MIXTURE MODELS IN MACHINE LEARNING

A Dissertation Presented

by

SOUMYABRATA PAL

Submitted to the Graduate School of the
University of Massachusetts Amherst in partial fulfillment
of the requirements for the degree of

DOCTOR OF PHILOSOPHY

February 2022

College of Information and Computer Sciences

MIXTURE MODELS IN MACHINE LEARNING

A Dissertation Presented

by

SOUMYABRATA PAL

Approved as to style and content by:

Arya Mazumdar, Chair

Akshay Krishnamurthy, Member

Barna Saha, Member

Patrick Flaherty, Member

James Allan, Chair of the Faculty
College of Information and Computer Sciences

DEDICATION

To my wonderful wife Raka and my parents.

ACKNOWLEDGMENTS

First and foremost, I would like to thank my Ph.D. advisor Arya Mazumdar for his unwavering support and immense guidance throughout the last five and a half years. I am one of the few students lucky enough to have an advisor who not only helped me understand how to conduct research but also genuinely cared about me. He taught me so much during these past years including information theory and probabilistic methods which forms the basis of this thesis. Arya always has a keen sense of the broad perspective of the research and the best way to present it to the general audience which I strive to develop. I wish to carry his teachings and make him proud in the next phases of my life.

I am thankful to Wasim Huleihel who has become a very close friend of mine during these years. He patiently helped me understand the art of writing technical ideas in a beautiful manner. He has always been there for me to guide me and provide a second pair of eyes whenever I needed them. His breadth of knowledge on the different problems has always astounded me and continues to inspire me. I am also grateful to the members of my dissertation committee especially Barna Saha and Akshay Krishnamurthy who have collaborated with me on a number of projects and have always tried their best to help me. Over the years, they have provided many invaluable pieces of advice to me.

At Amherst, I found close friends in Raj, Anil and Ryan. I do miss the weekly poker games where we had so much fun over the years. They helped me stay sane during the first few years in a foreign country and never let me feel that I was out of home. I am indeed indebted to them for their friendship and support. I will cherish the wonderful memories at Lantern Court Apartments in Amherst.

My parents, Malabika and Debabrata were the most supportive parents that one can wish for. They have always tried to understand the issues I faced and always supported me in this endeavor. My mother always kept a cool head in the difficult situations that I faced during the last few years and helped me immensely to cope with them. Without their unwavering support and love, I could not have completed this degree.

Finally, all of my accomplishments including the completion of this thesis would not have been possible without my wife Raka by my side. Each day, I am more grateful to God that she came into my life and stayed with me. The sacrifices that she had made for me and the way she always stood by me even at the cost of her own dreams and aspirations make me realize how lucky I am. We found our first home at Amherst and then at San Francisco where I feel so proud to see her become the proficient Data Scientist at Udemy that she deserves to be. She has been my best friend and the patient listener to all my problems. These last two and a half years in San Francisco with her have indeed been the best years of my life.

ABSTRACT

MIXTURE MODELS IN MACHINE LEARNING

FEBRUARY 2022

SOUMYABRATA PAL

B.Tech, INDIAN INSTITUTE OF TECHNOLOGY KHARAGPUR

M.S, UNIVERSITY OF MASSACHUSETTS AMHERST

Ph.D., UNIVERSITY OF MASSACHUSETTS AMHERST

Directed by: Professor Arya Mazumdar

Modeling with mixtures is a powerful method in the statistical toolkit that can be used for representing the presence of sub-populations within an overall population. In many applications ranging from financial models to genetics, a mixture model is used to fit the data. The primary difficulty in learning mixture models is that the observed data set does not identify the sub-population to which an individual observation belongs. Despite being studied for more than a century, the theoretical guarantