of the data matrix represent all points in RS^n, RANKCORR will select any feature that is constant on the cluster of interest and constant (with a different value) outside of the cluster of interest. On the other hand, if the features are very sparse (there are less nonzero entries than the number of cells in

the cluster of interest, i.e. $s < a \leq \frac{n}{2}$), then RANKCORR will choose the least sparse feature that exhibits a constant expression level, and all nonzero expression occurs in cells within the cluster. Moreover, there should be no expression outside of the cluster (this fact appears in the proof of Proposition 3.3.7 near (3.31)). Here we are making the assumption that the number of cells in any cluster is less than $\frac{n}{2}$; in order to separate a cluster of size larger than $\frac{n}{2}$ from all of the others, we could instead separate all of the other points from the cluster (so we could use $\tau_i = +1$ if cell i not in the cluster) to obtain the same markers. Thus, for a fixed cluster indicator vector τ, Proposition 3.3.7.2 reveals the optimal vector x that will be chosen by the RANKCORR method.

This ideal vector is a good approximation to the idea of a biological marker - it certainly would be useful for separating the cluster from all of the other cells. Due to fact that RANKCORR is constructed using the rank correlation distance studied here, RANKCORR explicitly prefers genes that show a constant nonzero expression level, however. That is, a gene that shows constant expression within the cluster would be preferred over a gene that shows varying expression levels within the cluster, even if both genes are not expressed outside of the cluster. The potential biological benefits or drawbacks of this preference are left for future work.

Note that Proposition 3.3.7.1 and Proposition 3.3.2 have an alternate geometric interpretation. They say that, given z in RS^n, the corner $\tau \in \{\pm 1\}^n$ of the n-dimensional cube that is nearest to $\bar{z}$ (in terms of cosine distance) is the one in the same orthant of $\mathbb{R}^n$ as $\bar{z}$. If z has any entries equal to μ, than this is not as well-defined of a concept, since $\bar{z}$ will be on the boundary between multiple orthants. For example, if $x = (-1, 0, 0, 0, 1) \in \mathbb{R}^5$, then $\overline{\Phi(x)}/\sigma(\Phi(x))$ is the same distance from both $\tau^1 = (-1, -1, -1, -1, 1)$ and $\tau^2 = (-1, -1, -1, 1, 1)$ since it sits inside both of the orthants containing τ_1 and τ_2. One can compute that $\overline{\Phi(x)}/\sigma(\Phi(x))$ is closer to $\overline{\Phi(\tau_1)}/\sigma(\Phi(\tau_1))$ than to $\overline{\Phi(\tau_2)}/\sigma(\Phi(\tau_2))$, however[2], since $\overline{\Phi(\tau_1)}$ is closer to the boundary between orthants than $\overline{\Phi(\tau_2)}$.

A proposition similar to Proposition 3.3.7.2 for the full value of rank correlation (without using the simplifications of Proposition 3.3.1) is provided below. In this proposition, we restrict x to $K^+_{n,s} = \{x \in \mathbb{R}^n \colon \|x\|_0 \leq s \text{ and } x_i > 0 \text{ for all } i\}$. This is for the ease of analysis, but it is also a reasonable restriction. These algorithms are developed for sparse integer counts data, which will always be positive in scRNA-seq (and in many other areas). It is possible to remove this assumption with more detailed computational analysis, similar to the proof of Proposition 3.3.7. We provide Proposition 3.3.8 here as a reference - the rank correlation scores calculated by the RANKCORR algorithm can be directly compared to these values to potentially provide information about the quality of markers that are selected.

Proposition 3.3.8. *Fix a dimension n and assume that $s < \frac{n+1}{2}$. In addition, fix $\tau \in \{\pm 1\}^n$ and let*

[2]This is not contrary to Proposition 3.3.1, since τ is assumed fixed in Proposition 3.3.1.

a be the number of entries in τ that are equal to $+1$. Then we have that

$$\max_{x\in K^+_{n,s}} \frac{\left\langle \overline{\Phi(x)}, \overline{\Phi(\tau)} \right\rangle}{\sigma(\Phi(x))\sigma(\Phi(\tau))} = \begin{cases} n & s \geq a \\ \frac{n-a}{2\sigma(\Phi(\tau))} \cdot \frac{sn}{\sqrt{s(n-s)}} & s \leq a \end{cases} \tag{3.34}$$

Proof. First assume that $s \geq a$. In this case, we have that there is a vector $x \in K^+_{n,s}$ with $\Phi(x) = \Phi(\tau)$; for example, x could be such that $x_i = 0$ if $\tau_i = -1$ and $x_i = 1$ if $\tau_i = 1$. Then, we see that

$$\frac{\left\langle \overline{\Phi(x)}, \overline{\Phi(\tau)} \right\rangle}{\sigma(\Phi(x))\sigma(\Phi(\tau))} = \frac{\left\langle \overline{\Phi(\tau)}, \overline{\Phi(\tau)} \right\rangle}{\sigma(\Phi(\tau))\sigma(\Phi(\tau))} = n \tag{3.35}$$

since $\overline{\Phi(\tau)}/\sigma(\Phi(\tau))$ has length $\sqrt{n}$ by the definition of $\sigma(\cdot)$. The fact that $\overline{\Phi(\tau)}/\sigma(\Phi(\tau))$ has length $\sqrt{n}$ also implies that $\frac{\left\langle \overline{\Phi(x)}, \overline{\Phi(\tau)} \right\rangle}{\sigma(\Phi(x))\sigma(\Phi(\tau))} \leq n$ for all x by the Cauchy-Schwarz inequality, since $\overline{\Phi(x)}/\sigma(\Phi(x))$ also has length $\sqrt{n}$.

Next assume that $a > s$. As in the proof of Proposition 3.3.7, we will consider the numerator and denominator separately. In the numerator, we again observe that we must have that the largest entries of x correspond to the entries of τ that are $+1$. That is, letting $P(\tau) = \{i \in [n] \colon \tau_i = +1\}$ and $M(\tau) = \{i \in [n] \colon \tau_i = -1\}$, we may assume that $x_i \geq x_j$ for all $i \in P(\tau)$ and $j \in M(\tau)$. Otherwise, if there is $i \in P(\tau)$ and $j \in M(\tau)$ with $x_i < x_j$, then define x' with $x'_k = x_k$ for all k except that $x'_i = x_j$ and $x'_j = x_i$. Then $\langle \overline{\Phi(x')}, \overline{\Phi(\tau)} \rangle - \langle \overline{\Phi(x)}, \overline{\Phi(\tau)} \rangle = (\Phi(\tau)_i - \Phi(\tau)_j)(\Phi(x)_j - \Phi(x)_i) > 0$ by the assumption that $x_i < x_j$ and that $i \in P(\tau)$ while $j \in M(\tau)$.

The preceding paragraph implies that $\{i \colon x_i \neq 0\} \subset P(\tau)$ since $x \in K^+_{n,s}$. Moreover, note that $\overline{\Phi(\tau)}_i = \frac{n-a+1}{2} - \frac{n+1}{2} = \frac{-a}{2}$ if $i \in M(\tau)$ while $\overline{\Phi(\tau)}_j = n - a + \frac{a+1}{2} - \frac{n+1}{2} = \frac{n-a}{2}$ if $j \in P(\tau)$. Likewise, if $x_i = 0$ we have that $\overline{\Phi(x)}_i = \frac{-s}{2}$. Thus we can compute:

$$\begin{aligned} \langle \overline{\Phi(x)}, \overline{\Phi(\tau)} \rangle &= (n-a)\frac{-a}{2} \cdot \frac{-s}{2} + (a-s)\frac{n-a}{2} \cdot \frac{-s}{2} + \frac{n-a}{2} \sum_{i=1}^{s} \left(n - s + i - \frac{n+1}{2} \right) \\ &= \frac{n-a}{4} \left(s^2 + 2s(n-s) - s(n+1) + s(s+1) \right) \\ &= \frac{n-a}{4}(s^2 + s(n-s)) = \frac{n-a}{4} sn. \end{aligned}$$

Note that this does not depend at all on the shape of $\Phi(x)$; this is the advantage of assuming that $x \in K^+_{n,s}$. On the other hand, $\sigma(\Phi(x))$ depends only on the shape of $\Phi(x)$. In addition, we have that $k(sh(\Phi(x))) = s$ for all $x \in K^+_{n,s}$. Therefore, following Proposition 3.3.5 and Lemma 3.3.6, we see that $\sigma(\Phi(x)) \geq \frac{1}{2}\sqrt{s(n-s)}$ for all $x \in K^+_{n,s}$ and that $\sigma(\Phi(x)) = \frac{1}{2}\sqrt{s(n-s)}$ when $sh(\Phi(x)) = (n-s, s)$. This means that the desired maximum value occurs for any $x \in K^+_{n,s}$ with $sh(\Phi(x)) = (n-s, s)$. □

It is apparent the maxima for Proposition 3.3.7.2 when $s \leq a \leq \frac{n}{2}$ and Proposition 3.3.8 when $s \leq a$ occur for the same shape of $\Phi(x)$. After calculating the constant c that appears in Proposition 3.3.1, it is comforting to note that the value of the bound in Proposition 3.3.8 can be obtained by multiplying the bound from Proposition 3.3.7.2 by c. Thus, it appears that choosing x from the full set $K_{n,s}$ (rather than the restricted set $K_{n,s}^{+}$ as in Proposition 3.3.8) results in a tighter bound only for when $\frac{n}{2} \leq a \leq n - s$. As previously argued, this is not an interesting regime, since we can consider $-\tau$ instead of τ to obtain the same markers in this case.

3.4 Further research directions: distribution of rank correlation

Although the bounds presented in Section 3.3 above can be used to get an idea of the quality of the markers selected by RANKCORR, it would be ideal to obtain further information about the distributions of rank correlation for vectors at different sparsity levels. This would allow for a more formal statistical certification of the selected markers. Of course, it is possible to use a permutation test to obtain a p-value for the significance of each marker separately. Nonetheless, it would still be beneficial to determine how likely it is that the rank correlation score of a given marker would occur (assuming that all possible ranks of vectors are equally likely to occur, for example). Compared to a permutation test (which reveals the likelihood of obtaining a fixed rank correlation score given a fixed vector of counts), this would give a better idea as to how close a feature selected by RANKCORR is to the ideal feature developed in the previous section (for example, we could determine what percentage of ranked vectors result in larger rank correlation scores).

It is known that the distribution of Spearman's ρ is asymptotically normal (as $n \to \infty$) in the case that ties are not allowed [HP36, KKS39]; however, little work has been done to understand the distribution of ρ when ties are allowed. The work [OL16] derives a population normality result[3], which is not quite the result that we desire here.

It may be possible to determine something about the distribution of rank correlation between sparse vectors and cluster indicator vectors τ using the framework developed in this chapter. For example, as mentioned in Section 3.3, the rank correlation between a sparse vector x and a vector $\tau \in \{\pm 1\}^n$ is (proportional to) the sum of the elements of a point in rotated standardized rank space; that is, it is the sum of the coordinates of a point on a sphere. In addition to this, the distribution of the sum of the coordinates of points chosen uniformly at random from the surface of a sphere is asymptotically normal [Spr07]. Unfortunately, it is possible to show[4] that the points of

[3]That is, given two random variables X and Y and samples $x \in \mathbb{R}^n$ and $y \in \mathbb{R}^n$ of X and Y respectively, let $\hat{\rho}$ denote the computed empirical correlation between x and y and let ρ denote the true correlation between X and Y. The result of [OL16] states that the distribution of $\hat{\rho} - \rho$ approaches a normal distribution as $n \to \infty$.

[4]Consider the point $z_1 \in RS^n$ that has shape $sh(z_1) = (n-1, 1)$ and also the point $z_2 \in RS^n$ that has shape $sh(z_2) = (n-2, 1, 1)$; z_2 is the closest point to z_1 in RS^n. Calculations show that the angle between z_1 and z_2 goes to $\pi/4$ as $n \to \infty$ (that is, $\lim_{n\to\infty} \rho(z_1, z_2) = \frac{1}{\sqrt{2}}$).

(standardized) rank space are not uniformly distributed on the n-sphere as $n \to \infty$. Nevertheless, the techniques developed in [Spr07] might be adaptable to the case of rank correlation.

Another possible avenue might be to consider central limit theorems (CLT) for dependent random variables. Given a point $z \in RS^n$, the individual ranks appear to be independent as long as a small enough subset of the coordinates of z is considered (that is, knowing only a few of the ranks does not give much information about the other ranks in z). Additionally, as a consequence of Proposition 3.3.1, the rank correlation between z and a vector $\tau \in \{\pm 1\}^n$ can be calculated (essentially) by adding up the coordinates of z. Thus, the rank correlation score for z is a sum of random variables that are close to independent.

One path that appears promising is a version of the CLT for mixing in stochastic processes: the coordinates of a point $z \in RS^n$ could potentially be considered a stationary stochastic process. When n is large, it reasonable to believe that the coordinates near the start of z and the coordinates towards the end of z are nearly independent (that is, the σ-algebra generated by the random variables Z_1, Z_2, $\dots$ Z_j and the σ-algebra generated by $Z_n, Z_{n-1}, \dots, Z_{j+k}$ could be almost independent (in a technical sense as described in [Bra16]) for some j and k). If this is true, and some other moment conditions can be established, then the random variable $S_n = \sum_{i=1}^{n} Z_n$ converges in distribution to a normal distribution, which is close to the desired result. Experiments generating points $z \in RS^n$ uniformly at random do indeed suggest that the distribution of rank correlation is normal. See [McL03, Bra16, Bra05] for more information about these definitions and ideas.

Finally, note that the bounds in Proposition 3.3.8 depend on s, the number of nonzero entries in the gene expression vector. Thus, it would be ideal to determine a different distribution for each value of s as well, in order to partially remove the effects of different sparsity levels (that is, less sparse genes are preferred by RANKCORR, as long as all nonzero expression occurs within the cluster of interest).

3.5 Discussion

In this chapter, the RANKCORR algorithm was analyzed in further detail. We proved that the SELECT algorithm (Algorithm 1) is correct, implying that RANKCORR runs as intended. Moreover, we showed that RANKCORR requires average time $\mathcal{O}(n^2)$ to select markers for a cluster of cells. Finally, we established the ideal markers that RANKCORR will select. This analysis involved delving into the properties of the rank transformation and rank space.

This concludes the section in this dissertation related to scRNA-seq; Chapter 4 is focused on algorithmic fairness. Nonetheless, the debiasing method presented in Chapter 4 is based on a method that is commonly used when processing genetic data: thus, although the major topic will change, the methods and analysis have wide applicability.

CHAPTER 4

Debiasing Representations by Removing Unwanted Variation Due to Protected Attributes

We now shift the focus of this dissertation to topics in algorithmic fairness. In practice, the use of algorithms does not remove all human bias from decision making. Machine learning (ML) methods do not intentionally produce biased or discriminatory predictive models. Instead, biases in classifiers are often introduced by training on biased or incomplete data. For example, there are fewer women than men currently in STEM professions, so a naively trained predictor would be more likely to predict a STEM profession for a male rather than a female. Indeed, Amazon recently discovered its ML-based resume screening system discriminates against women applying for technical positions [Das18].

Simply removing the protected attribute (e.g. "gender") from the data is not enough to prevent these biases from appearing[1]. This is due to the fact that it is often possible to predict the protected attribute from combinations of other features (e.g. "wears makeup") in the data. In this chapter, we account for this by treating the variation that is present in the data due to protected attributes as "unwanted" variation. After removing this unwanted variation from the data set, it should be possible to use standard machine learning methods to train fair algorithms on the debiased data.

A similar situation occurs when processing scRNA-seq data. Due to the remarkably small amount of genetic material in an individual cell, the mRNA counts collected during an scRNA-seq experiment are sensitive to environmental (and other) noise. Indeed, sequencing cells from the same sample in sequential experiments will result in data sets that cannot be directly compared - often, the variation due to these batch effects will be stronger than the underlying biological variation. One way of addressing these unwanted experimental variations was established in [JGBS15]. This work expanded on the removing unwanted variation (RUV) factor model analysis that was developed in [GbJS13] for bulk genetic analysis.

In this chapter, we use a factor model to formalize the contributions of the protected and permissible attributes to data. We then adapt the RUV method to remove the unwanted variation due to the protected attributes (and thus debias the data). We show that under certain idealized

[1] Moritz Hardt said "There is no such thing as fairness through unawareness" [Dro17].

conditions, the debiased representation is conditionally uncorrelated with the protected attributes. In other words, it satisfies a first order approximation of conditional parity [RSZ17] in these cases.

The task of debiasing features was also studied by [LJ16], but there are a few differences between their approach and the proposed one. First, their approach creates new features that satisfy unconditional rather than conditional parity. Second, their approach entails estimating the conditional distributions of the features. This is hard in general, especially if the features are high-dimensional.

We use ProPublica's COMPAS dataset and COMPAS risk recidivism scores as an example throughout the remainder of this chapter. More information can be found from [ALMK16] and the Practitioners Guide to COMPAS [2]. Much has been written questioning the fairness of these scores with respect to race, with concerns about the disparate false negative and false positive rates between African-Americans and Caucasians.

Notation We denote matrices by uppercase Greek or Latin characters and vectors by lowercase characters. A (single) subscript on a matrix indexes its rows (unless otherwise stated). We denote the span of the rows of the matrix M by $\mathcal{R}(M)$.

Consider a random matrix $X \in \mathbb{R}^{n \times d}$ that is distributed according to a *matrix-variate normal* distribution with mean $M \in \mathbb{R}^{n \times d}$, row covariance $\Sigma_r \in \mathbb{R}^{n \times n}$, and column covariance $\Sigma_c \in \mathbb{R}^{d \times d}$. We denote this situation by $X \sim \mathsf{MN}(M, \Sigma_r, \Sigma_c)$.

4.1 Adjusting for protected attributes

Consider the widely adopted model for matrix-variate data:

$$\underset{(n\times d)}{Y} = \underset{(n\times k)(k\times d)}{X}\underset{}{A^T} + \underset{(n\times l)(l\times d)}{Z}\underset{}{B^T} + \underset{(n\times d)}{E}. \tag{4.1}$$

In the context of genetics, Y is a matrix of gene expression levels: the columns of Y correspond to genes, and its rows correspond to samples. Here X and Z represent unwanted and wanted variation in the expression levels respectively. For example, the rows of X may encode the lab at which the sample was processed, and A may represent the effects on the measured expression levels of processing at different labs.

On the other hand, in the context of fairness, the rows of Y are representations of individuals, the rows of X (resp. Z) are protected attributes (resp. permissible attributes) of the samples, and the rows of E are error terms that represent idiosyncratic variation in the representations. In this paper, we assume $k, l \ll d$, but this low-dimensionality is not required for the use of the algorithms presented here.

[2] http://www.northpointeinc.com/files/technical_documents/FieldGuide2_081412.pdf

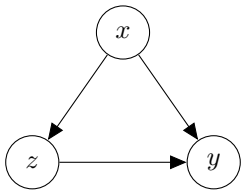


Figure 4.1: The model (4.1) and (4.2).

In practice, Y is usually observed, X is sometimes observed, and Z is unobserved. For example, in [BCZ+16], the representations are embeddings of words in the vocabulary, and the protected attribute is the gender bias of (the embeddings of) words. The rows of Z are unobserved factor loadings that represent the "good" variation in the word embeddings. In analogy to the framework in [FSV16], the rows of Z are points in the construct space while the rows of Y are points in the observed space. We emphasize that like the construct space, Z is unobserved.

We highlight that we permit non-trivial correlation between the protected and permissible attributes. In other words, we allow the protected attribute to *confound* the relationship between the permissible attribute and the representation (see Figure 4.1 for a graphical representation of the dependencies between the rows of Y, X, and Z). This complicates the task of debiasing the representations. To keep things simple, we assume the regression of Z on X is linear:

$$\underset{(n\times l)}{Z} = \underset{(n\times k)(k\times l)}{X}\,\underset{}{\Gamma^T} + \underset{(n\times l)}{W}. \tag{4.2}$$

The rows of W are error terms that represent variation in the permissible attributes not attributed to variation in the protected attributes. We specify the distributions of X, E, and W, in Sections 4.1.2 and 4.1.3.

Our goal is to obtain debiased representations Y_{db} such that the debiased representations are uncorrelated with the protected attributes conditioned on the permissible attributes:

$$\mathsf{Cov}\big[[Y_{\mathsf{db}}]_i, x_i \mid z_i\big] = 0. \tag{4.3}$$

This is implied by *conditional parity*: $[Y_{\mathsf{db}}]_i \perp x_i \mid z_i$, and we consider (4.3) as a first-order approximation of conditional parity. An ideal debiased representation is the variation in the representation attributed to the permissible attributes ZB^T, but this is typically unobservable in practice.

COMPAS example. Under this model, each row of Y corresponds to a person's data for recidivism prediction. In our experiments, this includes age, juvenile and adult felony and misdemeanor

counts, and whether the offense was a misdemeanor or felony. In this case, X is a vector, and each component indicates the person's race. We restrict to Caucasians and African-Americans in our experiments. A person's true propensity to choose to commit a crime, Z, is unknown.

4.1.1 Homogeneous subgroups

In the context of genetics, RUV methods rely on the knowledge of a set of control genes: genes whose variation in their expression levels are solely attributed to variation in Z; for example, genes unaffected by the treatments. Formally, a set of controls is a set of indices $\mathcal{I} \subset [d]$ such that $B_{\mathcal{I}} = 0$. Thus

$$Y_{\mathcal{I}} = XA_{\mathcal{I}}^T + E_{\mathcal{I}}, \tag{4.4}$$

where $Y_{\mathcal{I}}$ and $E_{\mathcal{I}}$ consist of subsets of the *columns* of Y and E, which suggests estimating $A_{\mathcal{I}}^T$ by linear regression. This is precisely the "transpose" of the method that we advocate here.

Specifically, the proposed approach relies crucially on knowledge of homogeneous subgroups: groups of samples in which the variation in their representations is mostly attributed to variation in their protected attributes. These homogeneous subgroups correspond to the control genes.

Formally, we presume knowledge of sets of indices $\mathcal{I}_1, \ldots, \mathcal{I}_G \subset [n]$ such that $H_g Z_{\mathcal{I}_g} \approx 0$, where $H_g = I_{|\mathcal{I}_g|} - \frac{1}{|\mathcal{I}_g|}\mathbf{1}_{|\mathcal{I}_g|}\mathbf{1}_{|\mathcal{I}_g|}^T$ is the centering matrix, for any $g \in [G]$. In other words,

$$H_g Y_{\mathcal{I}_g} \approx H_g X_{\mathcal{I}_g} A^T + H_g E.$$

Ideally, $H_g Z_{\mathcal{I}_g}$ exactly vanishes. This ideal situation arises when the samples in the g-th group share permissible attributes: $Z_{\mathcal{I}_g} = \mathbf{1}_{|\mathcal{I}_g|} z_g^T$ for some $z_g \in \mathbb{R}^l$.

Intuitively, homogeneous subgroups are groups of samples in which we expect a machine learning algorithm that only discriminates by the permissible attributes to treat similarly. For example, in [BCZ+16], the homogeneous subgroups are pairs of words that differ only in their gender bias: (*waiter*, *waitress*), (*king*, *queen*).

COMPAS example. In Section 4.2, we take the homogeneous groups to be people who either did not recidivate within two years or people who did recidivate within two years and were charged with the same degree of felony or misdemeanor. Although Z is unknown, we expect subjects who go on to commit similar crimes or those who do not recidivate to have similar Z regardless of race. We emphasize that the homogeneous subgroups are not defined by having similar attributes in Y.

4.1.2 Adjustment when the protected attribute is unobserved

We now show that the approach proposed by [BCZ+16] produces debiased representations that satisfy (4.3). When the protected attributes are not observed, it is generally not possible to attribute variation in the representations to variation in the protected and permissible attributes. Thus, [BCZ+16] settle on removing the variation in the representations in the subspace spanned by the protected attributes. In other words, we debias the representations by projecting them onto the orthogonal complement of $\mathcal{R}(A^T)$.

Formally, let $Q_g \in \mathbb{R}^{|\mathcal{I}_g| \times (|\mathcal{I}_g|-1)}$ be a subunitary matrix such that $\mathcal{R}(Q_g^T)$ coincides with $\mathcal{R}(H_g)$. Then we have that

$$Q_g^T Y_{\mathcal{I}_g} \approx Q_g^T X_{\mathcal{I}_g} A^T + Q_g^T E,$$

which, along with the assumptions (4.5), implies $\mathsf{Cov}\left[Q_g^T Y_{\mathcal{I}_g}\right] \approx \Sigma_E + AA^T$. This is a factor model, which allows us to consistently estimate A by factor analysis under mild conditions. We impose classical sufficient conditions for identifiability of A [AR56]:

1. Let A_{-i} be the $(d-1) \times k$ submatrix of A consisting of all but the i-th row of A. For any $i \in [n]$, there are two disjoint submatrices of A_{-i} of rank k.

2. $A^T \Sigma_E^{-1} A$ is diagonal, and the diagonal entries are distinct, positive, and arranged in decreasing order.

We remark that the additional assumptions we imposed in this section are a tad stronger than necessary: the assumptions actually imply identifiability of A, but we only wish to estimate $\mathcal{R}(A^T)$.

In light of the preceding development, here is a natural approach to adjustment when the protected attribute is unobserved:

1. estimate A by factor analysis.

2. debias Y by projection onto $\mathcal{R}(A^T)^\perp$: $Y_{\mathsf{db}} = Y(I - P_{\mathcal{R}(A^T)})$,

which gives

$$\mathsf{Cov}[[Y_{\mathsf{db}}]_i, x_i | z_i] = \mathsf{Cov}[P_{\mathcal{R}(A^T)^\perp}(Bz_i + e_i) | z_i] = 0.$$

Note that when the columns of B are in $\mathcal{R}(A^T)$ the debiased representations will be non-informative because they only contain noise.

4.1.3 Adjustment if the protected attribute is observed

If the protected attribute is observed, it is straightforward to debias the representations. The main challenge here is estimating A. Once we have a good estimator $\widehat{A}$, we debias the representations by subtracting $X\widehat{A}^T$. We summarize the approach in Algorithm 9.

Algorithm 9 Adjustment if the protected attr. is observed

Input: representations $Y \in \mathbb{R}^{n\times d}$, protected attributes $X \in \mathbb{R}^{n\times k}$ and groups $\mathcal{I}_1, \ldots, \mathcal{I}_G \subset [n]$
Estimate A **by regression:**

$$\widehat{A}^T \in \arg\min\{\tfrac{1}{2}\textstyle\sum_{g=1}^{G} \|Y_g - X_g A^T\|_F^2\},$$

where $Y_g = Y_{\mathcal{I}_g} - \mathbf{1}_{|\mathcal{I}_g|}(\frac{1}{|\mathcal{I}_g|}\mathbf{1}_{|\mathcal{I}_g|}^T Y_{\mathcal{I}_g})$ and X_g is defined similarly.
Debias Y**:** subtract the variation in Y attributed to X from Y: $Y_{\mathsf{db}} = Y - X\widehat{A}^T$.

To study the properties of Algorithm 9, we impose the following assumptions on the distributions of X, E, and W:

$$\begin{aligned} X &\sim \mathsf{MN}(0, I_n, \Sigma_x), \\ E \mid (X, Z) &\sim \mathsf{MN}(0, I_n, \Sigma_\epsilon). \end{aligned} \tag{4.5}$$

Proposition 4.1.1. *Let Z_g and E_g be defined similarly as Y_g and X_g. Under conditions* (4.1)*,* (4.2)*, and* (4.5)*,*

$$\widehat{A}^T - A^T \mid (X, Z) \sim \mathsf{MN}(T\textstyle\sum_{g=1}^{G} X_g^T Z_g B^T, T, \Sigma_\epsilon) \tag{4.6}$$

where $T = (\sum_{g=1}^{G} X_g^T X_g)^\dagger$.

Proof. Let $\tilde{X} = [X_1 \ldots X_G]^T$ and $\tilde{Z}, \tilde{E}, \tilde{Y}$ be similarly defined. Then

$$\begin{aligned} \hat{A}^T &= (\tilde{X}^T\tilde{X})^\dagger \tilde{X}^T \tilde{Y} \\ &= (\tilde{X}^T\tilde{X})^\dagger \tilde{X}^T (\tilde{Z}B^T + \tilde{E}) + A^T \\ E[\hat{A}^T - A^T | (X, Z)] &= (\tilde{X}^T\tilde{X})^\dagger \tilde{X}^T \tilde{Z} B^T \\ \mathsf{Cov}_{\text{row}}[\hat{A}^T - A^T | (X, Z)] &= (\tilde{X}^T\tilde{X})^\dagger \tilde{X}^T I \tilde{X} (\tilde{X}^T\tilde{X})^\dagger \\ &= (\tilde{X}^T\tilde{X})^\dagger \\ \mathsf{Cov}_{\text{col}}[\hat{A}^T - A^T | (X, Z)] &= \Sigma_E. \quad \square \end{aligned}$$

The (conditional) bias in the OLS estimator of A depends on the similarity of the permissible attributes in homogeneous subgroups. If $Z_g = 0$ for all $g \in G$, then $\widehat{A}$ is a (conditionally) unbiased estimator of A.

Proposition 4.1.2. *Under conditions* (4.1)*,* (4.2)*, and* (4.5)*, we have*

$$\mathsf{Cov}[y_i - \widehat{A}x_i, x_i \mid z_i] = -B\mathsf{Cov}[\tilde{Z}\tilde{X}(\tilde{X}^T\tilde{X})^\dagger x_i, x_i | z_i].$$

Proof.

$$\begin{aligned}\mathsf{Cov}[[Y_{\mathsf{db}}]_i, x_i \mid z_i] &= \mathsf{Cov}[(A - \hat{A})x_i + e_i | z_i] \\ &= \mathsf{Cov}[e_i - (B\tilde{Z}^T + \tilde{E})\tilde{X}(\tilde{X}^T\tilde{X})^\dagger x_i | z_i] \\ &= -B\mathsf{Cov}[\tilde{Z}\tilde{X}(\tilde{X}^T\tilde{X})^\dagger x_i, x_i | z_i] \quad \square\end{aligned}$$

We see that if $\hat{A} = A$ or $\tilde{Z} = 0$ then the debiased y_i is uncorrelated with the protected attributes x_i.

4.2 Experiments: Debiased representations for recidivism risk scores

We empirically demonstrate the efficacy of Algorithm 9 for reducing racial bias in recidivism risk scores based on data ProPublica [3] used in their investigation of COMPAS scores. We fit our own models to the raw and debiased data. Although simple, the scores output by our model perform comparably to the proprietary COMPAS scores (see Figure 4.2 and Table 4.3).

In particular, we show that logistic regression (LR) trained on debiased data obtained from Algorithm 9 reduces the magnitude of the difference in the false positive rates (FPR) and false negative rates (FNR) between Caucasians and African-Americans (AA) compared to LR trained on raw, potentially biased data. This "fairer" outcome is achieved with a relatively small impact on the percentage of correct predictions. The variables in our LR model are discussed in Section 4.1.

We split our data into three pieces: a training set used to estimate A from Equation (4.1), and a train and test set to evaluate the performance of the learned model. Figure 4.3 shows the distribution of the probabilities of recidivism for African-Americans according to a logistic model based on the raw and debiased representations. The distribution of the probabilities from the raw representation is skewed to the right. In particular, the right tail of the distribution of probabilities from the raw representation is noticeably heavier than that from the debiased representation.

The ROC curve of the LR model trained on raw data is similar to the ROC curve of COMPAS scores validating the choice of LR as a proxy for COMPAS scores. See Figure 4.2. For the remaining discussion, we average all results over 30 splits of the data into train and test sets. The average accuracy (the percentage of correct predictions) is 65% for COMPAS. The accuracy for LR trained on raw data and debiased data is comparable, again validating our proxy and justifying the slight loss in accuracy after debiasing in pursuit of fairer outcomes. See Table 4.3.

Table 4.1 and Table 4.2 show the average FPR and FNR for the LR model before and after debiasing. The two tables differ only in the threshold used to declare someone at risk for recidivism based on his or her logistic score; we choose to examine the 50th and 80th quantiles of LR scores since Northpointe specifies that COMPAS scores above the of 50th (respectively 80th) quantile are said to indicate a "Medium" (respectively "High") risk of recidivism. In Table 4.1, we see that

[3] https://github.com/propublica/compas-analysis/blob/master/compas-scores-two-years.csv

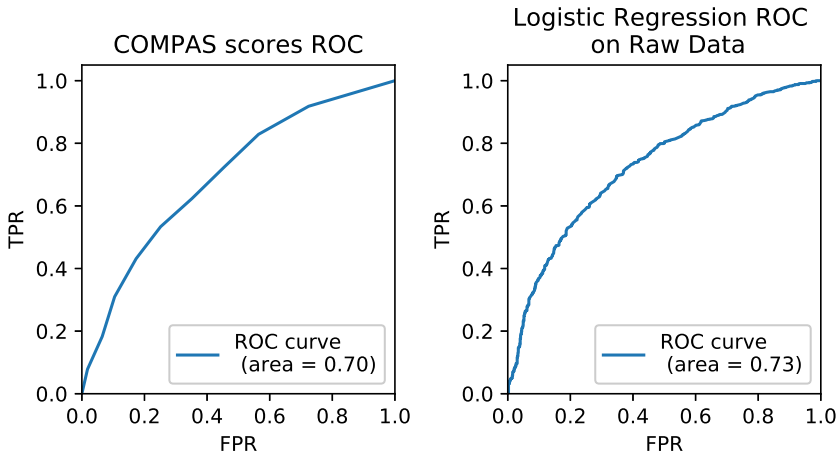


Figure 4.2

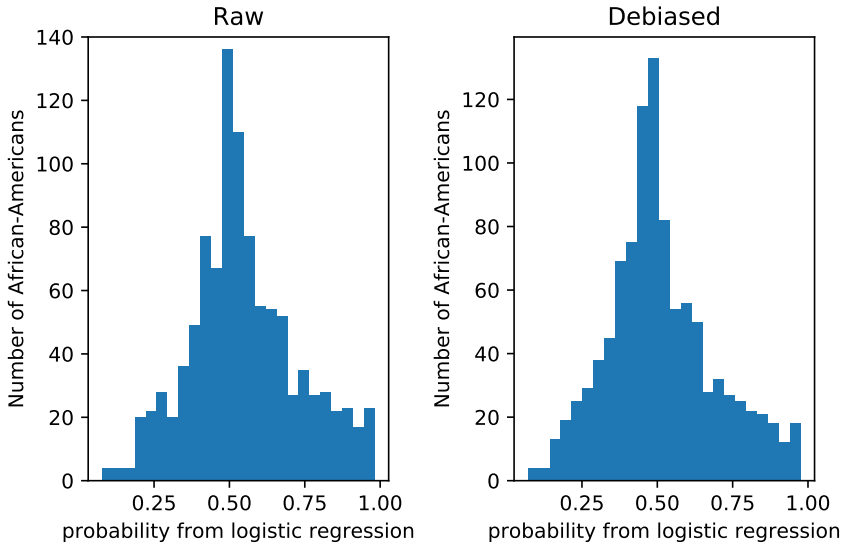


Figure 4.3: Distribution of recidivism probabilities from raw and debiased representations for African-Americans.

there is no difference in FPR after debiasing. The difference in FNR between the races goes from nearly 20% before debiasing to 4% after debiasing. In Table 4.2, we see FNR are nearly equalized, whereas the magnitude of the difference of FPR between both race groups is improved. However, now Caucasians suffer from disparate impact of FPR instead of African-Americans.

	LR raw		LR debiased	
	FPR (SE)	FNR (SE)	FPR (SE)	FNR (SE)
Population	0.08 (0.01)	0.68 (0.01)	0.9 (0.01)	0.69 (0.01)
Caucasian	0.05 (0.01)	0.81 (0.02)	0.9 (0.02)	0.72 (0.03)
AA	0.11 (0.01)	0.62 (0.01)	0.9 (0.01)	0.68 (0.02)

Table 4.1: Average proportion FPR and FNR with standard errors (SE) based on the 80th quantile of LR scores.

	LR raw		LR debiased	
	FPR (SE)	FNR (SE)	FPR (SE)	FNR (SE)
Population	0.32 (0.01)	0.32 (0.01)	0.4 (0.01)	0.34 (0.01)
Caucasian	0.22 (0.02)	0.5 (0.02)	0.42 (0.03)	0.31 (0.02)
AA	0.4 (0.02)	0.23 (0.01)	0.27 (0.02)	0.35 (0.02)

Table 4.2: Average proportion FPR and FNR with standard errors (SE) based on the 50th quantile of LR scores.

Accuracy	LR Raw (SE)	LR Debiased (SE)	COMPAS (SE)
50 quantile	0.67 (.011)	0.65 (.01)	0.65 (.008)
80 quantile	0.61 (.01)	0.60 (.01)	0.61 (.01)

Table 4.3: Proportion of correct predictions (with standard errors) by logistic regression and thresholding COMPAS scores

4.3 Summary and discussion

We study a factor model of representations that explicitly models the contributions of the protected and permissible attributes. Based on the model, we propose an approach to debias the representations. We show that under certain conditions, we can guarantee first order conditional parity for the debiased representations.

We present an experimental example examining at the COMPAS data set [ALMK16]. We find that using vanilla logistic regression produces similar accuracy results to the original COMPAS algorithm. Moreover, a logistic regression classifier trained on the COMPAS data after it has been debiased using the method introduced in this chapter has essentially the same accuracy as the vanilla

LR classifier, and it also shows significantly smaller FPR and FNR gaps between Caucasians and African-Americans. That is, debiasing the COMPAS data does not result in any lost accuracy but improves the fairness of the LR classifier.

As noted in the discussion above, in order to apply the debiasing methods presented in this chapter, it is essential to know several homogeneous subgroups of individuals within the training data. This often requires expert knowledge and deep study of the data set in question. In our work with the COMPAS data set, for example, our homogeneous subgroups (individuals who did not recidivate and individuals who commit crimes of a similar severity to their original crime) were based on our subjective interpretations of which groups of individuals we would expect to share similar discriminatory attributes (regardless of their protected attributes).

The determination of these homogeneous subgroups is thus a potentially costly task, both in terms of time and monetary resources (as it will usually require expert supervision). Therefore, although it is desirable to debias data before training a predictor, it will not always be possible to do so. In the following chapter, we explore a potential solution: a method for constructing a fair predictor that is trained on biased data.

CHAPTER 5

Individually Fair Gradient Boosting Through Robust Learning

5.1 Introduction

In the previous chapter, we introduced a way to debais data - standard machine learning (ML) models can then be trained on the debiased data to produce *fair* results. Specifically, according to Equation (4.3), the debiased representations created by the method in Chapter 4 are not correlated with the protected attributes conditioned on the permissible attributes (under certain assumptions). This statistical condition is one way to mathematically formalize a debiased representation, and it empirically results in a specific type of statistical fairness (as seen in Section 4.2). As alluded to in Section 1.2, however, much effort has been expended in defining what, precisely, it means for an ML method to be fair.

Broadly speaking, there are two definitions of algorithmic fairness in contemporary study: group fairness and individual fairness [CR18]. Group fairness requires that a small number of fixed groups are treated similarly [ZVRG15, HPS+16]. Most of the existing fairness literature is focused on group fairness methods, as is the debiasing method presented in Chapter 4. Although group fairness is amenable to statistical analysis, it suffers from two crucial drawbacks: there are definitions that are mutually incompatible [KMR16, Cho17], and there are examples of algorithms that satisfy group fairness but are blatantly unfair from individual users' perspectives [DHP+12]. The drawbacks of group fairness lead us to focus on on the less-studied individual fairness framework, which requires that similar individuals are treated similarly [DHP+12].

Despite its advantages over group fairness, there are few general approaches to enforcing individual fairness. The main barrier to broader adoption of individual fairness was a lack of consensus on which users are similar for many ML tasks, but a flurry of recent work proposes several solutions [Ilv19, WGL+19, YBS20]. Here, we assume there is a similarity metric for the ML task at hand and consider the task of individually fair gradient boosting.

Gradient boosting, especially gradient boosted decision trees (GBDT), is a popular method for tabular data problems. Indeed, several GBDT methods (such as XGBoost [CG16]), exhibit performance comparable to or superior than (fully connected) neural networks when working with

tabular data. XGBoost in particular has historically been part of a large percentage of the winning solution to problems on the coding competition website Kaggle [CG16]. Since GBDT methods are useful and ubiquitous, the creation of fair gradient boosting techniques is important for the establishment and use of fair methods in real-world applications. Unfortunately, existing approaches to enforcing individual fairness are either not suitable for training non-smooth ML models [YBS20] or perform poorly with flexible non-parametric ML models. Thus, in this chapter, we propose a method for constructing fair predictors with gradient boosting, regardless of bias in the training data. Unlike other individually-fair training methods, our approach also works with non-smooth ML models such as GBDTs.

A final note: as discussed in Section 1.2, the determination of a "fair" metric d_x on $\mathcal{X}$ is itself a difficult problem. In this work, the space $\mathcal{X}$ is assumed to be a subset of $\mathbb{R}^n$, and we approximate a fair metric d_x on $\mathcal{X}$ with a Mahalanobis measure that we learn from the training data. Essentially, we determine several directions in $\mathcal{X}$ that we deem to be protected, project onto the orthogonal complement of the span of those directions, and use the standard Euclidean distance in that projected subspace. Determining the protected directions can require a small amount of expert guidance, depending on the fair metric in question. Moreover, this is meant only as an approximation to a true fair metric d_x: it is an operational definition that nonetheless allows for the training of fair methods (as exhibited in the experiments in Section 5.5).

5.1.1 Notation

Here we define some general notation that we use in this chapter. Other important notation, defined with context in later sections, is collected here in Table 5.1 for convenience.

Given two matrices $A, B \in \mathbb{R}^{n \times m}$, we define the notation $\langle A, B \rangle = \sum_{i,j} a_{ij} b_{ij}$. Moreover, let $A \odot B$ denote the Hadamard product of A and B; that is $(A \odot B)_{ij} = A_{ij} \cdot B_{ij}$. Let $\delta_{(x,y)}$ be the Kronecker delta; that is, $\delta_{(x,y)} = 1$ if $x = y$ and $\delta_{(x,y)} = 0$ otherwise. Let $\mathbf{1}_n$ denote the vector with every entry equal to 1 in $\mathbb{R}^n$.

Consider the cost function on $\mathcal{Z}$ given by

$$c((x_1, y_1), (x_2, y_2)) = d_x^2(x_1, x_2) + \infty \cdot \mathbf{1}\{y_1 \neq y_2\} \tag{5.1}$$

Given two probability distributions P, Q on $\mathcal{Z}$, let $W(P, Q)$ denote the 1-Wasserstein distance between P and Q in relation to the cost function c. That is,

$$W(P, Q) = \inf_{\Pi} \int_{\mathcal{Z} \times \mathcal{Z}} c(z_1, z_2) d\Pi(z_1, z_2) \tag{5.2}$$

where the infimum is taken over all couplings Π between P and Q; that is, distributions Π on $\mathcal{Z} \times \mathcal{Z}$

with marginals P and Q.

Symbol	Definition
$\mathcal{X}$	space of individuals
$\mathcal{Y}$	space of outcomes
$\mathcal{Z} = \mathcal{X} \times \mathcal{Y}$	
d_x	a fair metric on $\mathcal{X}$
$\mathcal{D} = \{(x_i, y_i\}_{i=1}^n \subset \mathcal{Z}$	training data set
$\mathcal{D}_0 = \{x_1, \ldots, x_n\} \times \mathcal{Y}$	duplicated data set
P_*	training distribution on $\mathcal{Z}$ (e.g. $\mathcal{D} \sim P_*$)
$P_n = \sum_{i=1}^n \delta_{(x_i, y_i)}$	empirical distribution on $\mathcal{Z}$
$\mathcal{F}$	collection of candidate prediction functions
$\ell \colon \mathcal{F} \times \mathcal{Y} \to \mathbb{R}_+$	loss function used to promote accuracy in an ML task
$c((x_1, y_1), (x_2, y_2)) = d_x^2(x_1, x_2) + \infty \cdot \mathbf{1}_{\{y_1 \neq y_2\}}$	cost function on $\mathcal{Z}$
$W_{\mathcal{D}}(Q, P)$	1-Wasserstein distance between distributions Q on $\mathcal{D}_0$ and P on P on $\mathcal{D}$.
$L(f) = \sup_{Q \colon W_{\mathcal{D}}(Q, P_n) \leq \epsilon} \mathbb{E}_Q[\ell(f, \mathcal{Z})]$	Robust loss with perturbation budget ϵ defined in Section 5.2.1.
C	matrix of costs; $C_{i,j} = d_x^2(x_i, x_j)$ for $x_i, x_j \in \mathcal{D}$.
R	matrix of losses; $R_{i,j} = \ell(f, (x_i, y_j))$ for a candidate predictor f.
$\Gamma = \{M \mid M \in \mathbb{R}_+^{n \times n}, \langle C, M \rangle \leq \epsilon,$ $M^T \cdot \mathbf{1}_n = \frac{1}{n}\mathbf{1}_n\}$	set of couplings that marginalize to distributions satisfying $Q \colon W_{\mathcal{D}}(Q, P_n) \leq \epsilon$ (see Section 5.2.2)

Table 5.1: Notation for Chapter 5.

5.2 Enforcing individual fairness in gradient boosting

All notation introduced here is collected in Table 5.1. Let $\mathcal{X}$ be the space of individuals and $\mathcal{Y}$ be the space of possible outputs. For the purposes of this discussion, we will assume $\mathcal{Y} = \{0, 1\}$; the details presented here can be extended to other discrete sets of outcomes $\mathcal{Y}$. Define $\mathcal{Z} = \mathcal{X} \times \mathcal{Y}$. We equip $\mathcal{X}$ with an *fair metric* that quantifies the similarities between individuals in a fair way. The fair metric is typically application specific and is not usually a metric in the formal mathematical sense - we discuss the choice of d_x in our empirical studies in Section 5.5. Our task is to learn a predictor $f \colon X \to Y$ from a class of predictors $\mathcal{F}$ via gradient boosting such that $f(x_1)$ is similar to $f(x_2)$ whenever $d_x(x_1, x_2)$ is small. We accomplish this by defining a robust population loss function $L \colon \mathcal{F} \to \mathbb{R}$ that penalizes violations of the individual fairness constraints as well as poor predictive performance. We can apply functional gradient descent to the loss function L in order to obtain our desired gradient boosting technique.

5.2.1 Individually fair loss function

We begin by defining a robust population loss function $L\colon \mathcal{F} \to \mathbb{R}$ that promotes individual fairness. Let $\ell\colon \mathcal{F} \times \mathcal{Z} \to \mathbb{R}_+$ denote a (convex, differentiable) loss function that we use to measure the performance of a predictor (smaller values of ℓ indicate better performance), and let $\mathcal{D} = \{(x_i, y_i)\}_{i=1}^n \sim P_*$ be a set of training data. Intuitively, the robust loss L searches for fairness violations in a candidate predictor f by considering small perturbations (according to the fair metric d_x[1]) of the individuals that are examined during training and tallying how these perturbations affect the loss function ℓ. If x and x' are two individuals, the robust loss L effectively uses the loss function ℓ to measure the similarity between the predictions $f(x)$ and $f(x')$.

We formalize these perturbations through the framework of distributionally robust optimization (DRO; see e.g. [YBS20, SND18]) to obtain a robust loss L defined by:

$$L(f) = \sup_{P \in \mathcal{P}} \mathbb{E}_P[\ell(f, \mathcal{Z})] \tag{5.3}$$

where $\mathcal{P}$ is a set of distributions on $\mathcal{Z}$ that are *close to* the empirical distribution P_n corresponding to the data $\mathcal{D}$. We construct $\mathcal{P}$ so that the distributions $P \in \mathcal{P}$ are supported in regions of $\mathcal{Z}$ that contain individuals *close* to the individuals represented in the training data $\mathcal{D} = \{x_i, y_i\}_{i=1}^n$ (according to the fair distance d_x). To be precise, letting $B_{d_x}(r, x)$ represent the ball of d_x-radius r centered at the individual $x \in \mathcal{X}$, there is a value of ϵ such that each distribution $P \in \mathcal{P}$ has support contained in $\cup_{i=1}^n B_{d_x}(\epsilon, x_i) \times \{y_i\} \subset \mathcal{Z}$. Thus, every distribution $P \in \mathcal{P}$ represents perturbing the training individuals from their original locations (specified by the empirical distribution P_n) to nearby locations, without changing the labels of the individuals.

In addition, to allow for the construction of an arbitrary (potentially discontinuous or non-differentiable) fair classifier f via gradient boosting, we restrict the nature of the allowed perturbations encoded by the set $\mathcal{P}$. Specifically, we will only allow individuals represented in the training data $\mathcal{D}$ to be perturbed to other individuals represented in $\mathcal{D}$.

We now formalize these notions to define the set of distributions $\mathcal{P}$ that we use in our robust loss L. Let $P_n = \frac{1}{n}\sum_{i=1}^n \delta_{(x_i, y_i)}$ denote the empirical distribution on $\mathcal{D}$ and define the augmented data set $\mathcal{D}_0 = \{x_1, \ldots, x_n\} \times \mathcal{Y} \subset \mathcal{Z}$. Additionally, consider the cost function c on $\mathcal{Z} \times \mathcal{Z}$ defined by

$$c((x_1, y_1), (x_2, y_2)) = d_x^2(x_1, x_2) + \infty \cdot \mathbf{1}_{\{y_1 \neq y_2\}}. \tag{5.4}$$

Then, $c((x_1, y_1), (x_2, y_2))$ is small whenever x_1 and x_2 are comparable and $y_1 = y_2$; thus, c provides a cost for perturbing individuals without altering their labels. Finally, let $W_{\mathcal{D}}(Q, P)$ denote the

[1]Two individuals that are close in terms of d_x are not necessarily similar - they can differ in many protected ways (e.g. race, gender) that are not measured by the fair metric d_x. By allowing perturbations according to d_x during training, we are able to specifically penalize differences in predictions that arise from differences in protected directions.

1-Wasserstein distance between distributions Q on $\mathcal{D}_0$ and P on $\mathcal{D}$ according to the cost function c; that is,

$$W_{\mathcal{D}}(Q, P) = \inf_{\beta \in B(Q,P)} \int_{\mathcal{Z}\times\mathcal{Z}} c(z_1, z_2) \mathrm{d}\beta(z_1, z_2) \tag{5.5}$$

where $B(Q, P)$ denotes the set of all distributions on $\mathcal{Z} \times \mathcal{Z}$ that have Q and P as first and second marginals[2]. Given a budget ϵ, every distribution Q that satisfies $W_{\mathcal{D}}(Q, P_n) < \epsilon$ represents perturbing the training individuals from their original locations (specified by the empirical distribution P_n) to nearby training points. Thus, we ultimately define $\mathcal{P} = \{P \colon W_{\mathcal{D}}(P, P_n) \leq \epsilon\}$. This results in the robust loss function

$$L(f) = \sup_{Q \colon W_{\mathcal{D}}(Q,P_n)\leq\epsilon} \mathbb{E}_Q[\ell(f, \mathcal{Z})]. \tag{5.6}$$

In order to force L to audit for fairness, we only allow perturbations of the *individuals* x_i in the training data without allowing for those individuals to change their labels y_i. Given the individual x_i with true outcome y_i, we find a nearby individual x_i' such that $f(x_i')$ is as different as possible from y_i (as measured by ℓ). When $\ell(f(x_i'), y_i) > \ell(f(x_i), y_i)$, this indicates unfairness in the candidate predictor f: x_i and x_i' are comparable and thus should intuitively be assigned the label y_i (the true outcome for x_i), but $f(x_i')$ is further from y_i than $f(x_i)$ (as measured by ℓ)[3]. We are able to aggregate the worst of these areas given limited perturbations by taking the supremum over all of the distributions in $\mathcal{P}$ in Equation 5.3.

Note also that the ability of L to promote fairness depends on the samples in $\mathcal{D}$ and thus also on the distribution P_*. For example, suppose we would like to train a predictor is fair with respect to a specific protected attribute. Assume also that the protected attribute can take values in a (finite) set $A \subset \mathbb{R}$. Suppose that there is a point (x_i, y_i) in $\mathcal{D}$, where x_i is of type $a \in A$. In order to adequately promote fairness using the loss function L, for at least one other $a' \in A$ ($a' \neq a$), there should be a point (x_k, y_k) in $\mathcal{D}$ such that $d_x(x_i, x_k)$ is small and x_k is of type a' (otherwise, we would only consider moving x_i to individuals who are also of type a; a' will not be represented near x_i).

Although we will not provide a precise characterization of the distributions P_* that will allow for adequate fairness results, the preceding example suggests that P_* should have at least some support on all classes for any protected attribute. To obtain the theoretical results in Section 5.3, we invoke a strong assumption about P_* that probably can be relaxed in practice. The important point here is that the input data $\mathcal{D}$ directly influences the ability of the loss function L to enforce fairness.

[2] Given $\beta \in B(Q, P)$, the constraint on the marginals of β implies that the support of β is contained in $\mathcal{D}_0 \times \mathcal{D}$. Thus, the integral in (5.5) is a finite sum.

[3] We are not concerned about the case that $\ell(f, (x_i', y_i)) \leq \ell(f, (x_i, y_i))$ since we are using ℓ to quantify the performance of the candidate predictor f. In particular, we will eventually be minimizing $L(f)$ to create a useful classifier.

5.2.2 Individually fair gradient boosting

We would like to use the individually fair loss function L defined in (5.6) for gradient boosting. In order to perform the boosting step with current predictor f, we need to evaluate the derivatives

$$\begin{aligned}\frac{\partial L}{\partial f(x_i)} &= \frac{\partial}{\partial f(x_i)} \sup_{P:\, W_{\mathcal{D}}(Q,P_n)\leq\epsilon} \mathbb{E}_P[\ell(h,\mathcal{Z})] \\ &= \sum_{y=0}^{1}\sum_{k=1}^{n} \frac{\partial}{\partial f(x_i)}\ell(f,(x_k,y))P^*(\{(x_k,y)\})\end{aligned} \tag{5.7}$$

where P^* is a distribution that attains the supremum. The final equality in the above holds since the loss ℓ is assumed to be convex (see Section 5.3 for the precise assumptions on ℓ): that is, we can calculate these derivatives by first determining an optimal distribution $P^* \in \arg\max_{P:\, W_{\mathcal{D}}(P,P_n)\leq\epsilon} \mathbb{E}_P[\ell(f,\mathcal{Z})]$ and then computing the derivatives of $\mathbb{E}_{P^*}[\ell(f,\mathcal{Z})]$. Thus, all that remains for boosting is to find P^*.

Below, we derive a method for finding P^* via a linear program. Since W_0 is a discrete Wasserstein distance, if $c(z_i,z_j)=\infty$ for any $z_i \in \mathcal{D}_0$ and any $z_j \in \mathcal{D}$, then we will have that $P^*(z_i,z_j)=0$. Thus, we will only focus on the pairs $(z_i,z_j)\in\mathcal{D}_0\times\mathcal{D}$ with $c(z_i,z_j)<\infty$.

Following this logic, let $C \in \mathbb{R}^{n\times n}$ be the matrix with entries given by $C_{i,j} = c((x_i,y_k),(x_k,y_k)) = d_x^2(x_i,x_k)$. The asymmetry in the definition of C is due to the fact that we are only considering the finite costs between the elements in $\mathcal{D}$ and the elements of $\mathcal{D}_0$. We also define the class indicator vectors y^1 and y^0 by

$$y_j^1 = \begin{cases} 1 & : y_j = 1 \\ 0 & : y_j = 0 \end{cases} \qquad \text{and} \qquad y^0 = \mathbf{1}_n - y^1 \tag{5.8}$$

for all $y_j \in \mathcal{D}$. For a fixed distribution P on $\mathcal{D}_0$, let $P_{i,k} = P(\{(x_i,k)\})$ for $k\in\{0,1\}$. Then, the condition that $W_0(P,P_n)\leq\epsilon$ is implied by the existence of a matrix $\Pi\in\mathbb{R}^{n\times n}$ such that

1. $\Pi\in\Gamma$ with $\Gamma=\{M|M\in\mathbb{R}_+^{n\times n}, \langle C,M\rangle\leq\epsilon,\ M^T\cdot\mathbf{1}_n=\frac{1}{n}\mathbf{1}_n\}$.
2. $\Pi\cdot y^1=(P_{1,1},\ldots,P_{n,1})$, and $\Pi\cdot y^0=(P_{1,0},\ldots,P_{n,0})$.

Further define the matrix $R\in\mathbb{R}^{n\times n}$ by $R_{i,j}=\ell(f,(x_i,y_j))$ - this is the loss incurred if point j with label y_j is located at point i. With this setup, given the current predictor f, we can obtain a solution Π^* to the optimization as the solution to the linear program (in n^2 variables)

$$\Pi^* \in \arg\max_{\Pi\in\Gamma} \langle R,\Pi\rangle. \tag{5.9}$$

then the optimal distribution P^* on $\mathcal{D}_0$ is defined by $P^*(\{(x_i, k)\}) = (\Pi^* \cdot y^k)_i$. As outlined above, this allows us to compute the pseudo-residuals $\frac{\partial L}{\partial f(x_i)}$ for all $x_i \in \pi_1(\mathcal{D})$ and thus use the individually fair loss L for gradient boosting. An outline of this procedure is provided in Algorithm 10.

Algorithm 10 Fair gradient boosting

Input: Labeled training data $\{(x_i, y_i)\}_{i=1}^n$; class of weak learners $\mathcal{H}$; initial predictor f_0; search radius ϵ; number of steps T; sequence of step sizes $\alpha^{(t)}$; fair metric d_x on $\mathcal{X}$
Define the matrix C by $C_{i,j} \leftarrow d_x^2(x_i, x_j)$.
for $t = 0, 1, \ldots, T-1$ **do**
    Define the matrix R_t by $(R_t)_{ij} = \ell(f_t, (x_i, y_j))$
    Find $\Pi_t^* \in \arg\max_{\Pi \in \Gamma} \langle R_t, C \rangle$.
    $P_{t+1}(\{(x_i, k\}) \leftarrow (\Pi_t^* \cdot y^k)_i$
    Fit a base learner $h_t \in \mathcal{H}$ to the set of pseudo-residuals $\{\frac{\partial L}{\partial f_t(x_i)}\}_{i=1}^n$ (see (5.7)).
    Let $f_{t+1} = f_t + \alpha^{(t)} h_t$.
end for
return f_T

It is important to note that we have made no assumptions about the function h in finding the optimal transport map Π^* in equation (5.9). In particular, h can be a discontinuous function - for example, a decision tree or a sum of decision trees. This allows us to apply this fair gradient boosting algorithm to any class $\mathcal{F}$ of base classifiers.

5.2.3 Related Work

Other directions in individual fairness In an attempt to leverage the easily analyzed statistical definitions from the group fairness framework in the stronger individual fairness domain while avoiding the need for defining a fair metric, several recent works independently developed a notion of *subgroup (or subset) fairness* [HKRR17, KGZ19, KNRW17, KNRW19, CMJ+19]. These subgroup definitions of fairness require some statistical (group) fairness property to hold over a collection $\mathcal{S} \subset 2^{\mathcal{X}}$ of subsets from the universe of individuals ([KNRW17] considers both statistical parity and false positive fairness, [HKRR17] considers accuracy and calibration). In essence, these definitions consider all subsets in $2^{\mathcal{X}}$ to be protected, and attempt to apply the statistical guarantees to as many of these subsets as possible. Further, [CMJ+19] provides a framework for learning compact representations of data that can be easily adapted to be fair on a class of subsets.

There has also been recent work that re-frames the problem of individual fairness in a manner that is approachable without specific use (or complete knowledge) of a fair metric ([KRR18, GJKR18, JKN+19, KRS19]). For example, the authors of [KRR18] assume access to an oracle that can estimate a fair distance between two points in $\mathcal{X}$ and use this oracle to provide an algorithm for a predictor that will, on average, treat similar people from different subgroups in a similar

manner (a subgroup fairness result). [GJKR18] considers the case that there is an oracle that can be queried to determine when two individuals are far apart according to a fair metric (but cannot quantify the difference between the individuals). Assuming the fair metric comes from some specific families, the authors provide bounds on fair classifiers with limited violations of the metric constraints. [JKN+19] provides a similar framework: they show how to learn a predictor that minimizes classification error while subject to the fairness constraints provided by the oracle. On the other hand, [KRS19] considers two distributions: a distribution over individuals and a distribution over decisions. They redefine individual fairness to mean that statistical properties of the decisions (error, false positives, etc.) are approximately equivalent from individual to individual. Finally, [RY18] considers the case that a fair metric is known, and shows that, in general, it is still hard to learn a perfectly fair classifier with respect to the metric. They then provide a way to efficiently learn a fair classifier when the metric constraint is relaxed slightly.

Armed with a fair metric, practitioners can appeal to methods such as [XYS20, YBS20] to audit ML models for violations of individual fairness and to train individually fair ML models. These existing models are limited to differentiable classifiers and are thus not applicable to gradient tree boosting.

Learning fair metrics The problem of determining fair metrics is a necessary precursor for the method presented here. Towards this end, [Ilv19] considers the case that there is an oracle that can be queried to determine when two individuals are close together according to a fair metric (the author specifically considers queries of the form "is a closer to b or c?"). This information is then used to learn an approximation to (specifically, a contraction of) the underlying fair metric. In a similar fashion, [LGW19] operationalizes the idea of a fair metric by constructing a graph in which individuals are connected by an edge if they are deemed comparable by experts. They then use this graph to learn a representation of the data where points in connected clusters are mapped to points that are close to each other. This idea of learning fair representations of individuals has also been considered in other prior works; see Chapter 4 of this dissertation along with [ZWS+13] and the references within [LGW19]. Finally, the work [WGL+19] compares existing metric learning methods on a specific data set (that has been embellished with human annotations) to provide guidance about how to learn fair metrics.

Note that, when discussing a fair metric d, we do not necessarily require for d to be a metric in the mathematical sense of the word. For example, in [YBS20], the authors present two different ways of determining a fair Mahalanobis distance on $\mathcal{X}$ by projecting out "protected" directions in $\mathcal{X}$ and measuring Euclidean distance on the projected space. Using such a distance, if $x_1 - x_2$ points in one of the protected directions, then we will have that $d(x_1, x_2) = 0$ even if $x_1 \neq x_2$. We are simply using d to measure the dissimilarity (distance) between two points in a fair way.

Distributionally robust optimization The approach introduced here is an instance of distributionally robust optimization (DRO): $\sup_{P \in \mathcal{U}} \mathbb{E}_P [\ell(Z, \theta)]$. Here $\mathcal{U}$ is a (data-dependent) uncertainty set of probability distributions, which may be defined by moment or support constraints [CSS07, DY10, GS10], f-divergence balls [BdDW$^+$12, LZ15, MMK$^+$15, DGN16, ND16], and Wasserstein balls [SEK15, BKM16, DN16, EK15, LR18, SND18, HSNL18].

Most similar to our work is [YBS20], which uses DRO to train individually fair ML models. In particular, the method from [YBS20] essentially takes the ideas of adversarial training in [SND18] and applies them to the case of algorithmic fairness (considering perturbations with respect to a fair metric rather than, for example, ℓ_1 or ℓ_2 distances). Thus, potentially other robust machine learning methods could be adapted to the domain of fairness.

Adversarial training Our approach to fair training is also similar to adversarial training [SZS$^+$13, GSS14, MMS$^+$17], which hardens ML models against adversarial attacks by minimizing adversarial losses of the form $\sup_{u \in \mathcal{U}} \ell(z + u, \theta)$. Here $\mathcal{U}$ is a set of allowable perturbations [PMJ$^+$15, CW16, KGB16].

Two recent works [YRWC19, CZBH19] focus on the construction of robust non-parametric classifiers (including decision trees). These works have similar goals to our own; however, neither work is amenable to the use of a fair distance d_x as we have considered it here. For example, [CZBH19] requires a ball of finite radius to be bounded; this will not generally be the case for the fair distances that we consider in this chapter.

5.3 Theoretical results

In this section, we provide an analysis of the general fair gradient boosting method presented in Algorithm 10. Specifically, we show that the function L generalizes to the "true" population loss[4]. We require several assumptions to accomplish this. The following are fairly standard when analyzing methods in the ML literature (see e.g. [LR18]). Let $\mathcal{L} = \{\ell(\cdot, h) \colon h \in \mathcal{F}\}$ denote the loss class under consideration here.

Assumption 5.3.1. *The space $\mathcal{X}$ is bounded; i.e.* $\operatorname{diam}(\mathcal{X}) < \infty$.

Assumption 5.3.2. *The function ℓ is bounded:*

$$0 \leq \ell(h, z) \leq B \quad \text{for all} \quad f \in \mathcal{F} \text{ and } z \in \mathcal{Z}.$$

[4]Please refer to the published version of this work for analysis of the convergence of boosting.

Assumption 5.3.3. *The functions in the loss class $\mathcal{L}$ are ω-Lipschitz with respect to d_x:*

$$|\ell((x_1, y), h) - \ell((x_2, y), h)| \leq \omega d_x(x_1, x_2) \quad \text{for all} \quad x_1, x_2 \in \mathcal{X}, y \in \mathcal{Y} \text{ and } f \in \mathcal{F} \tag{5.10}$$

Additionally, the distribution P_* should contain information about the entire space of individuals $\mathcal{X}$ in order for the robust loss L to adequately emphasize fairness. Specifically, let $B_r(x_*) = \{x \in \mathcal{X} \colon d_x(x, x_*) < r\}$ be the ball of radius r around x_* in $\mathcal{X}$. We will add an additional assumption that there is a lower bound on the amount of weight that P_* assigns to $B_r(x) \times \mathcal{Y}$ for all $x \in \mathcal{X}$.

Assumption 5.3.4. *There are constants δ and d such that $P_*(B_r(x) \times \mathcal{Y}) \geq \delta r^d$ when $r < 1$.*

In our examples of interest, $\mathcal{X}$ is a subset of $\mathbb{R}^d$. In this case, we have that volume of the Euclidean ball of radius r is given by $c \cdot r^d$ for some constant c; this is the motivation behind the bound in Assumption 5.3.4. Moreover, in this work, we construct d_x as a Mahalanobis distance. In this context, Assumption 5.3.4 is reasonable when combined with Assumption 5.3.1 ($\operatorname{diam}(\mathcal{X}) < \infty$); Assumption 5.3.4 will be satisfied when P_* is bounded below by a value $b > 0$, for example. Overall, this assumption is close to the goal of the construction of the Pilot Parliaments Benchmark data from [BG18]: here, we require significant representation of a highly expressive family of subsets of the space $\mathcal{X}$ rather than a specific sample.

In addition to this, Assumption 5.3.4 is probably too strong. Realistically, we should only need that P_* is similar on sets that are close to each other according to d_x. We keep this framework to make the analysis simple here.

Define the "true" population loss L_r and the empirical population loss L_e by

$$L_r(f) = \sup_{P \colon W(P, P_*) \leq \epsilon} \mathbb{E}_P[\ell(f, \mathcal{Z})] \qquad \text{and} \qquad L_e(f) := \sup_{P \colon W(P, P_n) \leq \epsilon} \mathbb{E}_P\left[\ell(f, \mathcal{Z})\right] \tag{5.11}$$

respectively. Notice the only difference between L_e and the robust loss L from (5.6) is that there is less restriction on the distributions P for L_e: L_e is defined with respect to the normal 1-Wasserstein distance on $\mathcal{Z} \times \mathcal{Z}$. We can divide the proof of generalization error bounds into two parts. First, we show the gap between L and L_e vanishes in the large-sample limit.

Theorem 5.3.5. *Under assumptions 5.3.1–5.3.4, with probability at least $1 - n(1 - \delta/\sqrt{n})^n \to 1$, we have that*

$$|L_e(h) - L(h)| \leq \frac{1}{n^{1/(2d)}} \left(\omega + \frac{2L\operatorname{diam}(\mathcal{X})}{\sqrt{\epsilon}}\right) \tag{5.12}$$

Proof. It has been shown ([SND18, YBS20]) that the dual to the optimization L_e defined in (5.11) is given by

$$L_f(h) = \inf_{\lambda \geq 0} \lambda\epsilon - \frac{1}{n} \sum_{i=1}^{n} \sup_{x \in \mathcal{X}} \ell(h(x), y_i) + \lambda d_x^2(x, x_i). \tag{5.13}$$

Likewise, calculations (see Section 5.4.3) reveal that the dual to (5.6) is given by

$$L(h) = \inf_{\lambda \geq 0} \lambda\epsilon - \frac{1}{n}\sum_{i=1}^{n} \max_{x \in \pi_i(D)} \ell(h(x), y_i) + \lambda d_x^2(x, x_i). \tag{5.14}$$

where $\pi_1 \colon \mathcal{Z} \to \mathcal{R}$ is the projection onto the first component ($\pi_1(\mathcal{D}) = \{x_1, \ldots, x_n\}$). We thus need to establish a bound on

$$\begin{aligned} \delta_n = \Bigg| &\inf_{\lambda \geq 0} \lambda\epsilon - \frac{1}{n}\sum_{i=1}^{n} \sup_{x \in \mathcal{X}} \ell(h(x), y_i) + \lambda d_x^2(x, x_i) \\ &- \inf_{\lambda \geq 0} \lambda\epsilon + \frac{1}{n}\sum_{i=1}^{n} \max_{x \in \pi_i(D)} \ell(h(x), y_i) - \lambda d_x^2(x, x_i) \Bigg|. \end{aligned} \tag{5.15}$$

To do so, let λ_n be a minimizer of (5.14). Then, we have that

$$\begin{aligned} \delta_n &\leq \Bigg| \lambda_n\epsilon - \frac{1}{n}\sum_{i=1}^{n} \sup_{x \in \mathcal{X}} \ell(h(x), y_i) + \lambda_n d_x^2(x, x_i) \\ &\qquad - \lambda_n\epsilon + \frac{1}{n}\sum_{i=1}^{n} \max_{x \in \pi_i(D)} \ell(h(x), y_i) + \lambda_n d_x^2(x, x_i) \Bigg| \\ &= \left| \frac{1}{n}\sum_{i=1}^{n} \max_{x \in \pi_i(D)} \ell(h(x), y_i) + \lambda_n d_x^2(x, x_i) - \sup_{x \in \mathcal{X}} \ell(h(x), y_i) + \lambda_n d_x^2(x, x_i) \right| \end{aligned} \tag{5.16}$$

Define a map T on $\mathcal{D}$ such that $T(x_i) \in \arg\sup_{x \in \mathcal{X}} \ell(h(x), y_i) + \lambda_n d_x^2(x, x_i)$. By assumption, we have that

$$P_*(B_{n^{-1/2d}}(T(x_i)) \times \mathcal{Y}) \geq \delta/\sqrt{n}. \tag{5.17}$$

Thus, the probability that $\pi_1(\mathcal{D}) \cap \bigcup_i B_{n^{-1/2d}}(T(x_i)) = \emptyset$ (that is, the event that there are no points in the training set in$B_{n^{-1/2d}}(T(x_i))$) is upper bounded by $\sum_{i=1}^{n}(1 - \delta/n^{1/2})^n = n(1 - \delta/n^{1/2})^n$. Thus assume that for each i there is a point $(x_i^*, y_i^*) \in \mathcal{D}$ such that $x_i^* \in B_{n^{-1/2d}}(T(x_i))$. Then, continuing from (5.16), we have that

$$\delta_n \leq \frac{1}{n}\sum_{i=1}^{n} |\ell(h(x_i^*), y_i) - \ell(h(T(x_i), y_i)| + \left|\lambda_n(d_x^2(x_i^*, x_i) - d_x^2(T(x_i), x_i))\right| \tag{5.18}$$

$$\leq \frac{1}{n}\sum_{i=1}^{n} \omega d_x(x_i^*, T(x_i)) + |\lambda_n(d_x(x_i^*, x_i) - d_x(T(x_i), x_i))(d_x(x_i^*, x_i) + d_x(T(x_i), x_i))| \tag{5.19}$$

$$\leq \frac{1}{n}\sum_{i=1}^{n} \frac{\omega}{n^{1/2d}} + \left|\frac{2\lambda_n \mathrm{diam}(\mathcal{X})}{n^{1/2d}}\right|. \tag{5.20}$$

$$\leq \frac{\omega}{n^{1/2d}} + \frac{2\omega \mathrm{diam}(\mathcal{X})}{n^{1/2d}\sqrt{\epsilon}}. \tag{5.21}$$

In line (5.19), we use the fact that ℓ is ω-Lipschitz with respect to d_x (Assumption 5.3.3). In line (5.20), we use the fact that $x_i^* \in B_{n^{-1/2d}}(T(x_i))$ with the triangle inequality (to get that $d_x(x_i^*, x_i) - d_x(T(x_i), x_i) \leq d_x(x_i^*, T(x_i))$) and with the fact that $d_x(T(x_i), x_i) \leq \mathrm{diam}(\mathcal{X})$. Finally, in line (5.21), we apply Lemma A.1 from [YBS20] which asserts that $0 \leq \lambda_n \leq \frac{\omega}{\sqrt{\epsilon}}$. This completes the proof of the desired result. □

Note also that the exponent of $-\frac{1}{2d}$ in Equation (5.12) could be replaced with $-\frac{1}{(1+\rho)d}$ for any $\rho > 0$, though this would have an effect on the rate of convergence of the probability that the bound in Proposition 1 will hold. Furthermore, in general, it may be possible to take the value of d in Assumption 4 to be smaller than $\dim(\mathcal{X})$, leading to improvements in the bound. In particular, the distance d_x is assumed to be fair: usually, there are several directions in $\mathcal{X}$ that correspond to protected attributes, and d_x should ignore differences in these directions. That is, d_x corresponds to a distance on a space of dimension $k < \dim(\mathcal{X})$, so that potentially $\mathrm{vol}(B_r(x_i)) = \int_{B_r(x_i)} dx \approx c \cdot r^k$. This suggests that d could potentially be taken as k in these cases.

Compared to most theoretical studies of distributionally robust optimization, the proof of Theorem 5.3.5 is complicated by the restriction of the c-transform to the sample $\mathcal{D}$ in (5.14). This complication arises because the max over the sample is generally (much) smaller than the max over the sample space due to the curse of dimensionality. This discrepancy between the max over the sample and the max over the space space is responsible for the dependence of the rate of convergence on the dimension of the feature space in Theorem 5.3.5. See also [DN18] for an analysis of a similar result using Cressie-Read divergences rather than Wasserstein distances.

Theorem 5.3.5 can be used to show that L generalizes to the true population loss function:

Theorem 5.3.6. *Under Assumptions 5.3.1–5.3.4, we have*

$$|L(f) - L_r(f)| \to 0 \quad \text{as} \quad n \to \infty. \tag{5.22}$$

Proof of Theorem 5.3.6. The assumptions allow us to invoke Proposition 3.2 in [YBS20], which asserts that

$$|L_e(f) - L_r(f)| \to 0 \quad \text{as} \quad n \to \infty.$$

Then, the proof is completed by Theorem 5.3.5. □

5.4 Practical implementation considerations

As discussed in Section 5.2.2, the attack in line 5 of Algorithm 10 can be accomplished through a linear program. Unfortunately, this is a linear program in n^2 variables, where n is the size of the training data set $\mathcal{D}$. Considering that this linear program must be solved during every boosting step, and that it takes around 20 seconds to solve (on a modern laptop, using free LP solving software) when n is around 5000, it is important to improve the speed of this step for the scalability of the method.

In this section, we present several methods for efficiently obtaining a solution Π^* to 5.9. We start with a method based on entropic regularization that can be used to quickly obtain an approximate solution to the linear program in Equation 5.9. Moreover, this method requires only simple matrix computations; thus, we are able to implement it in TensorFlow for quick GPU computations.

It is also possible to quickly find the solution to the dual of the linear program (5.9). Obtaining the primal optimizer Π^* from the dual solution, however, can be difficult. For this reason, the Sinkhorn-based entropic regularization method runs significantly more quickly than the dual method in our implementations if we insist on using complementary slackness to determine the exact primal optimizer Π^*. In practice, we can quickly obtain another approximation to Π^* from the dual solution, however. A discussion of the dual method can be found in Section 5.4.3.

5.4.1 Entropic regularization

Equation (5.9) is an optimization problem over probability distributions. Thus, it is reasonable to regularize the objective function in Equation (5.9) with the entropy of the distribution.

As in Section 5.2.2, define the matrix R by $R_{i,j} = \ell(f, (x_i, y_j))$ - this is the loss incurred if point j with label y_j located at x_i. Moreover, define the matrix C by $C_{ij} = d_x^2(x_i, x_j)$, and let γ denote a regularization parameter. With this notation, including entropic regularization, the problem becomes:

$$\Pi^* = \arg\max_{\Pi \in \Gamma} \langle R, \Pi \rangle - \gamma \langle \log(\Pi), \Pi \rangle \tag{5.23}$$

where $\log(\Pi)_{ij} = \log(\Pi_{ij})$.

Adding the entropy of Π to the objective encourages the optimizer Π^* to be less sparse than the (low entropy) optimizer of the original optimal transport problem. Note that it is not inherently desirable to find a sparse optimizer Π^*, since we only consider the marginals of Π^* while boosting. Coming close to maximizing the original objective $\langle R, \Pi^* \rangle$ is far more important than sparsity.

We follow the Sinkhorn method to develop a solution to the problem (5.23). The Lagrangian is

given by

$$\mathcal{L}(\Pi, \lambda, \eta) = \sum_{ij} R_{ij}\Pi_{ij} - \gamma \sum_{ij} \log(\Pi_{ij})\Pi_{ij} - \sum_{j} \lambda_j (\sum_{i} \Pi_{ij} - \tfrac{1}{n}) - \eta(\sum_{ij} C_{ij}\Pi_{ij} - \epsilon) \quad (5.24)$$

so that at the optimum we have

$$0 = \frac{\partial \mathcal{L}}{\partial \Pi_{ij}} = R_{ij} - \gamma - \gamma \log(\Pi_{ij}) - \lambda_j - \eta C_{ij} = 0 \quad (5.25)$$

which means that

$$\Pi^*_{ij} = \exp(R_{ij}/\gamma) \exp(-\lambda_j/\gamma - 1) \exp(-\eta C_{ij}/\gamma). \quad (5.26)$$

Now, define matrices V and K and a vector u by the following expressions:

- $V_{ij} = \exp(LR_{ij}/\gamma)$,
- $K_{ij} = \exp(-\eta C_{ij}/\gamma)$, and
- $u_j = \exp(-\lambda_j/\gamma - 1)$.

Note that, by definition, we have that $\Pi^*_{ij} = V_{ij}K_{ij}u_j$. V is a constant (it does not depend on any of the variables of the Lagrangian). Exploiting the constraints of the set Γ (see the text before (5.9)), we see that u can be written in terms of K:

$$\sum_i \Pi_{ij} = u_j \sum_i K_{ij}V_{ij} = \frac{1}{n} \qquad \Rightarrow \qquad u_j = \frac{1}{n \sum_i K_{ij}V_{ij}}. \quad (5.27)$$

Thus, the solution Π^* depends only on K. Since K is determined completely by the Lagrange multiplier η, we need only to find the value of η that makes the other constraint of the set Γ tight:

$$\sum_{ij} C_{ij}\Pi_{ij} = \sum_{ij} C_{ij}V_{ij}K_{ij}u_j = \mathbf{1}_n^T (C \odot K(\eta) \odot V)u = \epsilon. \quad (5.28)$$

where $A \odot B$ denotes entrywise (Hadamard) product of the matrices A and B. We use a root finding algorithm to determine the optimal value of η: specifically, we find the root of $m(\eta) = \epsilon - \mathbf{1}_n^T (C \odot K(\eta) \odot V)u$. In our experiments, the bisection or secant methods generally converge in only around 10 to 20 evaluations of $m(\eta)$.

A fast method for evaluating $m(\eta)$ is presented in Algorithm 11. It consists of simple entrywise matrix operations (entrywise multiplications and exponentiations); this is highly amenable to processing on a GPU, and we have implemented this Sinkhorn-based method with TensorFlow. In Algorithm 12, we provide a method for using the optimal root η^* to obtain the coupling matrix Π^* following Equation 5.26.

Algorithm 11 Fast evaluation of Sinkhorn objective

Input: $\eta \geq 0$; cost matrix C; loss matrix R; tolerance ϵ; regularization strength γ.
Let $T \leftarrow \exp(R/\gamma - \eta C/\gamma)$. ▷ entrywise exponentiation
Let $u \leftarrow 1/(n\mathbf{1}_n^T \cdot T)$. ▷ entrywise reciprocation
return $\epsilon - \mathbf{1}_n^T \cdot (C \odot T) \cdot u$

Algorithm 12 Find Π using entropic regularization via (5.26)

Input: cost matrix C; loss matrix R; tolerance ϵ; regularization strength γ.
Let η^* be the root of Algorithm 11 with fixed inputs C, R, ϵ, and γ (The input η^* produces 0 in Algorithm 11)
Let $S \leftarrow \exp(R/\gamma - \eta^* C/\gamma)$.
Let $u \leftarrow 1/(n\mathbf{1}_n \cdot S)$.
Define Π by $\Pi_{ij} = S_{ij} u_j$.
return Π

5.4.2 Stochastic gradient descent for finding the root of $m(\eta)$

Although the Sinkhorn-based method presented above is fast, note that it is still not especially scalable, as it requires holding at least two $n \times n$ matrices (R and C) in memory at the same time. Assuming double precision floating point arithmetic, each of these matrices requires more than 10 GB of memory when n is approximately 4×10^4.

Thus, following [GCPB16], we express the determination of the optimal dual variable η^* as the minimization of an expectation over the empirical distribution P_n; thus, it is possible to leverage stochastic gradient descent (with mini-batching) to quickly find the optimal dual variable η^* while using a small amount of memory.

To develop this expression, note that the Lagrangian dual to the problem 5.23 is given by

$$\min_{\eta \geq 0, \lambda} \left[\max_{\Pi} \mathcal{L}(\Pi, \lambda, \eta) \right] \tag{5.29}$$

where $\mathcal{L}$ is defined in Equation 5.24. The optimal value of Π_{ij}^* for the inner maximum is established in Equation 5.26. Using this optimal value, we see that

$$\log(\Pi_{ij}^*) = \frac{1}{\gamma}\left(R_{ij} - \lambda_j - \eta C_{ij}\right) - 1 \tag{5.30}$$

and we can use this to calculate that

$$\mathcal{L}(\Pi^*, \lambda, \eta) = \eta\epsilon + \frac{1}{n}\sum_j \lambda_j + \gamma \sum_{ij} \Pi_{ij}^* = \eta\epsilon + \frac{1}{n}\sum_j \lambda_j + \gamma \sum_{ij} \exp\left(\frac{R_{ij} - \lambda_j - \eta C_{ij}}{\gamma} - 1\right). \tag{5.31}$$

To minimize this with respect to λ (which is unconstrained) we set

$$\frac{\partial}{\partial \lambda_j}(\mathcal{L}(\Pi^*, \lambda, \eta)) = \frac{1}{n} - \exp\left(\frac{-\lambda_j}{\gamma}\right) \sum_i \exp\left(\frac{R_{ij} - \eta C_{ij}}{\gamma} - 1\right) = 0 \tag{5.32}$$

which means that

$$\begin{aligned} \exp\left(\frac{-\lambda_j^*}{\gamma}\right) &= \frac{1}{n} \cdot \left(\sum_i \exp\left(\frac{R_{ij} - \eta C_{ij}}{\gamma} - 1\right)\right)^{-1} \\ \Rightarrow \quad \lambda_j^* &= -\gamma \log \tfrac{1}{n} + \gamma \log \left(\sum_i \exp\left(\frac{R_{ij} - \eta C_{ij}}{\gamma} - 1\right)\right). \end{aligned}$$

Thus, substituting into (5.31), we calculate that

$$\mathcal{L}(\Pi^*, \lambda^*, \eta) = \gamma \ + \sum_j \frac{1}{n} \left(\eta \epsilon + \gamma \log \left(\sum_i \exp\left(\frac{R_{ij} - \eta C_{ij}}{\gamma} - 1\right)\right) - \gamma \log \tfrac{1}{n}\right) \tag{5.33}$$

Thus, the value $\eta \geq 0$ that minimizes 5.33 is the same as the minimizer

$$\min_{\eta \geq 0} \mathbb{E}_{(x,y)\sim P_n} \left[\eta \epsilon + \gamma \log \left(\sum_i \exp\left(\frac{\ell(f, (x_i, y)) - \eta d_x^2(x_i, x)}{\gamma}\right)\right) \right] \tag{5.34}$$

where we ignored some constants that don't affect the minimizing value η^* and substituted in the definitions of R_{ij} and C_{ij} (see the paragraph at the start of Section 5.4.1).

The problem 5.34 is amenable to minimization via stochastic gradient descent (SGD) - this is presented in Algorithm 13. In every descent step of Algorithm 13, we are only working with a subset of the columns of R and C. Thus, R and C may be stored anywhere (or even computed on-the-fly) - they do not need to be kept in RAM or sent to the GPU memory. Thus, this SGD version can be used for essentially arbitrarily large data sets. Empirically, we also observe good results from running only a few gradient descent steps to obtain an approximation to η^* in every boosting step; this allows the SGD method to run as quickly as (or more quickly than) the normal Sinkhorn method (Algorithm 11). To find the final transport map Π, we either use Algorithm 12 (as before), or we use Algorithm 16, below, to rapidly compute an approximation to Π^* without requiring further memory usage.

5.4.3 Dual of robust empirical loss function L

Although the algorithms presented in the preceding section above run quickly and can handle arbitrarily large inputs, they only obtain an approximation to the true optimal transport map (due

Algorithm 13 SGD to find dual optimal dual variable η^*

Input: Starting point $\eta_1 > 0$; cost matrix C; loss matrix R; tolerance ϵ; regularization strength γ; batch size B, step sizes $\alpha_t > 0$.
repeat
  Sample indices $j_1, \dots, j_B$ uniformly at random from $\{1, \dots n\}$.
  Let $R_t \leftarrow$ columns $j_1, \dots, j_B$ of R. Let $C_t \leftarrow$ columns $j_1, \dots, j_B$ of C. $\triangleright$ $R_t, C_t \in \mathbb{R}^{n \times B}$
  Let $w_j^t(\eta) \leftarrow$ sum elements in column j of $\exp\left(\frac{1}{\gamma}(R_t - \eta C_t)\right)$ $\triangleright$ entrywise exponentiation
  $\eta_{t+1} \leftarrow \max\{0, \eta_t - \alpha_t \epsilon - \alpha_t \gamma \frac{d}{d\eta}\left[\sum_{j=1}^{B} \log w_j^t(\eta)\right]_{\eta=\eta_t}\}$
until converged
return η^*

to the entropic regularization). Here, we present a method to quickly obtain the solution to the dual of the original linear program (5.9). We find that constructing the transport map Π^* from the knowledge of the dual optimum is difficult; however, there are quick methods to produce reasonable approximations to Π^* from the knowledge of the dual solution.

Following standard methods, the dual of the linear program (5.9) is given by:

$$\begin{aligned} &\inf_{\eta \geq 0} \epsilon\eta + \tfrac{1}{n} \sum_{j=1}^{n} \nu_j \\ &\text{s.t. } \nu_j \geq R_{ij} - \eta C_{ij} \text{ for all } i, j \end{aligned} \tag{5.35}$$

Which can be simplified to:

$$\inf_{\eta \geq 0} \epsilon\eta + \frac{1}{n} \sum_{j=1}^{n} \max_i R_{ij} - \eta C_{ij}. \tag{5.36}$$

Define $\lambda_j(\eta) = \max_i R_{ij} - \eta C_{ij}$ and let $M(\eta) = \epsilon\eta + \frac{1}{n}\sum_{j=1}^{n} \lambda_j(\eta)$ denote the dual objective function from (5.36). Note that each $\lambda_j(\eta)$ is a monotone decreasing convex piecewise linear function of η. In addition, since $C_{jj} = 0$, we have that $\lambda_j(\eta) \geq R_{jj}$; that is, the λ_j are positive (following assumption 5.3.2). Moreover, each $\lambda_j(\eta)$ will become constant as $\eta \to \infty$ ($\lim_{\eta\to\infty} \lambda_j(\eta) = \max\{R_{ij} \colon C_{ij} = 0\} \geq R_{jj}$).

This means that the infimum (over η) will either occur at $\eta = 0$ or at one of the *corners* of one of the λ_j (a *corner* is a value of η such that there exist i_1 and i_2, $i_1 \neq i_2$, with $R_{i_1 j} - \eta C_{i_1 j} = R_{i_2 j} - \eta C_{i_2 j} = \max_i R_{ij} - \eta C_{ij}$; i.e. at least two of the lines that define λ_j are intersecting and creating a point where $M(\eta)$ is not differentiable). Thus, it is theoretically possible to exactly obtain the optimizer η^* by enumerating and testing all of these corners.

In practice, we have found that it is quicker to approximate η^* by noticing that an element of the

subgradient for $M(\eta)$ is given by

$$\epsilon - \frac{1}{n}\sum_j C_{i_j,j} \quad : \quad i_j \in \arg\max_i R_{ij} - \eta C_{ij}. \tag{5.37}$$

Thus, a bisection method can be implemented to approximate the point η^* such that $\frac{d}{d\eta}M(\eta) < 0$ for $\eta < \eta^*$ and $\frac{d}{d\eta}M(\eta) > 0$ for $\eta > \eta^*$. This is the corner that we are looking for. Using a bisection method (or similar) guarantees that we only need a small fixed number (≈ 30) of subgradient computations in every boosting step. An outline of this method is presented in Algorithm 14. Stochastic subgradient descent can also be used to obtain an approximate solution to 5.36 for larger data sets; this is outlined in Algorithm 15.

Algorithm 14 Approximate the optimizer η^* to the dual formulation (5.36) (following (5.37))

1: **Input**: Cost matrix C; loss matrix R; tolerance ϵ; root tolerance δ
2: Use the bisection method (or something similar) to find a value η^* such that for all η
   - $\sup\{\epsilon - \frac{1}{n}\sum_j C_{i_j j} \colon i_j \in \arg\max_i R_{ij} - \eta C_{ij}\} < 0$ when $\eta < \eta^* - \delta$
   - $\inf\{\epsilon - \frac{1}{n}\sum_j C_{i_j j} \colon i_j \in \arg\max_i R_{ij} - \eta C_{ij}\} \geq 0$ when $\eta > \eta^* + \delta$.
3: **return** η^*

Algorithm 15 Stochastic subgradient descent to approximate the optimizer η^* for (5.36)

1: **Input**: Starting point $\eta_1 > 0$; cost matrix C; loss matrix R; tolerance ϵ; batch size B; step sizes $\alpha_t > 0$.
2: **repeat**
3: Sample indices $j_1, \ldots, j_B$ uniformly at random from $\{1, \ldots n\}$.
4: Let $R_t \leftarrow$ columns $j_1, \ldots, j_B$ of R. Let $C_t \leftarrow$ columns $j_1, \ldots, j_B$ of C. ▷ $R_t, C_t \in \mathbb{R}^{n \times B}$
5: Let $w_t(\eta) \leftarrow \eta \cdot \epsilon + \frac{1}{B}\sum_{j=1}^{B} \max_i R_{ij} - \eta C_{ij}$
6: $\eta_{t+1} \leftarrow \max\{0, \eta_t - \alpha_t \frac{d}{d\eta} w_t(\eta)|_{\eta=\eta_t}\}$ ▷ If w_t is not differentiable at η_t, use (5.37) with R_t, C_t
7: **until** converged
8: **return** η^*

Complementary slackness can directly be exploited to quickly find the desired solution Π^* to the original discrete SenSR LP from the dual optimizer η^* if the following conditions hold:

1. There is an index j such that the point $\eta^* = \arg\min_{\eta \geq 0} M(\eta)$ is a corner of λ_j with $|\arg\max_i R_{ij} - \eta^* C_{ij}| = 2$ (only two of the lines that define λ_j intersect at η_*).
2. For all $k \neq j$, the point η^* is not a corner of λ_k (i.e. $|\arg\max_i R_{ik} - \eta^* C_{ik}| = 1$).

These conditions are based on the fact that, for all i and j, Π_{ij} is the primal variable corresponding to the constraint $v_j \geq R_{ij} - \eta C_{ij}$ in the dual (5.35). Condition 2 implies that, for all $k \neq j$, there

is a value of t such that $\Pi^*_{tk} = \frac{1}{n}$ and $\Pi^*_{ik} = 0$ for all $i \neq t$. Then, condition 1 implies that there are only two nonzero values in column j of Π^*. These two nonzero values must sum to $\frac{1}{n}$ and the constraint $\langle C, \Pi^* \rangle = \epsilon$ must be satisfied. With the complete knowledge of the other values of Π^*, this results in two linear equations with two unknowns (the two remaining nonzero values of Π^*), and this system can be easily solved to give the exact solution Π^*.

Similar tricks can be applied to rapidly solve for Π^* when slightly more entries are candidates for being nonzero (according to complementary slackness). Unfortunately, since d_x is a fair distance (and thus it usually ignores several directions in $\mathcal{X}$), it is often the case that there are multiple individuals $x_i \neq x_j$ in the training data such that $d_x(x_i, x_j) = 0$ (i.e. there are a significant number of off-diagonal entries of C that are 0). In practice, we have observed that these points also often produce the same predicted value $f(x_i) = f(x_j)$ after the first step of boosting. This results in a situation where complementary slackness presents a complicated system of equations for obtaining the primal optimizer Π^*.

Thus, in order to maintain an efficient boosting algorithm in practice, we use the dual optimizer η^* to construct an approximation $\hat{\Pi}$ to the primal optimizer Π^* using the following heuristic: for all j, randomly select t in $\arg\max_i R_{ij} - \eta^* C_{ij}$ and let $\hat{\Pi}_{tj} = \frac{1}{n}$. That is, in each column, randomly select one of the candidate nonzero entries (according to complementary slackness) and set it to $\frac{1}{n}$ in $\hat{\Pi}$. This approximation $\hat{\Pi}$ actually is an interpretable transport map: each point in the training data is mapped (completely) to another point in the training data. We present the construction of $\hat{\Pi}$ using η^* in Algorithm 16 in the main text, and we use Algorithm 16 in our experiments on the German credit data set the Adult data set in Section 5.5. Although it is not exactly clear as to how good (or bad) of an approximation $\hat{\Pi}$ is to Π^*, our experimental results show that it is functional.

Algorithm 16 Approximate Π^* from the dual optimizer η^*

1: **Input**: Cost matrix C; loss matrix R; value of η^*
2: Let Π be an $n \times n$ matrix of zeros.
3: **for** $j = 0$ to $n - 1$ **do**
4: Choose $t \in \arg\max_i R_{ij} - \eta^* C_{ij}$
5: Set $\Pi_{tj} = \frac{1}{n}$.
6: **end for**
7: **return** Π

5.4.4 Fair gradient boosted trees

Algorithm 17 summarizes our method BuDRO for constructing individually fair gradient boosted trees that combines the framework presented in the previous sections with a GBDT method. We refer to the GBDT method as TreeBoost; this can be replaced with any GBDT algorithm.

In every boosting step in Algorithm 17, we find the optimal transport map Π^* using one of the methods discussed in this section:

- Entropic regularization, discussed in 5.4.1 and outlined in Algorithm 12,
- SGD on the entropic regularization dual, discussed in 5.4.2 and outlined in Algorithm 13,
- The dual method, discussed in 5.4.3 and outlined in Algorithms 14 and 16,
- Stochastic subgradient descent on the dual to the original linear program (5.36), outlined in Algorithms 15 and 16.

We then boost for one step using the XGBoost training methods on the duplicated data set $\mathcal{D}_0$ weighted according to the distribution given by $P(\{x_i, k\}) = (\Pi^* \cdot y^k)_i$. The full details are outlined in Algorithm 17. There are multiple hyperparameters that can be tweaked in the GBDT model (e.g. the maximum depth of the trees in $\mathcal{F}$); we represent this by including a list of XGBoost parameters ρ.

5.5 Experiments

In this section, we report the results of experiments using the BuDRO algorithm (see Algorithm 17). We start with a synthetic motivation, then consider two real world data sets that are popular in the fairness literature: the German credit data set and the Adult data set [DG17]. BuDRO uses the XGBoost algorithm [CG16] for the GBDT method, and ℓ is the logistic loss. These experiments reveal that BuDRO is successful in enforcing individual fairness while achieving high accuracy (leveraging the power of GBDTs). We also observe improvement of the group fairness metrics.

5.5.1 Synthetic motivation

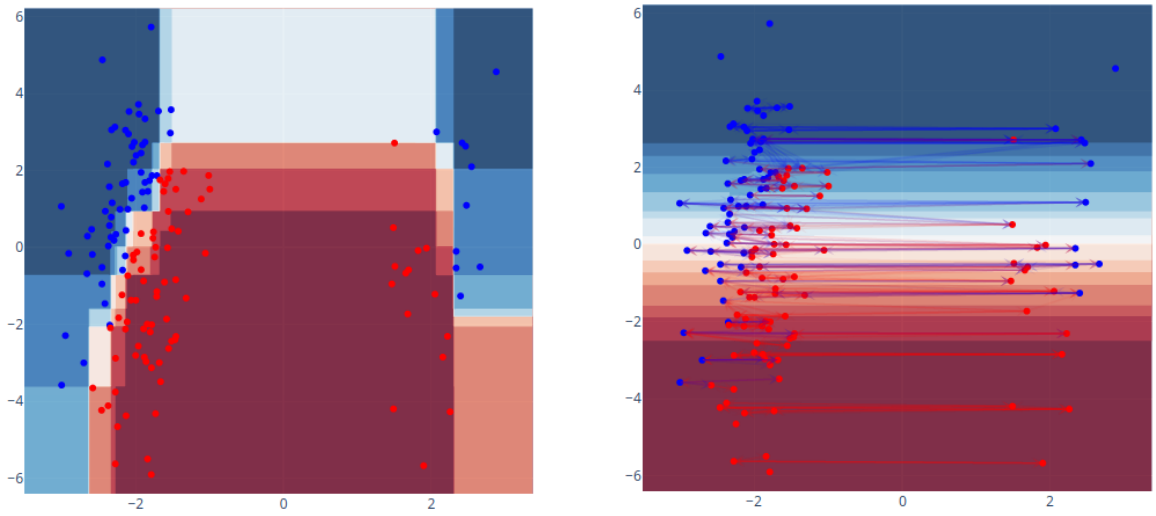
Consider a data set with two features, x_1 and x_2. Suppose that one feature x_1 is protected and the other feature x_2 is can be used for making fair decisions. For example, our task may be to decide which individuals to approve for a loan; in this case, the protected feature x_1 may correspond to the percentage of nonwhite residents in the applicant's zip code, and the other feature x_2 may correspond to the applicant's credit score (or some other fair measure of credit worthiness). In this case, we can approximate a fair metric d_x by the difference in the second feature: the fair distance between the two individuals (x_1^a, x_2^a) and (x_1^b, x_2^b) is given by $d_x((x_1^a, x_2^a), (x_1^b, x_2^b)) = |x_2^a - x_2^b|$.

We synthetically constructed such a data set in the following manner: 150 individuals were independently drawn from the same centered normal distribution. Let R be a rectangle of minimum area containing the 150 points; let L_1 be the line passing through the top right corner and the

Algorithm 17 Fair gradient boosted trees (BuDRO)

Input: Data $\mathcal{D} = \{(x_i, y_i)\}_{i=1}^n$; perturbation budget ϵ; loss function ℓ; fair metric d_x on $\mathcal{X}$; number of boosting steps T; entropic regularization parameter γ; batch size B; XGBoost parameters ρ.
Let $\mathcal{D}_0 = \{(x_i, 0)\}_{i=1}^n \cup \{(x_i, 1)\}_{i=1}^n$
Define C by $C_{ik} \leftarrow d_x(x_i, x_k)$.
Let $f_0 = \mathrm{TreeBoost.Train}(\rho, \mathrm{data} = \mathcal{D}_0, \mathrm{numSteps} = 1)$ ▷ Run one step of naive boosting
for $t = 0$ to $T - 1$ **do**
    Define R by $R_{ik} = \ell(f_t, (x_i, y_k))$.
    if $B > 0$ **then** ▷ Non-zero batch size - use SGD
        if $\gamma > 0$ **then** ▷ Use entropic regularization framework
            $\eta^* \leftarrow$ the output of Algorithm 13 with inputs C, R, ϵ, γ, and B.
            $\Pi_t \leftarrow$ the output of the framework in Algorithm 12 (or Algorithm 16) using η^*.
        else ▷ Use dual framework
            $\eta^* \leftarrow$ the output of Algorithm 13 with inputs C, R, ϵ, and B.
            $\Pi_t \leftarrow$ the output of Algorithm 16 with inputs C, R, and η^*.
        end if
    else ▷ $B = 0$: don't use SGD
        if $\gamma > 0$ **then** ▷ Use entropic regularization framework
            $\Pi_t \leftarrow$ the output of Algorithm 12 with inputs C, R, ϵ, and γ.
        else ▷ Use dual framework
            $\eta^* \leftarrow$ the output of Algorithm 14 with inputs C, R, ϵ, and a small value δ.
            $\Pi_t \leftarrow$ the output of Algorithm 16 with inputs C, R, and η^*.
        end if
    end if
    Let w_t be the concatenation of $\Pi_t \cdot y^0$ and $\Pi_t \cdot y^1$. ▷ y^0, y^1 defined in (5.8)
    Let $f_{t+1} \leftarrow \mathrm{TreeBoost.Train}(\rho, f_t, \mathrm{data} = \mathcal{D}_0, \mathrm{weights} = w_t, \mathrm{numSteps} = 1)$.
end for
Return f_T.

bottom left corner of R and L_2 be the line passing through the top left corner and the bottom right corner of R. 125 of the samples were chosen (uniformly at random) to belong to majority white neighbourhoods: these individuals were labeled 0 if they were below L_1 and 1 if they were above L_1, and then they were all shifted to the left (by 2, so that the highly white cluster is centered at $(-2, 0)$). The remaining 25 individuals make up the majority nonwhite cluster: they were classified according to L_2 and then centered at $(2, 0)$.



(a) Naive boosted tree classifier

(b) Fair boosted tree classifier

Figure 5.1: Comparison of GBDT classifiers without (a) and with (b) the fairness constraints introduced in this paper. The classifiers output a probability in $[0, 1]$ - these probabilities are discretized to binary labels to create a classification. In the figures, red points are individuals with true label 0 and blue points are individuals with true label 1. The darker red areas correspond to lower output probabilities and the darker blue areas correspond to higher output probabilities. The arrows in (b) indicate the transport map corresponding to the fairness constraint for the previous boosting step.

In Figure 5.1, we show such an example of this setup. The red points in the figure correspond to individuals that are labeled 0; these represent individuals that should be declined for the loan (e.g., they have defaulted on a loan, with these data collected in the past to make future predictions). The blue points are individuals that are labeled 1 and represent people who should be approved (e.g. they have paid back a loan in the past). The horizontal axis represents the fraction of nonwhite residents in an individual's zip code, while the vertical axis is taken as some fair measure of credit worthiness.

The naively trained GBDT classifier in Figure 5.1(a) shows high accuracy on this synthetic data set, but it is unfair - requiring that individuals from neighbourhoods containing medium to high percentages of nonwhite individuals maintain a significantly higher credit score than those in very

racially homogeneous neighbourhoods in order to be approved for a loan. Applying our individually fair gradient boosting algorithm to the data set, however, results in a fair classifier (visualized in Figure 5.1(b)). The credit score threshold for loan acceptance is consistent across the different racial make ups of zip codes. This provides a synthetic visualization of the performance of the BuDRO method and suggests that it works as described.

As a tangent: note that the clusters in this example are generated in a symmetric manner: this is a case where the unfairness in the naively trained classifier stems from a lack of data and a press for extra accuracy rather than a specific inherent bias in the data.

5.5.2 Details common to experimental data sets

We also apply the BuDRO method to two common ML tasks on well-studied data sets from the fairness literature: the German credit data set and the Adult data set. Here, we discuss the portions of the experimental structure that are common to both of these data sets.

Fair metric Recall that a practical implementation of our algorithm requires a fair metric. We consider the procedure from [YBS20] to learn a fair metric from data. They propose a Mahalanobis distance of the form $d_x^2(x_1, x_2) = \langle x_1 - x_2, P(x_1 - x_2)\rangle$, where P is the projecting matrix orthogonal to some (problem specific) sensitive subspace. Similar to the work in Chapter 4, the sensitive subspace is constructed to capture variation in the data due to protected information. A fair metric should treat individuals that only differ in their protected information similarly; i.e. the fair distance between a pair of individuals that only differ by a component in the sensitive subspace should be 0.

To be precise, for a given data set, we determine a finite set T of protected directions in $\mathcal{X}$; we treat T as a basis for the sensitive subspace. The fair distance d_x is thus defined by projecting onto the orthogonal complement of $\mathrm{span}(T)$ and considering the Euclidean distance on the projected space. In particular, let $A = \mathrm{span}(T)$ and $\mathrm{proj}(A)$ denote the projection onto A. Then,

$$d_x(x_1, x_2) = \|((I - \mathrm{proj}(A)) \cdot (x_1 - x_2)\|_2. \tag{5.38}$$

The set of protected directions T is determined in a similar fashion for each experimental data set. In particular, each data set contains one or more protected attributes (e.g. gender, race or age): the indicator[5] for each protected attribute is included in T. We additionally obtain one or more additional protected directions in the following manner. For a fixed protected attribute g (e.g. gender), we remove the feature g from the data set and train a linear model on the edited data set to predict the removed feature g (we use logistic regression with an ℓ_2 regularization of strength 0.1

[5] the indicator for an attribute is a vector with only one nonzero entry; that nonzero entry appears in the attribute that we are indicating.

when g is binary an we use ridge regression with cross validation for a non-binary g). Let w be the normal vector to the separating hyperplane corresponding to this linear model: we include w in T, the set of protected directions.

For example, on the Adult data set, we consider both the `gender` and `race` features as protected attributes (both features are binary, see the description in Section 5.5.4). The set T for Adult then contains three protected directions. Two of these protected directions are given by e_g, the indicator vector of the `gender` feature, and e_r, the indicator vector of the `race` feature. We obtain the third protected direction w by removing the `gender` feature from the data set and training logistic regression on this edited data set with targets given by the `gender` feature. Several of the features in the Adult data set (such as the `is_husband` and `is_wife` categories of the `relationship` feature) are highly predictive of gender. Combined with the gender imbalance in the data set, this allows for logistic regression to be able to predict gender with nearly 80% accuracy on a (holdout) test set. We thus take w to be the normal vector to the separating hyperplane discovered during this logistic regression.

Note that a metric defined in this way will assign a distance of 0 between two individuals that differ only in the protected attributes (e.g. gender or race) but are identical in all other features. Additionally, if the difference $x_1 - x_2$ between two individuals is nearly parallel to one of the logistic regression directions u, then $d(x_1, x_2)$ will also be small. For example, on the Adult data set, the vector w (defined in the previous paragraph) exhibits comparatively high support on the `is_husband` and `is_wife` categories of the `relationship` feature. Thus, if x_1 and x_2 are identical except that x_1 has `is_wife` $= 1$ and x_2 has `is_husband` $= 1$, then $d_x(x_1, x_2)$ will be small.

This fair metric d_x is an approximation to an actual fair metric on $\mathcal{X}$: it does not capture information about all protected features, and only makes a heuristic approximation to reduce differences between true race and gender groups of individuals.

Comparison Methods We consider other possible implementations of fair GBDTs using existing techniques from the literature. As previously mentioned, due to the non-differentiability of trees, the majority of other individual fairness methods are not applicable to the GBDT framework. For this reason, we limit our analysis to fair data preprocessing techniques before applying a vanilla GBDT method. We report the results when considering two preprocessing techniques: *projecting* and *reweighing*.

The *projecting* preprocessing method functions by eliminating the entire protected subspace from the data set before training a vanilla GBDT (see, for example Chapter 4, [YBS20, PTB19]). In particular, using the notation from the discussion of fair metrics (above), we project all of the data onto the orthogonal complement of $\mathrm{span}(T)$ as a preprocessing step. This has the effect that the final

classifier will be completely blind to differences in the protected directions in T. Our experiments (see e.g. Table 5.5) show that this is not enough to produce an individually fair classifier: attributes that are highly correlated with elements of T are still used to make classification decisions.

Reweighing is presented in [KC11]. Essentially, this method balances the representations of protected groups by assigning weights to the individuals in the training data. These weights are chosen to force the protected group status to appear statistically independent to the outcomes in $\mathcal{Y}$. This is inherently a group fairness method (the data are weighted to match a group fairness constraint) that cannot be used when the protected attribute takes an infinite number of values.

Our implementations of BuDRO, projecting, and reweighing all use the XGBoost framework [CG16]; we also compare to a baseline of training vanilla XGBoost with no preprocessing. The GBDT hyperparameters that we consider are:

- `max_depth`, the maximum depth of the decision trees used as weak learners;
- `lambda`, an ℓ_2 regularization parameter;
- `min_weight`, a tree regularization parameter; and
- `eta`, the boosting learning rate.

We additionally examine the effects of boosting for different numbers of steps. We always present the results of reweighing using the default XGBoost hyperparameters.

To test if GBDT classifiers are useful on the tabular data sets that we consider here, we also train (naive) one-layer 100 unit fully connected neural networks on all data sets. For the Adult data set, we additionally compare to the SenSR method from [YBS20] and the adversarial debiasing method from [ZLM18]. The SenSR method creates an individually fair neural network through stochastic gradient descent on a robust loss similar to the one considered in this work. Adversarial debiasing is based on minimizing the ability to predict the protected attributes from knowledge of the final outputs of a predictor, and also draws on ideas from robustness in machine learning. It is constructed to provide improvements to statistical group fairness quantities rather than for the creation of an individually fair classifier.

Evaluation measures To evaluate the individual fairness of each method without appealing to the underlying fair metric (i.e. to avoid giving our method and projecting an unfair advantage at test time and to verify that the fair metrics used for training in this work are good approximations to true fair metrics), we report data set specific *consistency evaluation measures*. Specifically, for each data set, we find a set of attributes that are not explicitly protected but are correlated with protected attributes (e.g. `is_husband` or `is_wife` when `gender` is protected) and vary these attributes to create artificial counterfactual individuals. Such counterfactual individuals are intuitively similar

to the original individuals[6] and thus classifier output should be the same for all counterfactuals to satisfy individual fairness. The classification consistency is then computed as the frequency that all counterfactual individuals are assigned to the same outcome. This type of consistency evaluation is inspired by the work [GPL+18] that examined changes in predicted sentiment when changing one word in a sentence.

To be formal, suppose an attribute g takes values in a set V. To measure the consistency of the predictor f with respect to g (the "g-consistency of f"), we construct $|V|$ copies of the test data. These copies are altered so that the value of the attribute g is constant on each copy and so that each value in V is represented in a copy of the data. We then apply f to each copy of the data to obtain $|V|$ vectors of predicted outcomes $\hat{y}_1, \ldots, \hat{y}_{|V|}$, $\hat{y}_j \in \mathcal{Y}^n$. The g-consistency of f is then the fraction of individuals who are assigned the same outcome in every copy of the data set. That is, it is the fraction of indices i such that $(\hat{y}_1)_i = (\hat{y}_2)_i = \ldots = (\hat{y}_{|V|})_i$

It is not generally true that individual fairness will imply group fairness: it will depend on the group fairness constraint in question as well as the fair metric d_x on $\mathcal{X}$. We thus also report several group fairness metrics for completeness. We consider the group fairness metrics introduced in [DARW+19], discussed in more detail below. it remains future work to establish the precise conditions under which individual fairness will imply group fairness.

Suppose that there are two groups of individuals, labeled by $r = 0$ (a protected group) and $r = 1$ (a privileged group). Then, given true outcomes Y and predicted outcomes $\hat{Y}$, for each possible outcome $y \in \mathcal{Y}$ we can consider a statistical fairness gap defined by

$$\mathrm{Gap}_y = P(\hat{Y} = y | Y = y, r = 0) - P(\hat{Y} = y | Y = y, r = 1).$$

Note that a large value of $|\mathrm{Gap}_y|$ corresponds to (correctly) assigning the outcome y to a larger fraction of individuals belonging one of the groups than the other. For example, suppose that $\mathcal{Y} = \{0, 1\}$, with an 1 indicating a favorable outcome. Then, a large value of $|\mathrm{Gap}_1|$ means that our classifier is able to identify the successful individuals from one group at a higher rate than it is able to identify the successful individuals from the other. Thus, large values of Gap_y indicate unfair performance by the classifier at the group level. The group fairness measures from [DARW+19] are then given by

$$\begin{aligned} \textsc{Gap}_{\mathrm{Max}} &= \max_{y \in \mathcal{Y}} |\mathrm{Gap}_y| \\ \textsc{Gap}_{\mathrm{RMS}} &= \sqrt{\frac{1}{|\mathcal{Y}|} \sum_{y \in \mathcal{Y}} \mathrm{Gap}_y^2}. \end{aligned} \tag{5.39}$$

[6] This is somewhat debatable. For example, does changing the `gender` feature but keeping all other attributes the same truly result in an equivalent individual of a different gender?

5.5.3 German credit data set

German credit is a data set that is commonly evaluated in the fairness literature [DG17]. It contains information about 20 attributes (seven numerical, 13 categorical) from 1000 individuals; the individuals are labeled as being good or bad credit risks. Several of the features, such as `amount_in_savings` and `length_current_employment`, are numerical but have been recorded as categorical. Additionally, some of the other features, such as `employment_skill_level` and `credit_history_status` are categorical but could be considered to be ordered (for example, "all credits paid" is better than "past delays in payments" which is better than "account critical"). In some existing analyses, these ordered categorical features are converted to integers; for the purposes of this analysis, we one-hot encode all of the categorical features (which results in 62 features in our input data). We additionally standardize each numerical feature (that is, center by subtracting the mean and divide by the standard deviation). None of the data points are removed.

We treat `age` (a numerical feature that we standardize) as the protected attribute in the German credit data set. The `age` feature is not binary; this precludes the usage of fair methods that assume two protected classes (including the reweighing preprocessing technique). In order to report the group fairness metrics, we consider two protected groups: one group consists of individuals who are younger than 25, the other group consists of those who are 25 or older. This split was proposed by [KC09] to formalize the potential for age discrimination with this data set.

In the United States, it is unlawful to make credit decisions based on the or the age of the applicant. Thus, there are distinct legal reasons to be able to create a classifier that can be shown to be fair based on the non-binary age feature.

Fair metric For the German credit data set, we consider two protected directions, the first of which is the indicator vector e_a of the `age` attribute. Additionally, following the framework described in 5.5.2, we eliminate the `age` attribute from the data and use ridge regression to train a classifier to predict age from the other features (we use the default parameters in the `RidgeCV` class from the `scikit-learn` package, version `0.21.3` [PVG+11]). The second protected direction w is a normal vector to the hyperplane created by ridge regression (i.e. it is the vector of ridge regression coefficients).

Evaluation metrics As discussed above, we consider the $\mathrm{Gap}_{\mathrm{RMS}}$ and $\mathrm{Gap}_{\mathrm{MAX}}$ for a binarized version of the `age` attribute.

For an individual fairness metric, we cannot examine an age consistency, since age is a numerical feature. German credit contains a categorical `personal_status` feature that encodes some

information about gender and marital status[7], however. After one-hot encoding, we find that several of the `personal_status` categories are well-correlated with the `age` attribute according to ridge regression. Specifically, the `male_divorced/separated` category is positively correlated with age (the ridge regression coefficient has an average of 0.26 over our 10 train/test splits) and the `male_married/widowed` feature is negatively correlated with age (the ridge regression coefficient has an average of -0.25). Since this `personal_status` attribute can be considered to be a protected attribute, we examine a consistency measure based on this `personal_status` attribute as described in 5.5.2. We refer to this consistency measure as *status consistency* (or *S-cons*).

Training and hyperparameter selection The labels of the German credit data set are quite unbalanced (only 30% of the labels are 1). We thus train all ML methods to optimize the balanced accuracy. For the GBDT methods (baseline, projecting, and BuDRO), this is accomplished by setting the XGBoost parameter `scale_pos_weight` to $\frac{0.7}{0.3}$, the ratio of zeros to ones in the labels of the training data. For naive (baseline) NNs, we sample each minibatch to contain equal numbers of points labelled 0 and labelled 1.

For both the baseline GBDT and the projecting method, we search over a grid of GBDT hyperparameters on ten 80% train/20% test splits and choose the set of hyperparameters that optimizes the average balanced test accuracy over those ten splits. See 5.5.2 for the parameters that we tune and contact the author for the code to find the exact values considered. For BuDRO, we tune parameters by hand on one 80% train/20% test split. We use the dual implementation to find the optimal transport map (without SGD, see Algorithms 14 and 16); thus the only extra hyperparameter to consider is the perturbation budget ϵ. The data in Table 5.3 are collected by averaging the results obtained using the optimal hyperparameter choices across 10 new 80% train/20% test splits (the same splits for each method). The optimal hyperparameters are presented in Table 5.2.

Method	`max_depth`	`lambda`	`min_weight`	`eta`	steps
Baseline	10	1000	2	0.5	105
Projecting	7	2000	2	0.5	111
BuDRO	4	1.0	1/80	0.005	90

Table 5.2: Optimal XGBoost parameters for German credit data set. For BuDRO, we also used a perturbation budget of $\epsilon = 1.0$.

We also train a neural network on the German credit data set (see Section 5.5.2 for a description of the architecture). We find that we consistently obtain high accuracy when we use a learning rate of 10^{-4} and run for 4100 epochs (without any ℓ_2 regularization).

[7]The `personal_status` categories are `male_single`, `male_married/widowed`, `male_divorced/separated`, `female_single`, and `female_divorced/separated/married`.

Results We present the results in Table 5.3. To compare classification performance we report balanced accuracy due to class imbalance in the data. The baseline (GBDTs with XGBoost) is the most accurate and significantly outperforms the baseline neural network (NN). The BuDRO method has the highest individual fairness (S-cons) score while maintaining a high accuracy and improved group fairness metrics. We see that a simple preprocessing by projecting out the sensitive subspace is not as effective as BuDRO in improving individual fairness and also can negatively impact the accuracy.

Note that BuDRO produces a higher accuracy than the vanilla NN method; this shows that BuDRO is able to leverage the strength of GBDTs on tabular data while producing individual fairness. Following the trends in the data presented in [YBS20], we would expect for BuDRO to also be more accurate than other NN-based fairness methods (e.g. Adversarial Debiasing).

			Age gaps	
Method	BAcc	Status cons	GAP_{Max}	GAP_{RMS}
BuDRO	0.715±0.032	**0.974**±0.025	**0.185**±0.055	0.151±0.048
Baseline	**0.723**±0.019	0.920±0.022	0.310±0.159	0.241±0.109
Project	0.698±0.024	0.960±0.029	0.188±0.086	**0.144**±0.064
Baseline NN	0.687±0.031	0.826±0.028	0.234±0.126	0.179±0.093

Table 5.3: Results on German credit data set. We report the balanced accuracy in the second column. These results are averages from 10 splits into 80% training and 20% test data; the standard deviations are also reported.

5.5.4 Adult data set

The Adult data set [DG17] is a commonly considered benchmark data set in the algorithmic fairness literature. After preprocessing and removing entries that contain missing data, we consider a subset of Adult containing 11 demographic attributes from $45{,}222$ individuals. Each individual is labelled with a binary that is 0 if the individual makes less than $\$50{,}000$ per year and 1 if the individual makes more than $\$50{,}000$ per year; the ML task is to predict this label. In our preprocessing, we standardize the (five) continuous features and one-hot encode the (six) categorical features, resulting in 41 total features that we use in our analysis. These features include several attributes that could be considered protected, such as `age` (continuous), `marital_status` (categorical), `gender` (binary), and `race` (here we consider a binary white vs non-white `race` feature). For this experiment, we choose to construct a predictor f for the labels that is fair according to the `gender` and `race` attributes.

We follow the precise experimental setup and consider the same comparison metrics from the prior work on individual fairness [YBS20] studying this data. In particular, we do not remove the

gender and race attributes from the training data. This helps to produce an adversarial analysis (how fair can the method become when it is explicitly training on gender and race). This also more closely matches the methods used in previous analyses, to allow for easier comparisons.

Fair distance The fair distance on Adult is described in detail in 5.5.2. Briefly, we consider three protected directions: e_g, the indicator for the gender attribute; e_r, the indicator for the race attribute; and w, the normal vector to the separating hyperplane obtained via logistic regression trained to predict the gender attribute from the other attributes.

Evaluation metrics As previously mentioned, we consider both gender and race as protected attributes on the Adult data set; thus we report $\mathrm{Gap}_{\mathrm{Max}}$ and $\mathrm{Gap}_{\mathrm{RMS}}$ for both the gender and race features.

We examine two types of counterfactual individual fairness metrics. The first we refer to as *spouse consistency* (S-cons). The S-cons is determined by creating two copies of the data: one in which every point belongs to the husband category of the relationship_status feature and the other in which every point belongs to the wife category. Unlike the framework described in 5.5.2, we do not consider all categories of the relationship_status feature. To be concrete, two altered copies of the data are used to obtain two vectors of predicted outcomes $\hat{y}^h$ and $\hat{y}^w$, and the S-cons is the fraction of outcomes that are the same in these two vectors. Explicitly, S-cons is calculated as $\frac{1}{n}|\{i\colon \hat{y}_i^w = \hat{y}_i^h\}|$. Intuitively, the S-cons measures how likely an individual is to be assigned to a different outcome simply from labeling themselves as a wife rather than a husband.

The other evaluation metric is the *gender and race consistency* (GR-cons), which involves four copies of the input data. Each copy is altered so that the gender and race features are constant on that copy of the data, and so that each copy has a different combination of the race and gender feature from the other three copies. We then apply the classifier to each copy of the data, to produce four vectors of predicted outcomes $\hat{y}_i \in \{0,1\}^n$, $i = 0, 1, 2, 3$. The GR-cons is defined as the fraction of outcomes that are the same in all of the y_i. Note that the projecting preprocessing method is guaranteed to have a perfect GR-cons score since the gender and race features are explicitly projected out by this method; it is interesting to assess if projecting obtains a favorable S-cons score, however.

Training and hyperparameter selection The labels for the Adult data set are unbalanced (only about 25% of people make more that $\$50{,}000$ per year) and thus all methods are again trained to maximize balanced accuracy. For the GBDT methods (baseline, projecting, BuDRO) this is accomplished by setting the XGBoost parameter scale_pos_weight to be the ratio of zeros to ones in the labels of the training data. The data from the other methods (baseline NNs, SenSR, and

adversarial debiasing) are obtained from [YBS20]; see that work for further information about the method training.

The baseline GBDT and projecting methods were trained to optimize balanced test accuracy over a grid of hyperparameters on one 80% train/20% test seed. The optimal hyperparameters discovered in this way were then used to collect data on ten different 80% train/20% test splits: the average of the results on the test data from these new train/test splits are presented in Table 5.5 (see Table 5.6 for information about standard error). The GBDT hyperparameters that maximize accuracy are presented in Table 5.4.

Method	`max_depth`	`lambda`	`min_weight`	`eta`	steps
Baseline	3	0.01	0.5	0.05	816
Projecting	8	0.5	0.5	0.05	668
BuDRO	14	10^{-4}	0.1/36177	0.005	180

Table 5.4: Optimal XGBoost parameters for the Adult data set. For BuDRO, we also used a perturbation budget of $\epsilon = 0.4$. The value of `min_weight` for BuDRO is computed relative to the size of an 80% training set, which contains 36177 individuals.

BuDRO was implemented using the Dual SGD formulation to find the optimal transport map Π as described in Algorithms 15 and 16. This involves several additional hyperparameters that we set in the following ways: an initial guess of the dual variable (set to 0.1) and a batch size (set to 200). Our implementation used also included a maximum number of SGD iterations (set to 100), a momentum parameter (set to 0.9), and an SGD learning rate (set to 10^{-4}). The optimal perturbation budget ϵ was found to be 0.4. See Section 5.5.4.1 for more information about the hyperparameter selection on Adult.

5.5.4.1 Results

Results when methods are trained to maximize balanced accuracy are presented in Table 5.5 (the standard errors can be found in Table 5.6). The baseline GBDT method is again the most accurate; it produces poor gender gaps, however. BuDRO is less accurate than the baseline, but the gender gaps have shrunken considerably, and both the S-cons and GR-cons are very high. Projecting and reweighing produce the best group fairness results; however, their S-cons values are worse than the baseline (indicating individual fairness violations) and they also result in significant accuracy reduction. As previously noted, the projecting method eliminates the protected attributes from the data set and thus it trivially attains a perfect GR-cons of 1 (the predictor constructed in this way cannot directly depend on the gender or race attributes). Nonetheless, the S-cons of projecting is low - it does not produce individual fairness.

Comparing to the results from [YBS20], BuDRO is slightly less accurate than the baseline NN,

but it improves on all fairness metrics. BuDRO matches the accuracy of adversarial debiasing and greatly improves the individual fairness results there. Finally, BuDRO improves upon the accuracy of SenSR while maintaining similar individual fairness results. We present additional studies of the trade-off between accuracy and fairness in a following discussion.

The baseline GBDT method is only slightly more accurate than the baseline NN method on the Adult data set; nonetheless, BuDRO is able to use this power to construct an accurate classifier with favorable individual fairness consistency scores and group fairness gaps that are improved from the baseline. Overall, this and the preceding experiment provide empirical evidence that BuDRO trains individually fair classifiers while still obtaining high accuracy due to the power of GBDT methods.

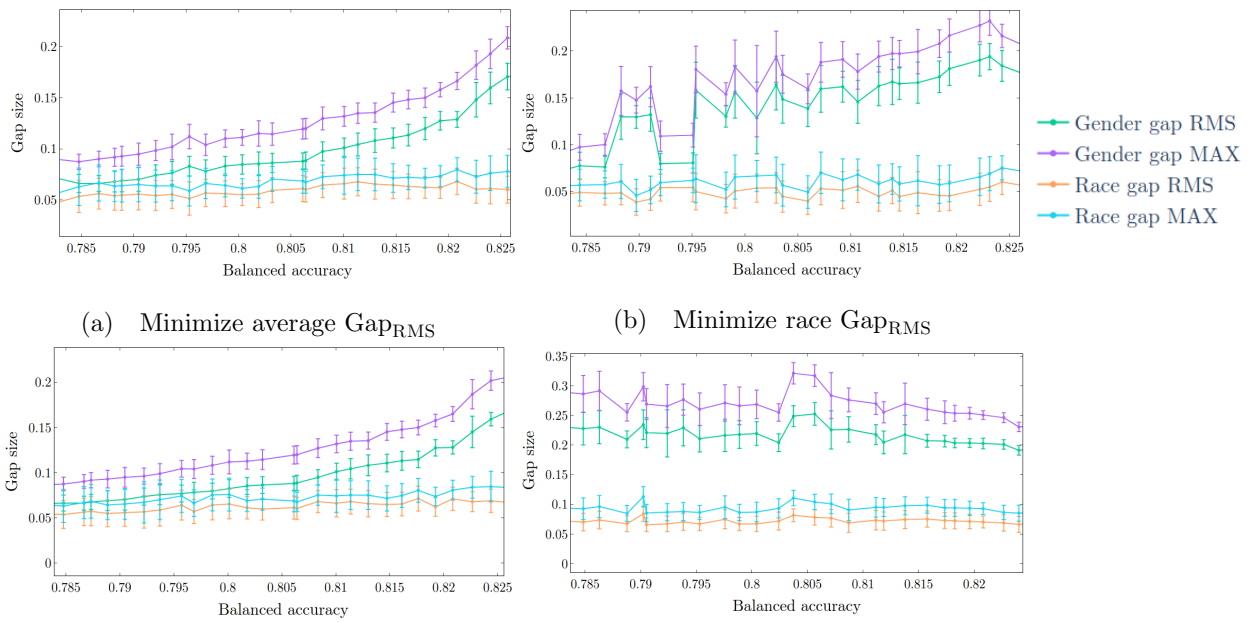
		Individual fairness		Gender gaps		Race gaps	
Method	BAcc	S-cons	GR-cons	GAP_{Max}	GAP_{RMS}	GAP_{Max}	GAP_{RMS}
BuDRO	.816	**.943**	.957	.146	.113	.084	.073
Baseline	**.844**	.942	.913	.200	.166	.098	.082
Project	.787	.881	**1**	**.079**	.069	.064	.050
Reweigh	.784	.853	.949	.131	.093	**.056**	**.043**
Baseline NN	.829	.848	.865	.216	.179	.105	.089
SenSR	.789	.934	.984	.087	**.068**	.067	.055
Adv. Deb.	.815	.807	.841	.110	.082	.078	.070

Table 5.5: Adult: average results over 10 splits into 80% training and 20% test data when methods are trained to maximize the balanced accuracy (BAcc). NN, SenSR and Adversarial Debiasing [ZLM18] numbers are from [YBS20].

		Individual fairness		Gender gaps		Race gaps	
Method	BAcc	S-cons	GR-cons	GAP_{Max}	GAP_{RMS}	GAP_{Max}	GAP_{RMS}
BuDRO	.0.005	0.012	0.006	0.013	0.013	0.019	0.018
Baseline	0.003	0.007	0.008	0.006	0.008	0.013	0.013
Project	0.005	0.079	0.000	0.022	0.018	0.021	0.016
Reweigh	0.005	0.010	0.009	0.021	0.015	0.031	0.022
Baseline NN	0.001	0.008	0.004	0.003	0.004	0.003	0.003
SenSR	0.003	0.012	0.000	0.005	0.004	0.004	0.003
Adv. Deb.	0.002	0.002	0.012	0.006	0.005	0.005	0.006

Table 5.6: Adult: Standard deviations of results reported in Table 5.5.

Trading optimal accuracy to improve the group fairness gaps. Varying hyperparameters reveals a trade-off between classifier accuracy and the group fairness metrics considered in this chapter when using the BuDRO method. This relationship is explored in Figures 5.2 and 5.3. The data in



(a) Minimize average Gap_{RMS}

(b) Minimize race Gap_{RMS}

(c) Minimize gender Gap_{RMS}

(d) Maximize S-cons

Figure 5.2: Fairness measures at given accuracy levels for the BuDRO method on the Adult data set, considered over 10 train/test splits. The results from each choice of hyperparameters are averaged over all train/test splits before being grouped into accuracy bins. The plotted points are the ones from each bin that optimize the specified quantity ((a) average Gap_{RMS}, (b) race Gap_{RMS}, (c) gender Gap_{RMS}, (d) spouse consistency). Error bars represent one standard deviation. Empirically, the gender gaps were harder to reduce than the race gaps. The S-cons in all of these pictures never drops below 94%.

Figure 5.2 was also used to guide hyperparameter selection for the BuDRO method on the Adult data set.

Before analyzing these figures, we describe how they were generated. We separate the horizontal accuracy axis into bins of a fixed width (here, the bin size is 0.0016). We then consider ten 80% train/20% test splits of the data set, and explore a grid of hyperparameters for each of these train/test splits. We examine a different grid of hyperparameters for each method; contact the author for specifics. In addition to the GBDT hyperparameters discussed above (`max_depth`, `lambda`, `min_weight`, `eta`, number of boosting steps), for BuDRO we also vary the perturbation budget ϵ and the SGD learning rate. The results from each choice of hyperparameters are averaged (over all ten train/test splits) and are placed in the bin that contains the average accuracy; the error bars in the figures correspond to the standard error from this averaging.

In Figure 5.2, we present four figures: each is obtained by determining the set of hyperparameters in each accuracy bin that optimize the average (over the 10 seeds) of a given fairness quantity (on the test set). Thus, each figure contains the data from approximately 30 hyperparameter settings (one for each point, different for each figure). For example, in Figure 5.2(c), each point was obtained

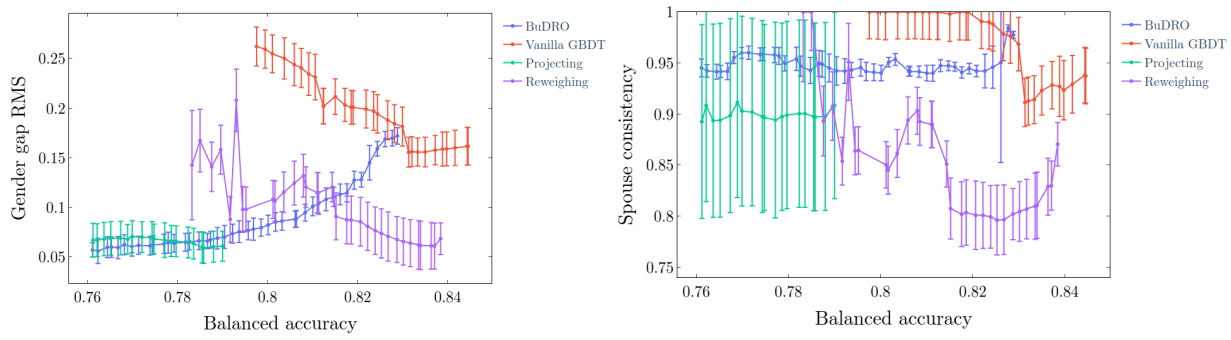
from the classifier constructed using the hyperparameters that minimize the gender GAP_{RMS} in the corresponding accuracy bin. In a similar fashion, Figure 5.2(a) is constructed by selecting the hyperparameters from each accuracy bin that minimize the average of the race GAP_{RMS} and the gender GAP_{RMS}. In all of the plots, the S-cons never drops below 94%, and the GR-cons remains similarly high; thus, we do not focus on the consistency measures here. To be clear, these plots reveal the relationship between model accuracy and the group fairness metrics; BuDRO can always produce classifiers with favorable individual fairness scores when ϵ is greater than 0 on the Adult data set.

In this chapter we concentrate on minimizing the gender GAP_{RMS} (Figure 5.2(c)) due to the fact that the gender gaps appear to be more difficult to shrink than the race gaps in Figure 5.2. In fact, as seen in Table 5.5, the baseline classier exhibits fairly small race gaps, even though it is trained to maximize accuracy with knowledge of the `race` feature. In real-world use it is not clear which fairness quantity a user would be required to optimize (or how to balance the group fairness gaps for different protected features); thus, Figure 5.2 presents a larger picture of the different trade-offs involved.

In Figures 5.2(a) and (c), we observe a trade-off between accuracy and group fairness gaps with the BuDRO method. That is, by decreasing accuracy (which generally corresponds to increasing the perturbation parameter ϵ), we are able to decrease the group fairness gaps, all at a high level of S-cons. Thus, it is possible to chose parameters to produce smaller group fairness gaps (with potentially lower accuracy) if required by the application.

The BuDRO data in Table 5.5 was collected from ten different (new) 80% train/20% test splits, using hyperparameters chosen by examining Figure 5.2(c). We were interested in finding a point with high accuracy and high spouse consistency; thus, we examined hyperparameters corresponding to the points in the accuracy range 0.81 to 0.825 and looked for patterns in these hyperparameters. We attempt this generalization to make hyperparameter selection slightly more realistic: it seems willfully ignorant to ignore the data in Figure 5.2 when selecting hyperparameters for our final tests, and it is at least plausible that a user could find some of these generally good hyperparameters via hand-tuning. This results in the set of BuDRO hyperparameters that were presented in Table 5.4. Contact the author for the code for more details about the selected hyperparameters.

To investigate the trade-off between accuracy and group fairness gaps, we followed the procedure discussed in the previous paragraph to select a different set of hyperparameters for the BuDRO method, aiming for smaller group fairness gaps (with lower accuracy). The obtained parameters are `max_depth`: 12, `lambda`: 10^{-6}, `min_weight`: $1/36177$, `eta`: 0.001, boosting for 130 steps, a perturbation budget ϵ of 1.4, and an SGD learning rate of 0.001. The results of running BuDRO on the Adult data set with these hyperparameters are shown in Table 5.7, along with the methods that produced the smallest group fairness gaps previously presented in Table 5.5. Running BuDRO



(a) (b)

Figure 5.3: The accuracy vs fairness trade-off of BuDRO when compared to other baseline boosting algorithms on the Adult data set. All lines are chosen to minimize gender $\mathrm{Gap}_{\mathrm{RMS}}$. (a) contains a comparison of the Gender gap RMS. The BuDRO line also appears in Figure 5.2(c). (b) contains a comparison of the S-cons. Error bars represent one standard deviation.

with this second set of parameters dramatically reduces the group fairness gaps so that they are competitive with the gaps produced by the other methods. Moreover, BuDRO maintains high S-cons and GR-cons scores.

		Individual fairness		Gender gaps		Race gaps	
Method	BAcc	S-cons	GR-cons	$\mathrm{GAP}_{\mathrm{Max}}$	$\mathrm{GAP}_{\mathrm{RMS}}$	$\mathrm{GAP}_{\mathrm{Max}}$	$\mathrm{GAP}_{\mathrm{RMS}}$
BuDRO ($\epsilon = 0.3$)	**.816**	.943	.957	.146	.113	.084	.073
BuDRO ($\epsilon = 1.4$)	.791	**.949**	.959	.095	.073	.068	.055
Project	.787	.881	**1**	**.079**	.069	.064	.050
Reweigh	.784	.853	.949	.131	.093	**.056**	**.043**
SenSR	.789	.934	.984	.087	**.068**	.067	.055

Table 5.7: Adult: average results over 10 splits into 80% training and 20% test data using a second set of hyperparameters discussed in the text. SenSR numbers are from [YBS20].

To further illustrate the trade-off between accuracy and fairness, Figure 5.3 includes a comparison of the BuDRO method to other baseline GDBT methods. Each point in Figure 5.3 is chosen to be the one from the accuracy bin that minimizes the gender $\mathrm{Gap}_{\mathrm{RMS}}$.

This figure is constructed specifically to explore how fair we can make a GBDT-based classifier at given (fixed) levels of accuracy. The vanilla GBDT method always produces high accuracy classifiers in the hyperparameter grid that we consider. This comes with large group fairness gaps that we are unable to decrease. On the other hand, the projecting method always produces good group fairness gaps, but we are unable to obtain high accuracy with this method. Finally, the reweighing method can obtain high accuracy with low fairness gaps, but it never produces an acceptable S-cons.

Figure 5.3 suggests that BuDRO is a better method than projecting at all accuracy levels on Adult: at any accuracy level, BuDRO matches the group fairness gaps produced by projecting while improving upon the (consistency of the) individual fairness metric. Projecting always produces a classifier with small group fairness gaps; unlike projecting, however, BuDRO can also be used to construct a more accurate classifier if we are allow larger group fairness gaps. Depending on the requirements of the application (e.g. as defined by law or other application specific fairness goals), this allows for more flexibility in the creation of individually fair and accurate classifiers.

It is also interesting to observe how the BuDRO curve in Figure 5.3 meets the curve for Vanilla GBDTs. Specifically, as the curves meet, the gender gap of the BuDRO method is increasing, while the gender gap of the Vanilla GBDT curve appears to be slightly decreasing. We speculate that this is due to the fact that the grid of hyperparameters used in the construction of Figure 5.3 does not contain values of ϵ smaller that 0.1. That is, the value of ϵ does not go down to 0, so we can not expect to precisely recover the baseline results. Essentially, we *force* a small perturbation in the data while also pushing for high accuracy. Thus, it appears that the high accuracy points in Figure 5.3 correspond to solutions that are obtained by improving race fairness without significantly improving gender fairness (see also Figure 5.2(c) - the race fairness always remains small for BuDRO). These high accuracy cases apparently find a perturbation that is mostly along the race axis.

A full analysis of the trade-off between group fairness and accuracy, including examining if such a trade-off can be found when using NN-based fairness methods, is left for future work.

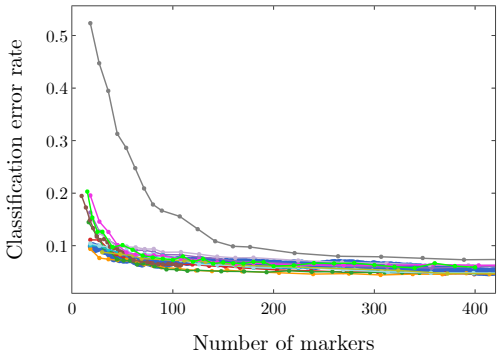
5.6 Conclusion

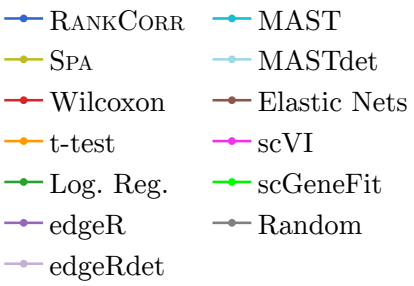
The primary focus of this chapter is deriving an individually fair method based on gradient boosting that also provides the benefits of using non-smooth models such as decision trees to a fairness setting. Our BuDRO method converges globally and generalizes. The experimental results demonstrate our method is empirically effective for learning a fair classifier.

APPENDIX A

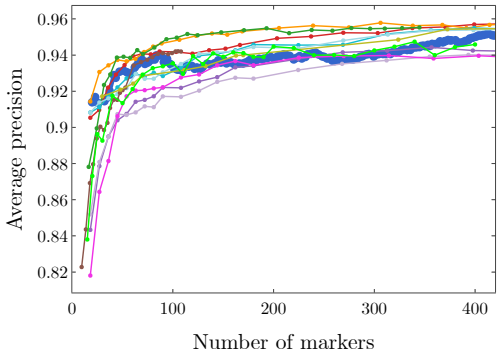
Supplementary figures

This Appendix contains pictures to supplement the content of Chapter 2.

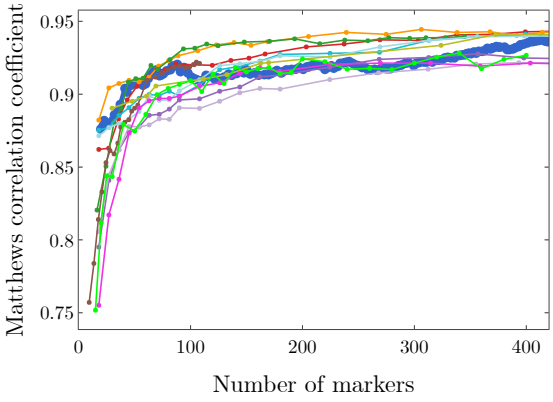




(a) Error rate of NCC with random markers

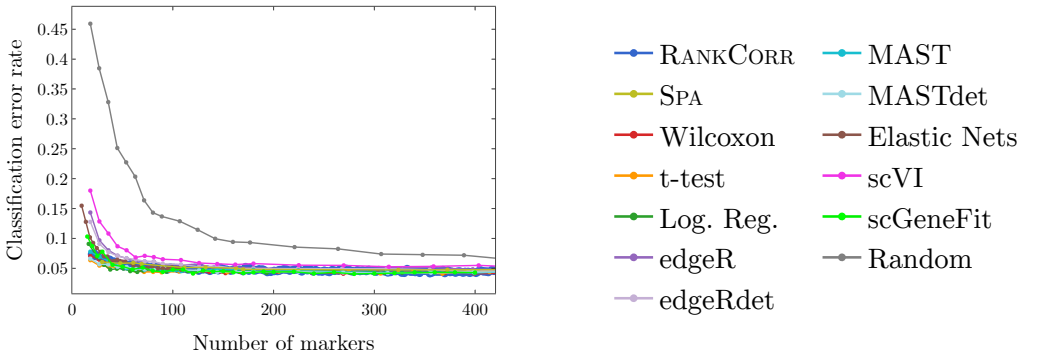


(b) Average precision of NCC

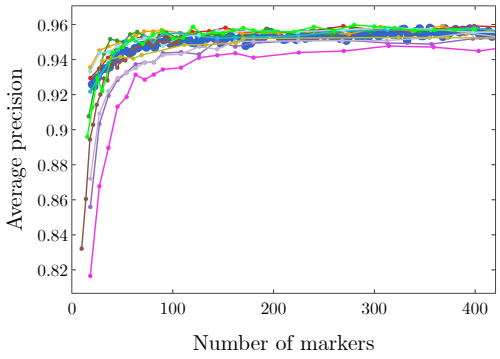


(c) Matthews correlation coefficient of NCC

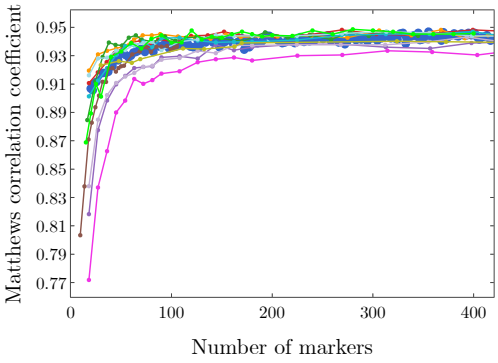
Figure A.1: Supervised classification metrics for the ZEISEL data set using the nearest centroid classifier (NCC). Included is the performance of the scGeneFit method. (a) is the classification error rate including a curve corresponding to random marker selection (the curve that starts at around 50% error). (b) contains the precision and (c) contains the Matthews correlation coefficient. Data from random marker selection are not included in figures (b) and (c) for clarity. Note that the curves in (b) are similar in shape to the curves in (c); they are also similar in shape to the classification accuracy ($1-$ classification error rate from Figure 4(a) of the main manuscript).



(a) Error rate of RFC with random markers

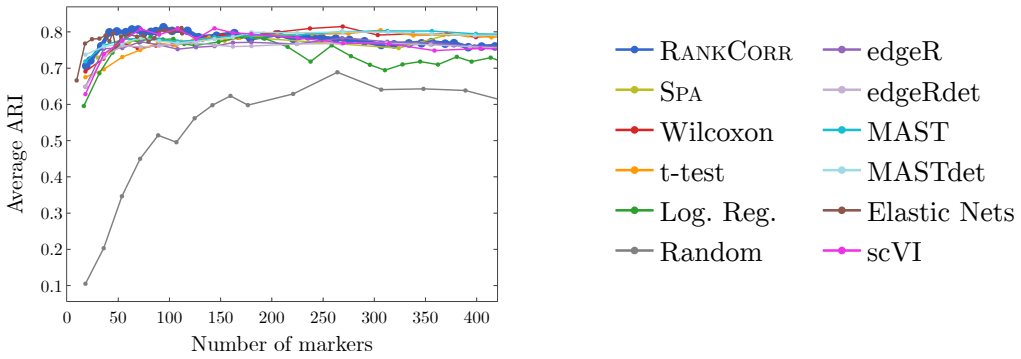


(b) Average precision of RFC

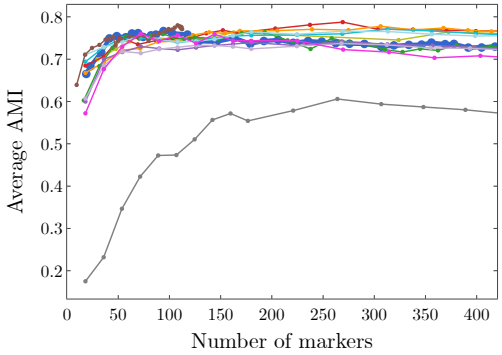


(c) Matthews correlation coefficient of RFC

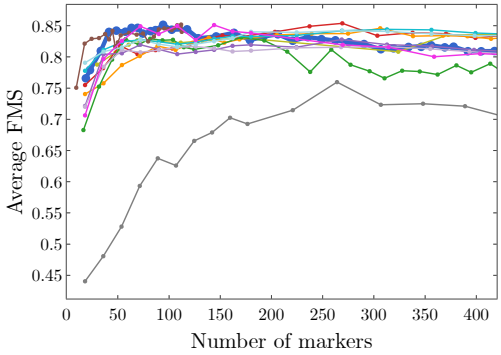
Figure A.2: Supervised classification metrics for the ZEISEL data set using the random forests classifier (RFC). Included is the performance of the scGeneFit method. (a) is the classification error rate including a curve corresponding to random marker selection (the curve that starts at around 45% error). (b) contains the precision and (c) contains the Matthews correlation coefficient. Data from random marker selection are not included in figures (b) and (c) for clarity. Note that the curves in (b) are similar in shape to the curves in (c); they are also similar in shape to the classification accuracy (1− classification error rate from Figure 4(c) of the main manuscript).



(a) Average ARI scores

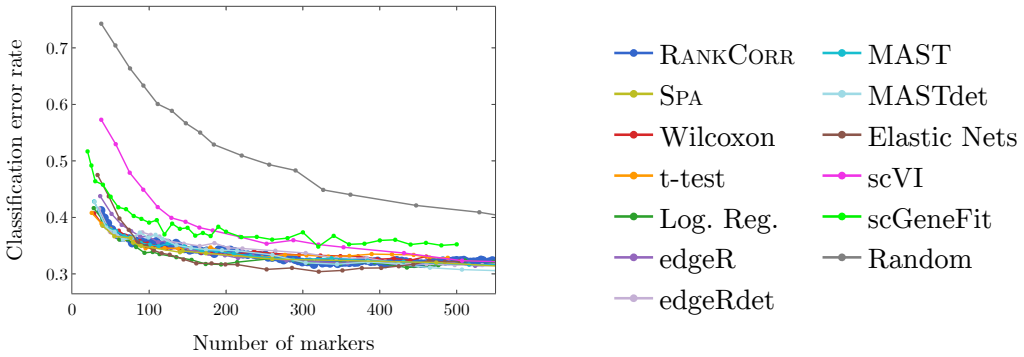


(b) Average AMI scores

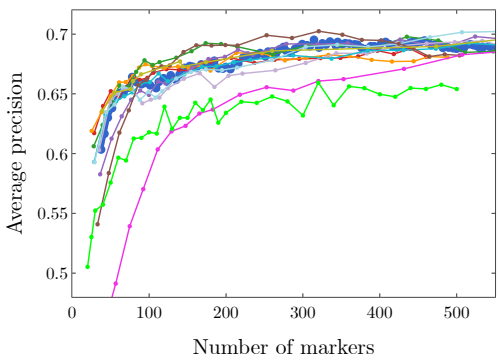


(c) Average Fowlkes-Mallows scores

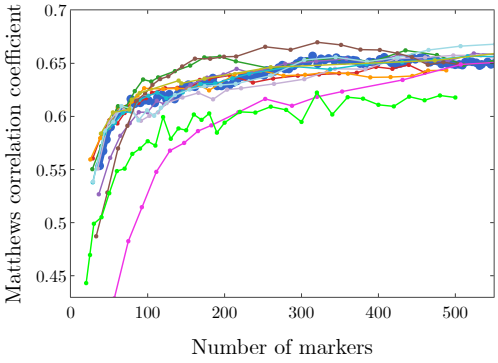
Figure A.3: Unsupervised clustering metrics for the ZEISEL data set including data from random marker selection. The curve corresponding to random marker selection is shown in grey and is the lowest (worst) curve in all three plots. The ARI score is shown in (a), the AMI score is shown in (b), and the Fowlkes-Mallows score is shown in (c). The clustering is carried out using 5-fold cross validation and scores are averaged across folds.



(a) Error rate of NCC with random markers

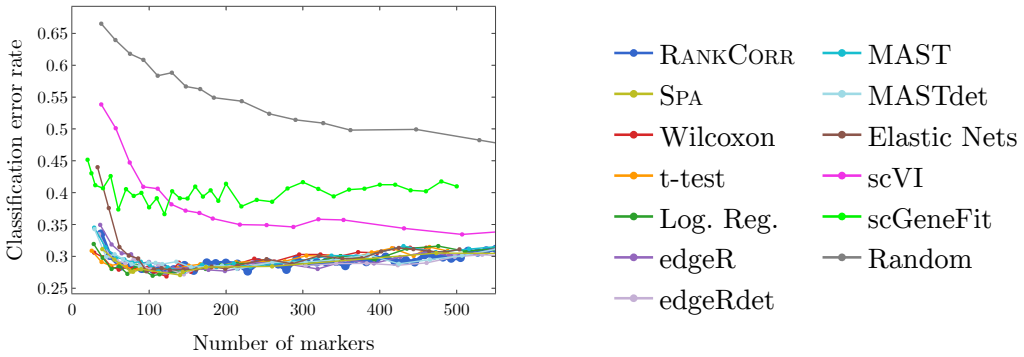


(b) Precision of NCC

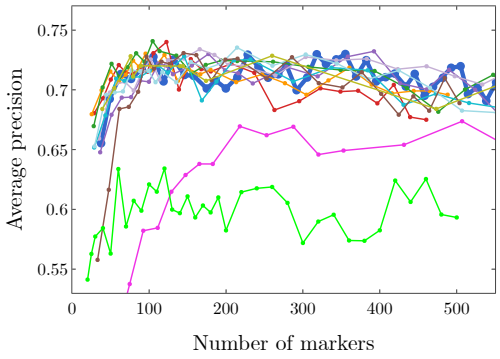


(c) Matthews correlation coefficient of NCC

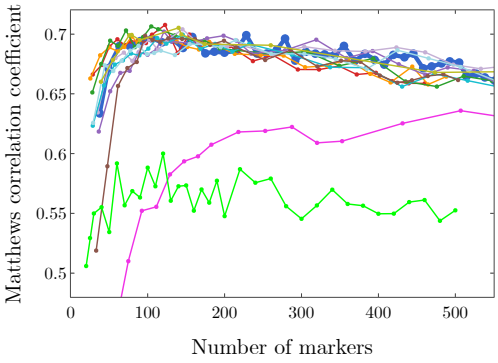
Figure A.4: Supervised classification metrics for the PAUL data set using the nearest centroid classifier (NCC). Included is the performance of the scGeneFit method. (a) is the classification error rate including a curve corresponding to random marker selection (the curve that starts at around 75% error). (b) contains the precision and (c) contains the Matthews correlation coefficient. Data from random marker selection are not included in figures (b) and (c) for clarity. Note that the curves in (b) are similar in shape to the curves in (c); they are also similar in shape to the classification accuracy (1− classification error rate from Figure 6(a) of the main manuscript).



(a) Error rate of RFC with random markers

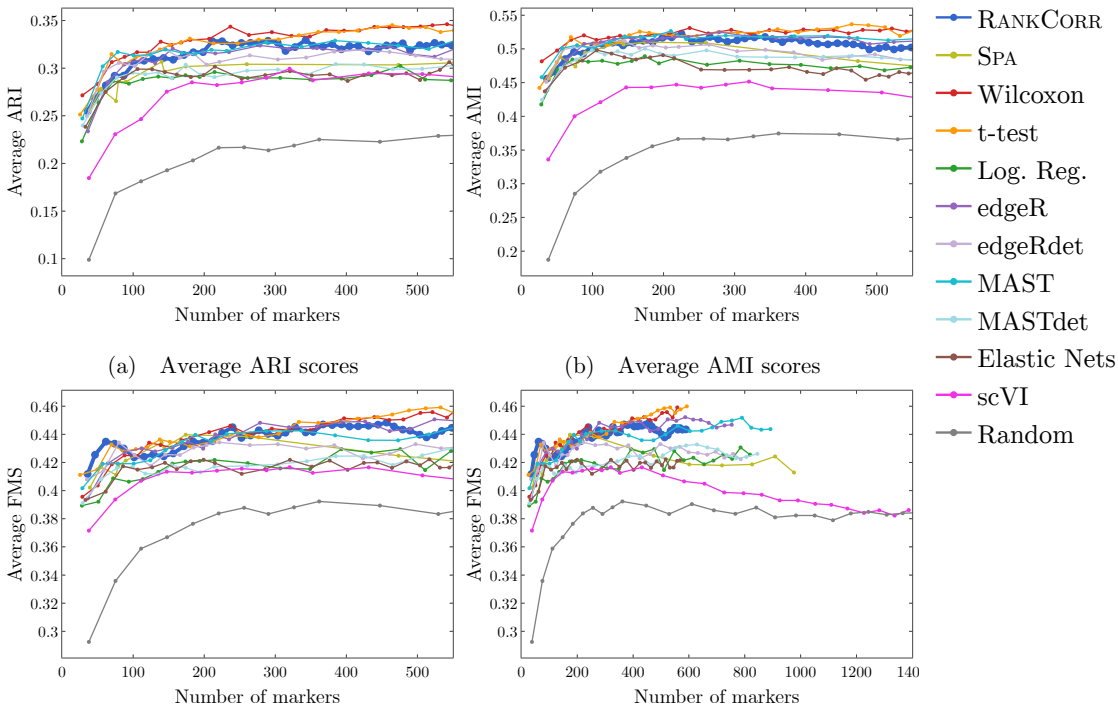


(b) Precision of RFC



(c) Matthews correlation coefficient of RFC

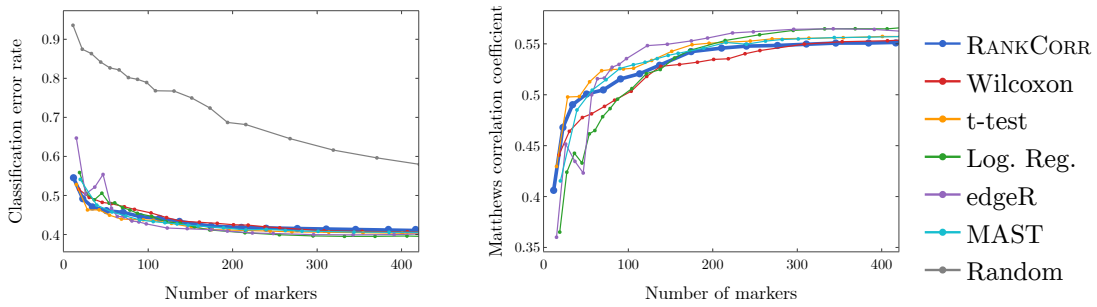
Figure A.5: Supervised classification metrics for the PAUL data set using the random forests classifier (RFC). Included is the performance of the scGeneFit method. (a) is the classification error rate including a curve corresponding to random marker selection (the curve that starts at around 65% error). (b) contains the precision and (c) contains the Matthews correlation coefficient. Data from random marker selection are not included in figures (b) and (c) for clarity. Note that the curves in (b) are similar in shape to the curves in (c); they are also similar in shape to the classification accuracy (1− classification error rate from Figure 6(c) of the main manuscript).



(a) Average ARI scores (b) Average AMI scores

(c) Average Fowlkes-Mallows scores (d) Tail behavior of Fowlkes-Mallows scores

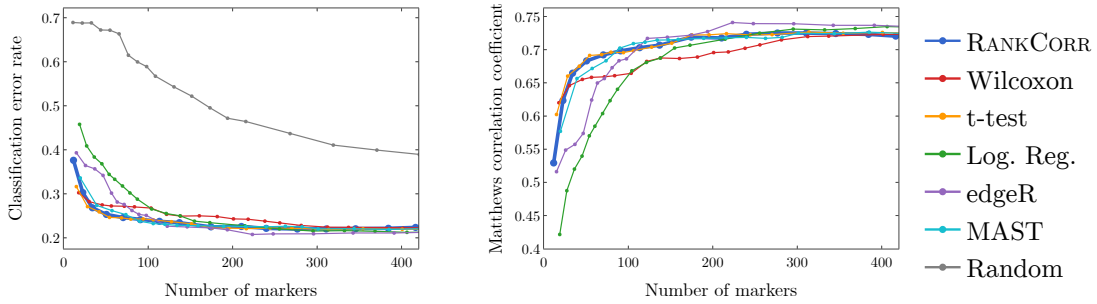
Figure A.6: Unsupervised clustering metrics for the Paul data set including data from random marker selection. The curve corresponding to random marker selection is shown in grey and is the lowest (worst) curve in all four plots. The ARI score is shown in (a), the AMI score is shown in (b), and the Fowlkes-Mallows score is shown in (c). (d) contains the FM scores for larger numbers of markers selected, showing the scVI does approach the behavior of random marker selection in this case. Each clustering is carried out using 5-fold cross validation and scores are averaged across folds.



(a) Error rate of NCC with random markers

(b) Matthews correlation coefficient of NCC

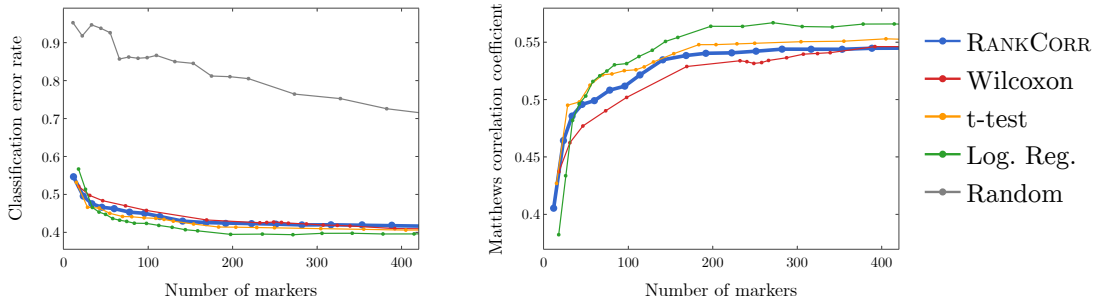
Figure A.7: Supervised classification metrics for the ZhengFilt data set using the nearest centroid classifier (NCC). (a) is the classification error rate including a curve corresponding to random marker selection (the curve that starts at around 90% error). (b) contains the Matthews correlation coefficient. Data from random marker selection are not included in (b). Note that (b) is similar in shape to the classification accuracy (1− classification error rate from Figure 8(a) of the main manuscript).



(a) Error rate of RFC with random markers

(b) Matthews correlation coefficient of RFC

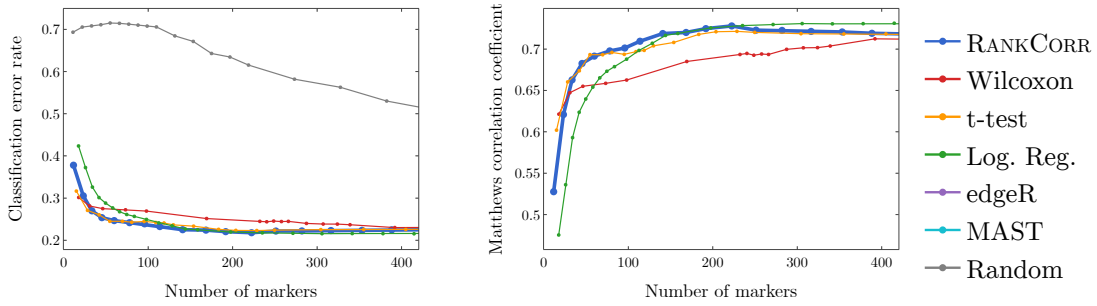
Figure A.8: Supervised classification metrics for the ZhengFilt data set using the random forests classifier (RFC). (a) is the classification error rate including a curve corresponding to random marker selection (the curve that starts at around 70% error). (b) contains the Matthews correlation coefficient. Data from random marker selection are not included in (b). Note that (b) is similar in shape to the classification accuracy (1− classification error rate from Figure 9(a) of the main manuscript).



(a) Error rate of NCC with random markers

(b) Matthews correlation coefficient of NCC

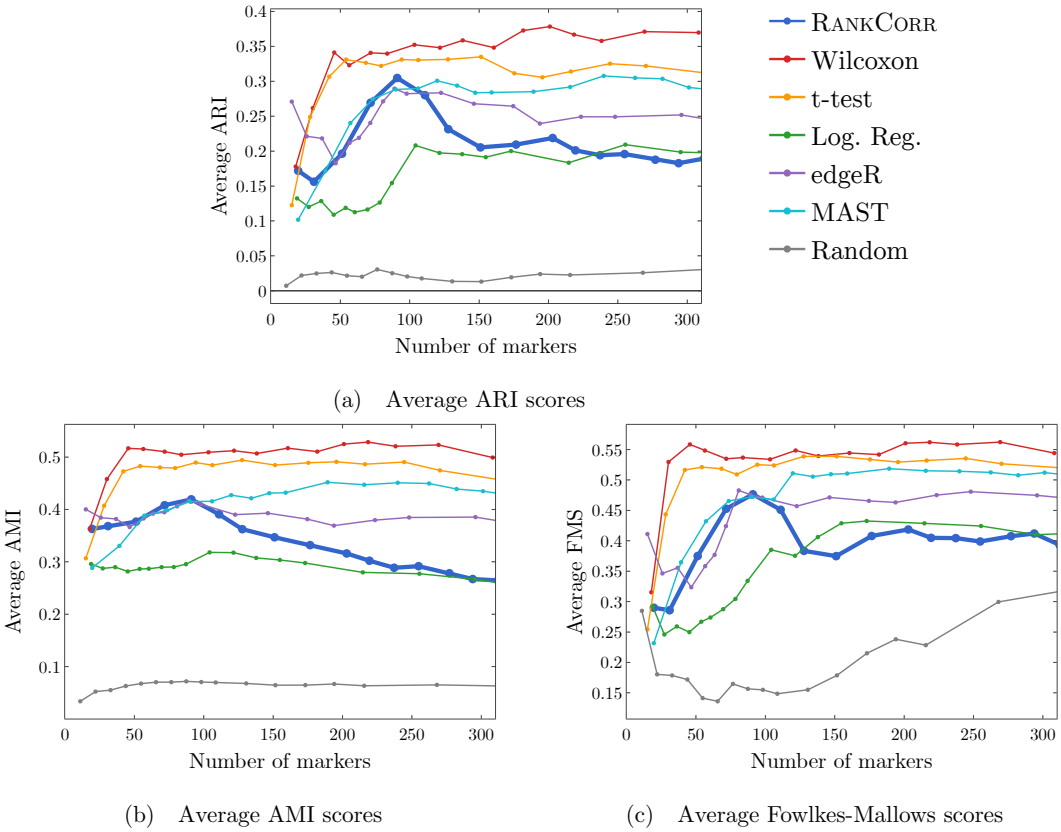
Figure A.9: Supervised classification metrics for the ZhengFull data set using the nearest centroid classifier (NCC). (a) is the classification error rate including a curve corresponding to random marker selection (the curve that starts at around 90% error). (b) contains the Matthews correlation coefficient. Data from random marker selection are not included in (b). Note that (b) is similar in shape to the classification accuracy (1− classification error rate from Figure 8(c) of the main manuscript).



(a) Error rate of RFC with random markers

(b) Matthews correlation coefficient of RFC

Figure A.10: Supervised classification metrics for the ZhengFull data set using the random forests classifier (RFC). (a) is the classification error rate including a curve corresponding to random marker selection (the curve that starts at around 70% error). (b) contains the Matthews correlation coefficient. Data from random marker selection are not included in (b). Note that (b) is similar in shape to the classification accuracy (1− classification error rate from Figure 9(c) of the main manuscript).

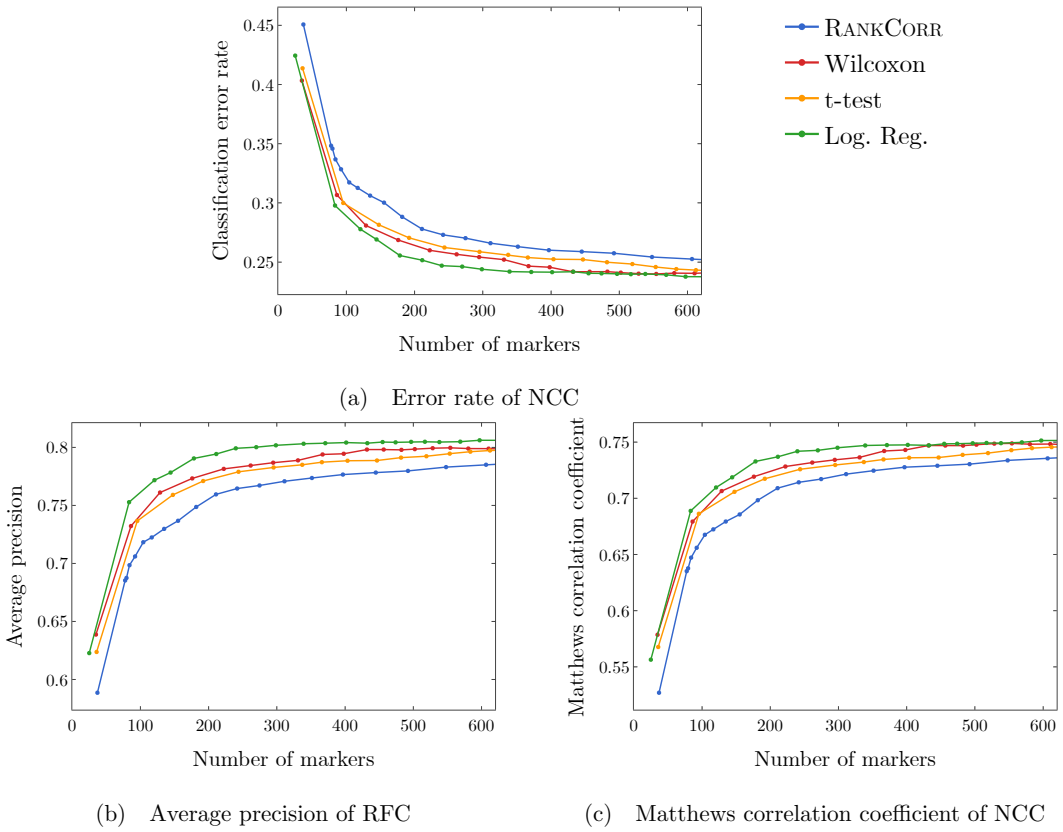


(a) Average ARI scores

(b) Average AMI scores

(c) Average Fowlkes-Mallows scores

Figure A.11: Unsupervised clustering metrics for the ZhengFilt data set including data from random marker selection. The curve corresponding to random marker selection is shown in grey and is the lowest (worst) curve in all three plots. The ARI score is shown in (a), the AMI score is shown in (b), and the Fowlkes-Mallows score is shown in (c). Each clustering is carried out using 5-fold cross validation and scores are averaged across folds.



(a) Error rate of NCC

(b) Average precision of RFC

(c) Matthews correlation coefficient of NCC

Figure A.12: Supervised classification metrics for the 10xMouse data set using the nearest centroid classifier (NCC). (a) is the classification error rate figure found in Figure 9 of the main manuscript, for reference here. (b) contains the average precision curves and (c) contains the Matthews correlation coefficient curves. We do not compare to random markers on this data set.

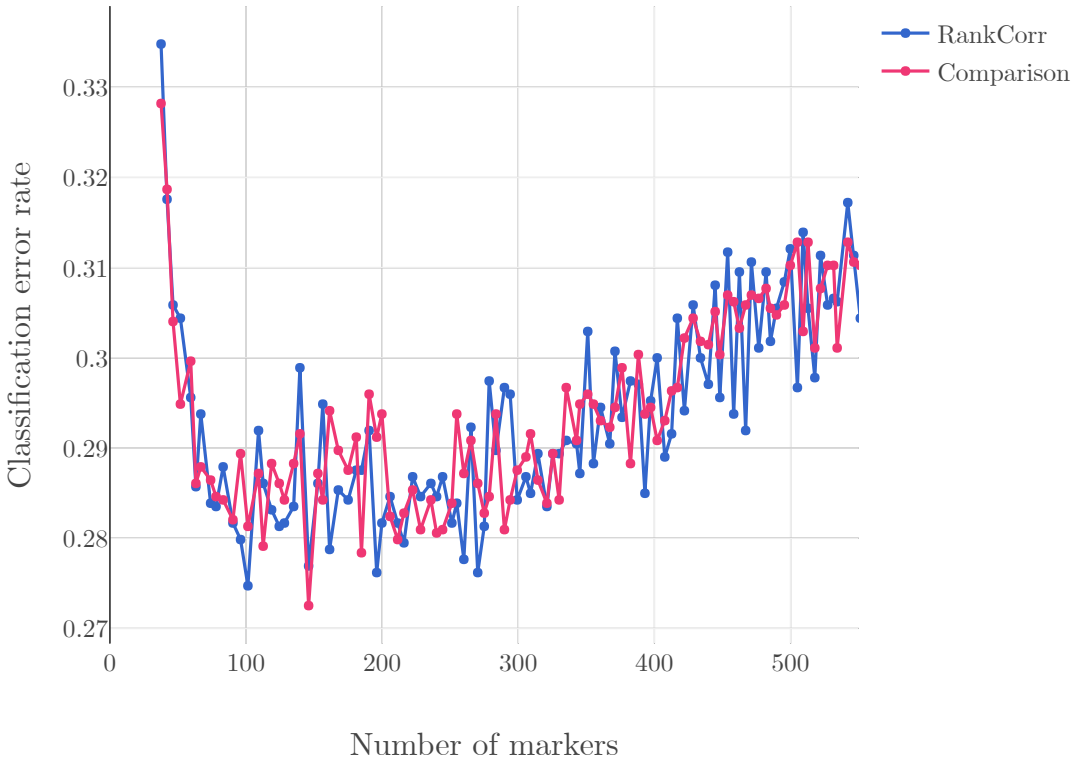


Figure A.13: The classification accuracy under the RFC on the PAUL data set (see the Methods in the main manuscript) run twice with the same markers used for each point. Significant variation is observed in the classification accuracy over the two classification attempts. Differences of nearly 2% are observed between the two curves.

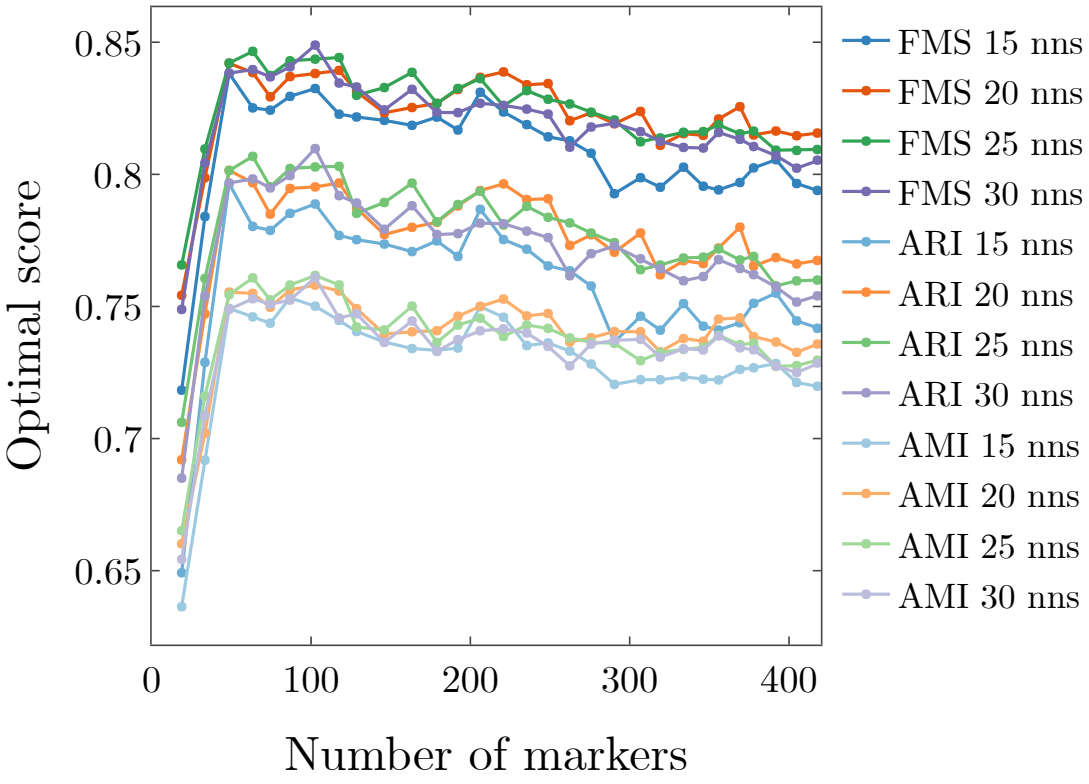


Figure A.14: Effect of changing the number of nearest neighbors on the ARI, AMI, and FM scores for the ZEISEL data set using RANKCORR to select markers. Clustering was performed with Louvain and the scores were optimized over the resolution. It appears that 15 nearest neighbors is too few, while 30 nearest neighbors is too many.

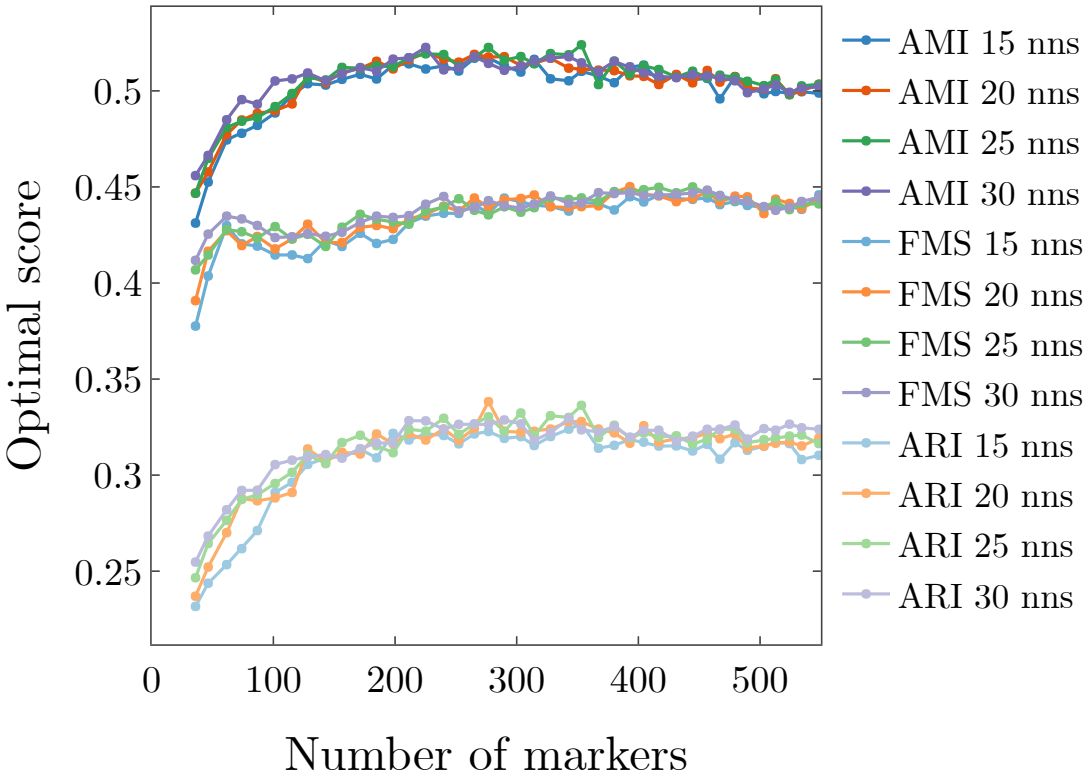
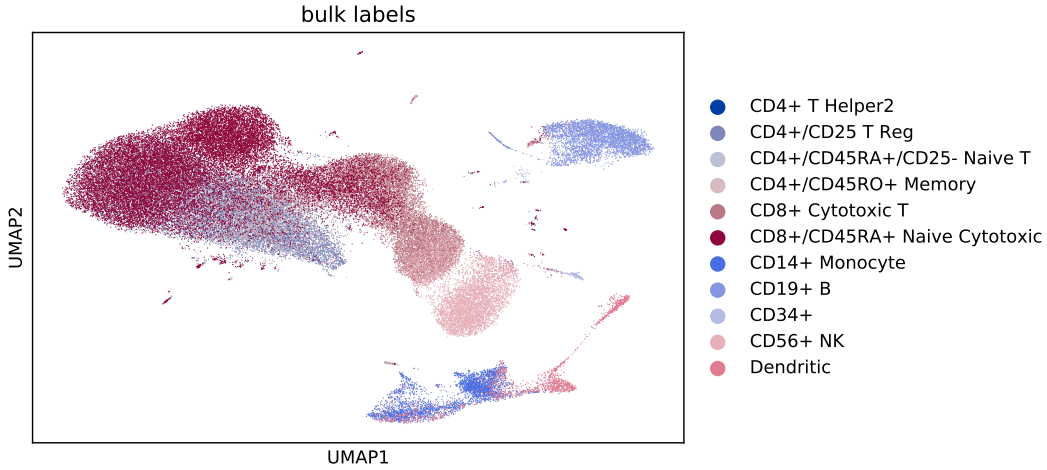
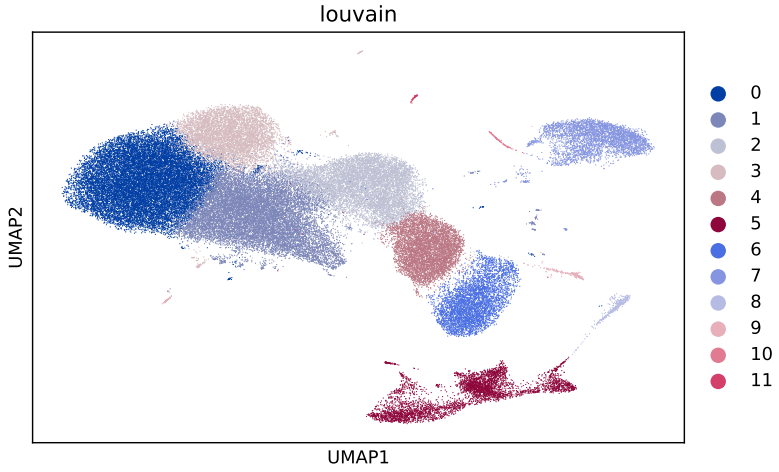


Figure A.15: Effect of changing the number of nearest neighbors on the ARI, AMI, and FM scores for the PAUL data set using RANKCORR to select markers. Clustering was performed with Louvain and the scores were optimized over the resolution. All of the choices of numbers of nearest neighbors produce similar curves for all three scores. Choosing 30 nearest neighbors appears to provide increased performance for small numbers of markers.



(a)



(b)

Figure A.16: Clustering the 68k PBMC data set from [ZTB+17] (reference [2] from the main manuscript) with Louvain clustering. (a) contains a UMAP plot of the bulk labels. (b) is a UMAP plot of a Louvain clustering of the data set. It was created by first filtering to the 1000 most variable genes using the `cell_ranger` flavor of the `filter_genes_dispersion` function in the `scanpy` python package. The Louvain algorithm was run on the top 50 PCs and used 25 nearest neighbours for each cell with a resolution parameter of 0.3. The Louvain clustering solution subjectively looks similar to the bulk labels. The ARI for the clustering compared to the bulk labels is 0.345, the AMI is 0.565, and the FMS is 0.462 (these values have been rounded to 3 significant digits).

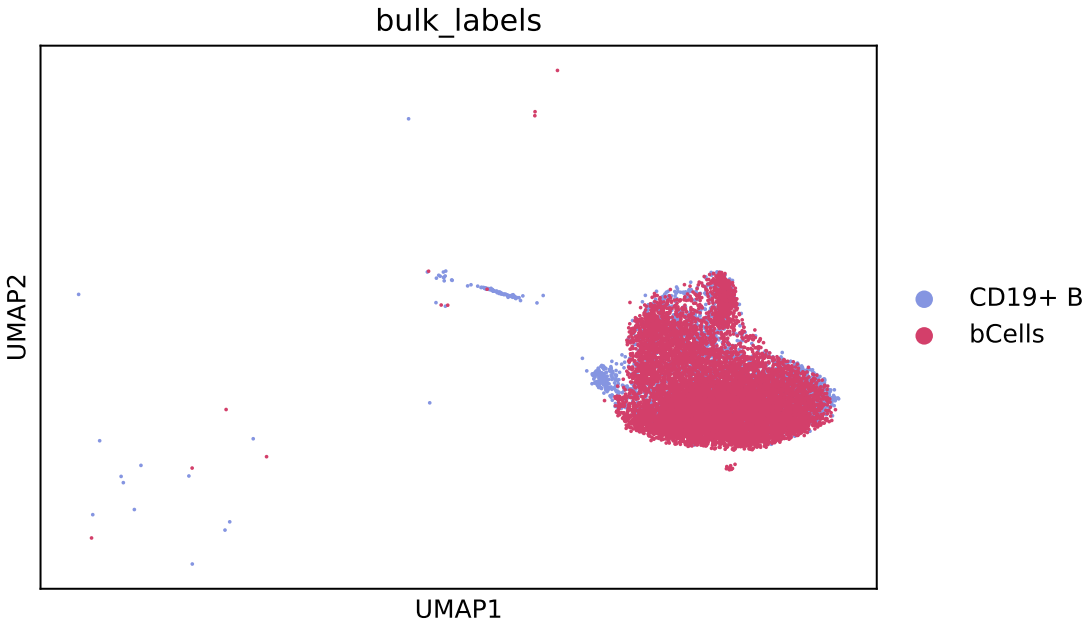


Figure A.17: UMAP projection of the data consisting of ZHENGFULL combined with the isolated CD19+ B cell data set from [ZTB+17] (reference [2] from the main manuscript) that was used to estimate parameters in Splatter simulations for generating synthetic data. We show only the isolated CD19+ sample (labeled "bCells") and the cluster of B cells from ZHENGFULL. The overlap between the two clusters is quite good.

BIBLIOGRAPHY

[ACN08] Nir Ailon, Moses Charikar, and Alantha Newman. Aggregating inconsistent information. *Journal of the ACM*, 55(5):1–27, October 2008.

[Ail08] Nir Ailon. Aggregation of partial rankings, p-ratings and top-m lists. *Algorithmica*, 57(2):284–300, jul 2008.

[AK98] Edoardo Amaldi and Viggo Kann. On the approximability of minimizing nonzero variables or unsatisfied relations in linear systems. *Theoretical Computer Science*, 209(1):237 – 260, 1998.

[ALC18] Ángeles Arzalluz-Luque and Ana Conesa. Single-cell RNAseq for the study of isoforms—how is that possible? *Genome Biology*, 19(1), August 2018.

[ALMK16] Julia Angwin, Jeff Larson, Surya Mattu, and Lauren Kirchner. Machine bias. *ProPublica*, 2016.

[ALPV14] Albert Ai, Alex Lapanowski, Yaniv Plan, and Roman Vershynin. One-bit compressed sensing with non-gaussian measurements. *Linear Algebra and its Applications*, 441:222 – 239, 2014. Special Issue on Sparse Approximate Solution of Linear Systems.

[AR56] Theodore W Anderson and Herman Rubin. Statistical inference in factor analysis. In *Proceedings of the third Berkeley symposium on mathematical statistics and probability*, volume 5, pages 111–150, 1956.

[Bai98] Ralph W. Bailey. The number of weak orderings of a finite set. *Soc. Choice Welf.*, 15(4):559–562, 1998.

[BCZ+16] Tolga Bolukbasi, Kai-Wei Chang, James Y Zou, Venkatesh Saligrama, and Adam T Kalai. Man is to computer programmer as woman is to homemaker? debiasing word embeddings. In *Advances in Neural Information Processing Systems*, pages 4349–4357, 2016.

[BdDW+12] Aharon Ben-Tal, Dick den Hertog, Anja De Waegenaere, Bertrand Melenberg, and Gijs Rennen. Robust Solutions of Optimization Problems Affected by Uncertain Probabilities. *Management Science*, 59(2):341–357, November 2012.

[BG18] Joy Buolamwini and Timnit Gebru. Gender Shades: Intersectional Accuracy Disparities in Commercial Gender Classification. In *Proceedings of Machine Learning Research*, volume 87, pages 77–91, 2018.

[BGE13] William C. Bunting, L Cano Garcia, and Ezekiel Edwards. The war on marijuana in black and white. 2013.

[BHS+18] Andrew Butler, Paul Hoffman, Peter Smibert, Efthymia Papalexi, and Rahul Satija. Integrating single-cell transcriptomic data across different conditions, technologies, and species. *Nature Biotechnology*, 36(5):411–420, April 2018.

[BKM16] Jose Blanchet, Yang Kang, and Karthyek Murthy. Robust Wasserstein Profile Inference and Applications to Machine Learning. *arXiv:1610.05627 [math, stat]*, October 2016.

[BNSV18] Amanda Bower, Laura Niss, Yuekai Sun, and Alexander Vargo. Debiasing representations by removing unwanted variation due to protected attributes. *ArXiv*, abs/1807.00461, 2018.

[BPZL12] Gavin Brown, Adam Pocock, Ming-Jie Zhao, and Mikel Luján. Conditional likelihood maximisation: A unifying framework for information theoretic feature selection. *J. Mach. Learn. Res.*, 13:27–66, January 2012.

[Bra05] Richard C. Bradley. Basic properties of strong mixing conditions. a survey and some open questions. *Probability Surveys*, 2(0):107–144, 2005.

[Bra16] Richard C. Bradley. Strong mixing conditions. `https://encyclopediaofmath.org/wiki/Strong_mixing_conditions`, 2016. Accessed: 2020-07-30.

[Car09] Ben Carterette. On rank correlation and the distance between rankings. In *Proceedings of the 32nd international ACM SIGIR conference on Research and development in information retrieval - SIGIR '09*. ACM Press, 2009.

[CBN17] Aylin Caliskan, Joanna J. Bryson, and Arvind Narayanan. Semantics derived automatically from language corpora contain human-like biases. *Science*, 356(6334):183–186, 2017.

[CG16] Tianqi Chen and Carlos Guestrin. Xgboost: A scalable tree boosting system. *ArXiv*, abs/1603.02754, 2016.

[CGC+17] T. Conrad, M. Genzel, N. Cvetkovic, N. Wulkow, A. Leichtle, J. Vybiral, G. Kutyniok, and C. Schütte. Sparse proteomics analysis – a compressed sensing-based approach for feature selection and classification of high-dimensional proteomics mass spectrometry data. *BMC Bioinformatics*, 18, 2017.

[Cho17] Alexandra Chouldechova. Fair prediction with disparate impact: A study of bias in recidivism prediction instruments. *arXiv:1703.00056 [cs, stat]*, February 2017.

[CMJ+19] Elliot Creager, David Madras, Jörn-Henrik Jacobsen, Marissa A. Weis, Kevin Swersky, Toniann Pitassi, and Richard S. Zemel. Flexibly fair representation learning by disentanglement. *ArXiv*, abs/1906.02589, 2019.

[CR18] Alexandra Chouldechova and Aaron Roth. The frontiers of fairness in machine learning. *arXiv preprint arXiv:1810.08810*, 2018.

[CSG+19] Helena L. Crowell, Charlotte Soneson, Pierre-Luc Germain, Daniela Calini, Ludovic Collin, Catarina Raposo, Dheeraj Malhotra, and Mark D. Robinson. On the discovery of population-specific state transitions from multi-sample multi-condition single-cell rna sequencing data. *bioRxiv*, 2019.

[CSS07] Xin Chen, Melvyn Sim, and Peng Sun. A Robust Optimization Perspective on Stochastic Programming. *Operations Research*, 55:1058–1071, 2007.

[CW16] Nicholas Carlini and David Wagner. Towards Evaluating the Robustness of Neural Networks. August 2016.

[CZBH19] Hongge Chen, Huan Zhang, Duane S. Boning, and Cho-Jui Hsieh. Robust decision trees against adversarial examples. *ArXiv*, abs/1902.10660, 2019.

[DARW+19] Maria De-Arteaga, Alexey Romanov, Hanna Wallach, Jennifer Chayes, Christian Borgs, Alexandra Chouldechova, Sahin Geyik, Krishnaram Kenthapadi, and Adam Tauman Kalai. Bias in bios: A case study of semantic representation bias in a high-stakes setting. In *FAT* '19: Conference on Fairness, Accountability, and Transparency*, January 2019.

[Das18] Jeffrey Dastin. Amazon scraps secret AI recruiting tool that showed bias against women. *Reuters*, October 2018.

[DG77] Persi Diaconis and R. L. Graham. Spearman's footrule as a measure of disarray. *Journal of the Royal Statistical Society: Series B (Methodological)*, 39(2):262–268, 1977.

[DG17] Dheeru Dua and Casey Graff. UCI machine learning repository. `http://archive.ics.uci.edu/ml`, 2017.

[DGN16] John Duchi, Peter Glynn, and Hongseok Namkoong. Statistics of Robust Optimization: A Generalized Empirical Likelihood Approach. *arXiv:1610.03425 [stat]*, October 2016.

[DHP+12] Cynthia Dwork, Moritz Hardt, Toniann Pitassi, Omer Reingold, and Richard Zemel. Fairness through awareness. In *Proceedings of the 3rd innovations in theoretical computer science conference*, pages 214–226. ACM, 2012.

[DK11] Abhimanyu Das and David Kempe. Submodular meets spectral: Greedy algorithms for subset selection, sparse approximation and dictionary selection. In *Proceedings of the 28th International Conference on International Conference on Machine Learning*, ICML'11, pages 1057–1064, USA, 2011. Omnipress.

[DN16] John Duchi and Hongseok Namkoong. Variance-based regularization with convex objectives. *Journal of Machine Learning Research*, October 2016.

[DN18] John Duchi and Hongseok Namkoong. Learning Models with Uniform Performance via Distributionally Robust Optimization. *arXiv:1810.08750 [cs, stat]*, October 2018.

[Dro17] Christoph Droösser. In Order Not to Discriminate, We Might Have to Discriminate. `https://simons.berkeley.edu/news/algorithms-discrimination`, 2017. Accessed: 2020-07-30.

[DSC+19] Conor Delaney, Alexandra Schnell, Louis V Cammarata, Aaron Yao-Smith, Aviv Regev, Vijay K Kuchroo, and Meromit Singer. Combinatorial prediction of marker panels from single-cell transcriptomic data. *Molecular Systems Biology*, 15(10):e9005, 2019.

[DVME19] Bianca Dumitrascu, Soledad Villar, Dustin G. Mixon, and Barbara E. Engelhardt. Optimal marker gene selection for cell type discrimination in single cell analyses. *bioRxiv*, 2019.

[DY10] Erick Delage and Yinyu Ye. Distributionally Robust Optimization Under Moment Uncertainty with Application to Data-Driven Problems. *Operations Research*, 58:595–612, 2010.

[EK15] Peyman Mohajerin Esfahani and Daniel Kuhn. Data-driven Distributionally Robust Optimization Using the Wasserstein Metric: Performance Guarantees and Tractable Reformulations. *Mathematical Programming*, May 2015.

[EM02] Edward J. Emond and David W. Mason. A new rank correlation coefficient with application to the consensus ranking problem. *Journal of Multi-Criteria Decision Analysis*, 11(1):17–28, 2002.

[FFM+15] Michael Feldman, Sorelle A Friedler, John Moeller, Carlos Scheidegger, and Suresh Venkatasubramanian. Certifying and removing disparate impact. In *Proceedings of the 21th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, pages 259–268. ACM, 2015.

[FMY+15] Greg Finak, Andrew McDavid, Masanao Yajima, Jingyuan Deng, Vivian Gersuk, Alex K. Shalek, Chloe K. Slichter, Hannah W. Miller, M. Juliana McElrath, Martin Prlic, Peter S. Linsley, and Raphael Gottardo. MAST: a flexible statistical framework for assessing transcriptional changes and characterizing heterogeneity in single-cell RNA sequencing data. *Genome Biology*, 16(1), dec 2015.

[FSA19] Cornelia Fuetterer, Georg Schollmeyer, and Thomas Augustin. Constructing simulation data with dependency structure for unreliable single-cell rna-sequencing data using copulas. In Jasper De Bock, Cassio P. de Campos, Gert de Cooman, Erik Quaeghebeur, and Gregory Wheeler, editors, *Proceedings of the Eleventh International Symposium on Imprecise Probabilities: Theories and Applications*, volume 103 of *Proceedings of Machine Learning Research*, pages 216–224, Thagaste, Ghent, Belgium, 03–06 Jul 2019. PMLR.

[FSV16] Sorelle A. Friedler, Carlos Scheidegger, and Suresh Venkatasubramanian. On the (im)possibility of fairness. *CoRR*, abs/1609.07236, 2016.

[GbJS13] Johann A. Gagnon-bartsch, Laurent Jacob, and Terence P. Speed. Removing unwanted variation from high dimensional data with negative controls. Technical report, UC Berkeley, 2013.

[GCPB16] Aude Genevay, Marco Cuturi, Gabriel Peyré, and Francis Bach. Stochastic Optimization for Large-scale Optimal Transport. In NIPS, editor, *NIPS 2016 - Thirtieth Annual Conference on Neural Information Processing System*, Proc. NIPS 2016, Barcelona, Spain, December 2016.

[Gen15] M. Genzel. Sparse proteomics analysis: Toward a mathematical foundation of feature selection and disease classification. Master's thesis, Technische Universität Berlin, Berlin, Germany, 2015.

[GJKR18] Stephen Gillen, Christopher Jung, Michael Kearns, and Aaron Roth. Online learning with an unknown fairness metric. In *NeurIPS*, 2018.

[GKvO14] Dominic Grün, Lennart Kester, and Alexander van Oudenaarden. Validation of noise models for single-cell transcriptomics. *Nature Methods*, 11(6):637–640, April 2014.

[GLPR11] José Luis García-Lapresta and David Pérez-Román. *Consensual Processes*, chapter Measuring Consensus in Weak Orders, pages 213–234. Springer Berlin Heidelberg, Berlin, Heidelberg, 2011.

[GMM+18] Christopher Daniel Green, Qianyi Ma, Gabriel L. Manske, Adrienne Niederriter Shami, Xianing Zheng, Simone Marini, Lindsay Moritz, Caleb Sultan, Stephen J. Gurczynski, Bethany B. Moore, Michelle D. Tallquist, Jun Z. Li, and Saher Sue Hammoud. A comprehensive roadmap of murine spermatogenesis defined by single-cell RNA-seq. *Developmental Cell*, 46(5):651–667.e10, September 2018.

[Goo75] I. J. Good. The number of orderings of n candidates when ties are permitted. *Fibonacci Quart.*, 13:11–18, 1975.

[GPL+18] Sahaj Garg, Vincent Perot, Nicole Limtiaco, Ankur Taly, Ed Huai hsin Chi, and Alex Beutel. Counterfactual fairness in text classification through robustness. In *AIES '19*, 2018.

[GS10] Joel Goh and Melvyn Sim. Distributionally Robust Optimization and Its Tractable Approximations. *Operations Research*, 58(4-part-1):902–917, August 2010.

[GSS14] Ian J. Goodfellow, Jonathon Shlens, and Christian Szegedy. Explaining and Harnessing Adversarial Examples. *arXiv preprint arXiv:1412.6572*, December 2014.

[GWP+15] Minzhe Guo, Hui Wang, S. Steven Potter, Jeffrey A. Whitsett, and Yan Xu. SINCERA: A pipeline for single-cell RNA-seq profiling analysis. *PLOS Computational Biology*, 11(11):e1004575, November 2015.

[HKRR17] Úrsula Hébert-Johnson, Michael P. Kim, Omer Reingold, and Guy N. Rothblum. Calibration for the (Computationally-Identifiable) Masses. *arXiv e-prints*, page arXiv:1711.08513, Nov 2017.

[HP36] Harold Hotelling and Margaret Richards Pabst. Rank correlation and tests of significance involving no assumption of normality. *The Annals of Mathematical Statistics*, 7(1):29–43, March 1936.

[HPS+16] Moritz Hardt, Eric Price, Nati Srebro, et al. Equality of opportunity in supervised learning. In *Advances in neural information processing systems*, pages 3315–3323, 2016.

[HSNL18] Tatsunori B. Hashimoto, Megha Srivastava, Hongseok Namkoong, and Percy Liang. Fairness Without Demographics in Repeated Loss Minimization. *arXiv:1806.08010 [cs, stat]*, June 2018.

[HTF09] Trevor Hastie, Robert Tibshirani, and Jerome Friedman. The elements of statistical learning. 2009.

[IK19] Mahmoud M Ibrahim and Rafael Kramann. genesorter: Feature ranking in clustered single cell data. *bioRxiv*, 2019.

[Ilv19] Christina Ilvento. Metric Learning for Individual Fairness. *arXiv:1906.00250 [cs, stat]*, June 2019.

[JGBS15] Laurent Jacob, Johann A. Gagnon-Bartsch, and Terence P. Speed. Correcting gene expression data when neither the unwanted variation nor the factor of interest are observed. *Biostatistics*, 17(1):16–28, 08 2015.

[JKMR16] Matthew Joseph, Michael J. Kearns, Jamie Morgenstern, and Aaron Roth. Fairness in learning: Classic and contextual bandits. *CoRR*, abs/1605.07139, 2016.

[JKN+19] Christopher Jung, Michael J. Kearns, Seth Neel, Aaron Roth, Logan Stapleton, and Zhiwei Steven Wu. Eliciting and enforcing subjective individual fairness. *CoRR*, abs/1905.10660, 2019.

[KC09] Faisal Kamiran and Toon Calders. Classifying without discriminating. In *2009 2nd International Conference on Computer, Control and Communication*. IEEE, February 2009.

[KC11] Faisal Kamiran and Toon Calders. Data preprocessing techniques for classification without discrimination. *Knowledge and Information Systems*, 33(1):1–33, December 2011.

[Ken48] Maurice G. Kendall. Rank correlation methods. 1948.

[KGB16] Alexey Kurakin, Ian Goodfellow, and Samy Bengio. Adversarial Machine Learning at Scale. *arXiv preprint arXiv:1611.01236*, November 2016.

[KGZ19] Michael P. Kim, Amirata Ghorbani, and James Zou. Multiaccuracy: Black-box post-processing for fairness in classification. In *Proceedings of the 2019 AAAI/ACM Conference on AI, Ethics, and Society*, AIES '19, pages 247–254, New York, NY, USA, 2019. ACM.

[KKS39] M. G. Kendall, Sheila F. H. Kendall, and B. Babington Smith. The distribution of spearman's coefficient of rank correlation in a universe in which all rankings occur an equal number of times:. *Biometrika*, 30(3/4):251–273, 1939.

[KMR16] Jon M. Kleinberg, Sendhil Mullainathan, and Manish Raghavan. Inherent trade-offs in the fair determination of risk scores. *ArXiv*, abs/1609.05807, 2016.

[KNRW17] Michael Kearns, Seth Neel, Aaron Roth, and Zhiwei Steven Wu. Preventing Fairness Gerrymandering: Auditing and Learning for Subgroup Fairness. *arXiv e-prints*, page arXiv:1711.05144, Nov 2017.

[KNRW19] Michael Kearns, Seth Neel, Aaron Roth, and Zhiwei Steven Wu. An empirical study of rich subgroup fairness for machine learning. In *Proceedings of the Conference on Fairness, Accountability, and Transparency*, FAT* '19, pages 100–109, New York, NY, USA, 2019. ACM.

[KRR18] Michael Kim, Omer Reingold, and Guy Rothblum. Fairness through computationally-bounded awareness. In S. Bengio, H. Wallach, H. Larochelle, K. Grauman, N. Cesa-Bianchi, and R. Garnett, editors, *Advances in Neural Information Processing Systems 31*, pages 4842–4852. Curran Associates, Inc., 2018.

[KRS19] Michael Kearns, Aaron Roth, and Saeed Sharifi-Malvajerdi. Average Individual Fairness: Algorithms, Generalization and Experiments. *arXiv e-prints*, page arXiv:1905.10607, May 2019.

[KSS14] Peter V Kharchenko, Lev Silberstein, and David T Scadden. Bayesian approach to single-cell differential expression analysis. *Nature Methods*, 11(7):740–742, may 2014.

[KST+17] Hyun Min Kang, Meena Subramaniam, Sasha Targ, Michelle Nguyen, Lenka Maliskova, Elizabeth McCarthy, Eunice Wan, Simon Wong, Lauren Byrnes, Cristina M Lanata, Rachel E Gate, Sara Mostafavi, Alexander Marson, Noah Zaitlen, Lindsey A Criswell, and Chun Jimmie Ye. Multiplexed droplet single-cell RNA-sequencing using natural genetic variation. *Nature Biotechnology*, 36(1):89–94, December 2017.

[LGW19] Preethi Lahoti, Krishna P. Gummadi, and Gerhard Weikum. Operationalizing individual fairness with pairwise fair representations. *ArXiv*, abs/1907.01439, 2019.

[LHA14] Michael I Love, Wolfgang Huber, and Simon Anders. Moderated estimation of fold change and dispersion for RNA-seq data with DESeq2. *Genome Biology*, 15(12), December 2014.

[LJ16] Kristian Lum and James E. Johndrow. A statistical framework for fair predictive algorithms. *ArXiv*, abs/1610.08077, 2016.

[LL18] Wei Vivian Li and Jingyi Jessica Li. An accurate and robust imputation method scImpute for single-cell RNA-seq data. *Nature Communications*, 9(1), March 2018.

[LMRW18] Yuan Li, Benjamin Mark, Garvesh Raskutti, and Rebecca Willett. Graph-based regularization for regression problems with highly-correlated designs, 2018.

[LR18] Jaeho Lee and Maxim Raginsky. Minimax statistical learning with wasserstein distances. In S. Bengio, H. Wallach, H. Larochelle, K. Grauman, N. Cesa-Bianchi, and R. Garnett, editors, *Advances in Neural Information Processing Systems 31*, pages 2687–2696. Curran Associates, Inc., 2018.

[LRC+18] Romain Lopez, Jeffrey Regier, Michael B Cole, Michael Jordan, and Nir Yosef. Bayesian inference for a generative model of transcriptome profiles from single-cell rna sequencing. *bioRxiv*, 2018.

[LSB+15] Jacob H. Levine, Erin F. Simonds, Sean C. Bendall, Kara L. Davis, El ad D. Amir, Michelle D. Tadmor, Oren Litvin, Harris G. Fienberg, Astraea Jager, Eli R. Zunder, Rachel Finck, Amanda L. Gedman, Ina Radtke, James R. Downing, Dana Pe'er, and Garry P. Nolan. Data-driven phenotypic dissection of AML reveals progenitor-like cells that correlate with prognosis. *Cell*, 162(1):184–197, July 2015.

[LZ15] H. Lam and Enlu Zhou. Quantifying uncertainty in sample average approximation. In *2015 Winter Simulation Conference (WSC)*, pages 3846–3857, December 2015.

[MBS+15] Evan Z. Macosko, Anindita Basu, Rahul Satija, James Nemesh, Karthik Shekhar, Melissa Goldman, Itay Tirosh, Allison R. Bialas, Nolan Kamitaki, Emily M. Martersteck, John J. Trombetta, David A. Weitz, Joshua R. Sanes, Alex K. Shalek, Aviv Regev, and Steven A. McCarroll. Highly parallel genome-wide expression profiling of individual cells using nanoliter droplets. *Cell*, 161(5):1202–1214, May 2015.

[McL03] Don McLeish. Lecture notes in probability. `http://sas.uwaterloo.ca/~dlmcleis/s901/chapt3.pdf`, September 2003.

[MF19] Boris Mirkin and Trevor I. Fenner. Distance and consensus for preference relations corresponding to ordered partitions. *Journal of Classification*, 36(2):350–367, April 2019.

[MMK+15] Takeru Miyato, Shin-ichi Maeda, Masanori Koyama, Ken Nakae, and Shin Ishii. Distributional Smoothing with Virtual Adversarial Training. *arXiv:1507.00677 [cs, stat]*, July 2015.

[MMS+17] Aleksander Madry, Aleksandar Makelov, Ludwig Schmidt, Dimitris Tsipras, and Adrian Vladu. Towards deep learning models resistant to adversarial attacks. *ArXiv*, abs/1706.06083, 2017.

[ND16] Hongseok Namkoong and John C. Duchi. Stochastic Gradient Methods for Distributionally Robust Optimization with F-divergences. In *Proceedings of the 30th International Conference on Neural Information Processing Systems*, NIPS'16, pages 2216–2224, Barcelona, Spain, 2016. Curran Associates Inc.

[Nel06] Roger B. Nelsen. An introduction to copulas. 2006.

[Nešl07] Johanna Nešlehovà. On rank correlation measures for non-continuous random variables. *Journal of Multivariate Analysis*, 98(3):544 – 567, 2007.

[NYMP18] Vasilis Ntranos, Lynn Yi, Pall Melsted, and Lior Pachter. Identification of transcriptional signatures for cell types from single-cell RNA-Seq. *bioRxiv*, 2018.

[OL16] Petra Ornstein and Johan Lyhagen. Asymptotic properties of spearman's rank correlation for variables with finite support. *PLOS ONE*, 11(1):e0145595, January 2016.

[PAG+15] Franziska Paul, Ya'ara Arkin, Amir Giladi, Diego Adhemar Jaitin, Ephraim Kenigsberg, Hadas Keren-Shaul, Deborah Winter, David Lara-Astiaso, Meital Gury, Assaf Weiner, Eyal David, Nadav Cohen, Felicia Kathrine Bratt Lauridsen, Simon Haas, Andreas Schlitzer, Alexander Mildner, Florent Ginhoux, Steffen Jung, Andreas Trumpp, Bo Torben Porse, Amos Tanay, and Ido Amit. Transcriptional heterogeneity and lineage commitment in myeloid progenitors. *Cell*, 163(7):1663 – 1677, 2015.

[PMJ+15] Nicolas Papernot, Patrick McDaniel, Somesh Jha, Matt Fredrikson, Z. Berkay Celik, and Ananthram Swami. The Limitations of Deep Learning in Adversarial Settings. *IEEE European Symposium on Security and Privacy*, November 2015.

[PTB19] Flavien Prost, Nithum Thain, and Tolga Bolukbasi. Debiasing Embeddings for Reduced Gender Bias in Text Classification. *arXiv:1908.02810 [cs, stat]*, August 2019.

[PV13] Y. Plan and R. Vershynin. Robust 1-bit compressed sensing and sparse logistic regression: A convex programming approach. *IEEE Transactions on Information Theory*, 59(1):482–494, Jan 2013.

[PVG+11] F. Pedregosa, G. Varoquaux, A. Gramfort, V. Michel, B. Thirion, O. Grisel, M. Blondel, P. Prettenhofer, R. Weiss, V. Dubourg, J. Vanderplas, A. Passos, D. Cournapeau, M. Brucher, M. Perrot, and E. Duchesnay. Scikit-learn: Machine learning in Python. *Journal of Machine Learning Research*, 12:2825–2830, 2011.

[RMS09] M. D. Robinson, D. J. McCarthy, and G. K. Smyth. edgeR: a bioconductor package for differential expression analysis of digital gene expression data. *Bioinformatics*, 26(1):139–140, nov 2009.

[RSZ17] Y. Ritov, Y. Sun, and R. Zhao. On conditional parity as a notion of non-discrimination in machine learning. *ArXiv*, June 2017.

[RY18] Guy N. Rothblum and Gal Yona. Probably Approximately Metric-Fair Learning. *arXiv e-prints*, page arXiv:1803.03242, Mar 2018.

[SA10] Emily Steel and Juila Angwin. On the web's cutting edge, anonymity in name only. *The Wall Street Journal*, 2010.

[SEK15] Soroosh Shafieezadeh-Abadeh, Peyman Mohajerin Esfahani, and Daniel Kuhn. Distributionally Robust Logistic Regression. In *Advances in Neural Information Processing Systems*, pages 1576–1584, September 2015.

[sld19a] scikit-learn developers. Clustering performance evaluation. `https://scikit-learn.org/stable/modules/clustering.html#clustering-evaluation`, 2019. Online: last accessed 9 May 2019.

[sld19b] scikit-learn developers. Model evaluation: quantifying the quality of predictions. `https://scikit-learn.org/stable/modules/model_evaluation.html`, 2019. Online: last accessed 9 May 2019.

[SND18] Aman Sinha, Hongseok Namkoong, and John Duchi. Certifiable distributional robustness with principled adversarial training. In *International Conference on Learning Representations*, 2018.

[Spr07] Marcus Spruill. Asymptotic distribution of coordinates on high dimensional spheres. *Electron. Commun. Probab.*, 12:234–247, 2007.

[SR17] Charlotte Soneson and Mark D Robinson. Towards unified quality verification of synthetic count data with countsimQC. *Bioinformatics*, 34(4):691–692, 10 2017.

[SR18] Charlotte Soneson and Mark D Robinson. Bias, robustness and scalability in single-cell differential expression analysis. *Nature Methods*, 15(4):255–261, feb 2018.

[SRL+16] Debarka Sengupta, Nirmala Arul Rayan, Michelle Lim, Bing Lim, and Shyam Prabhakar. Fast, scalable and accurate differential expression analysis for single cells. *bioRxiv*, 2016.

[ST19] Mario L. Suvà and Itay Tirosh. Single-cell RNA sequencing in cancer: Lessons learned and emerging challenges. *Molecular Cell*, 75(1):7–12, July 2019.

[SZS+13] Christian Szegedy, Wojciech Zaremba, Ilya Sutskever, Joan Bruna, Dumitru Erhan, Ian Goodfellow, and Rob Fergus. Intriguing properties of neural networks. *arXiv e-prints*, page arXiv:1312.6199, Dec 2013.

[Tib97] Robert Tibshirani. The Lasso Method for variable selection in the Cox model. *Statistics in Medicine*, 16(4):385–395, February 1997.

[vDE12] Stijn van Dongen and Anton J. Enright. Metric distances derived from cosine similarity and pearson and spearman correlations, 2012.

[vDSN+18] David van Dijk, Roshan Sharma, Juozas Nainys, Kristina Yim, Pooja Kathail, Ambrose J. Carr, Cassandra Burdziak, Kevin R. Moon, Christine L. Chaffer, Diwakar Pattabiraman, Brian Bierie, Linas Mazutis, Guy Wolf, Smita Krishnaswamy, and Dana Pe'er. Recovering gene interactions from single-cell data using data diffusion. *Cell*, 174(3):716–729.e27, July 2018.

[Wag17] Florian Wagner. The XL-mHG test for gene set enrichment. February 2017.

[WAT18] F. Alexander Wolf, Philipp Angerer, and Fabian J. Theis. SCANPY: large-scale single-cell gene expression data analysis. *Genome Biology*, 19(1), February 2018.

[WGL+19] Hanchen Wang, Nina Grgic-Hlaca, Preethi Lahoti, Krishna P. Gummadi, and Adrian Weller. An Empirical Study on Learning Fairness Metrics for COMPAS Data with Human Supervision. *arXiv e-prints*, page arXiv:1910.10255, Oct 2019.

[WYY18] Florian Wagner, Yun Yan, and Itai Yanai. K-nearest neighbor smoothing for high-throughput single-cell rna-seq data. *bioRxiv*, 2018.

[xG] 10x Genomics. 1.3 million brain cells from e18 mice. `https://support.10xgenomics.com/single-cell-gene-expression/datasets/1.3.0/1M_neurons`. Accessed: 2018-09-22.

[XYS20] Songkai Xue, Mikhail Yurochkin, and Yuekai Sun. Auditing ML models for individual bias and unfairness. In *Proceedings of the 14th International Conference on Artificial Intelligence and Statistics (AISTATS)*, Palermo, Italy, 2020. submitted to AISTATS 2020.

[YBS20] Mikhail Yurochkin, Amanda Bower, and Yuekai Sun. Training individually fair ML models with sensitive subspace robustness. In *International Conference on Learning Representations*, Addis Ababa, Ethiopia, 2020.

[YRWC19] Yao-Yuan Yang, Cyrus Rashtchian, Yizhen Wang, and Kamalika Chaudhuri. Adversarial examples for non-parametric methods: Attacks, defenses and large sample limits. *ArXiv*, abs/1906.03310, 2019.

[ZH05] Hui Zou and Trevor Hastie. Regularization and variable selection via the elastic net. *Journal of the Royal Statistical Society: Series B (Statistical Methodology)*, 67(2):301–320, April 2005.

[ZLM18] Brian Hu Zhang, Blake Lemoine, and Margaret Mitchell. Mitigating unwanted biases with adversarial learning. In *AIES '18*, 2018.

[ZMMC+15] Amit Zeisel, Ana B. Muñoz-Manchado, Simone Codeluppi, Peter Lönnerberg, Gioele La Manno, Anna Juréus, Sueli Marques, Hermany Munguba, Liqun He, Christer Betsholtz, Charlotte Rolny, Gonçalo Castelo-Branco, Jens Hjerling-Leffler, and Sten Linnarsson. Cell types in the mouse cortex and hippocampus revealed by single-cell rna-seq. *Science*, 347(6226):1138–1142, 2015.

[ZPO17] Luke Zappia, Belinda Phipson, and Alicia Oshlack. Splatter: simulation of single-cell RNA sequencing data. *Genome Biology*, 18(1), sep 2017.

[ZTB+17] Grace X. Y. Zheng, Jessica M. Terry, Phillip Belgrader, Paul Ryvkin, Zachary W. Bent, Ryan Wilson, Solongo B. Ziraldo, Tobias D. Wheeler, Geoff P. McDermott, Junjie Zhu, Mark T. Gregory, Joe Shuga, Luz Montesclaros, Jason G. Underwood, Donald A. Masquelier, Stefanie Y. Nishimura, Michael Schnall-Levin, Paul W. Wyatt, Christopher M. Hindson, Rajiv Bharadwaj, Alexander Wong, Kevin D. Ness, Lan W. Beppu, H. Joachim Deeg, Christopher McFarland, Keith R. Loeb, William J. Valente, Nolan G. Ericson, Emily A. Stevens, Jerald P. Radich, Tarjei S. Mikkelsen, Benjamin J. Hindson, and Jason H. Bielas. Massively parallel digital transcriptional profiling of single cells. *Nature Communications*, 8:14049, Jan 2017. Article.

[ZVRG15] Muhammad Bilal Zafar, Isabel Valera, Manuel Gomez Rodriguez, and Krishna P Gummadi. Fairness constraints: Mechanisms for fair classification. *arXiv preprint arXiv:1507.05259*, 2015.

[ZWS+13] Rich Zemel, Yu Wu, Kevin Swersky, Toni Pitassi, and Cynthia Dwork. Learning fair representations. In Sanjoy Dasgupta and David McAllester, editors, *Proceedings of the 30th International Conference on Machine Learning*, volume 28 of *Proceedings of Machine Learning Research*, pages 325–333, Atlanta, Georgia, USA, 17–19 Jun 2013. PMLR.

[ZWT+17] Xun Zhu, Thomas K. Wolfgruber, Austin Tasato, Cédric Arisdakessian, David G. Garmire, and Lana X. Garmire. Granatum: a graphical single-cell RNA-seq analysis pipeline for genomics scientists. *Genome Medicine*, 9(1), December 2017.

[ZYJ14] Lijun Zhang, Jinfeng Yi, and Rong Jin. Efficient algorithms for robust one-bit compressive sensing. In Eric P. Xing and Tony Jebara, editors, *Proceedings of the 31st International Conference on Machine Learning*, volume 32 of *Proceedings of Machine Learning Research*, pages 820–828, Bejing, China, 22–24 Jun 2014. PMLR.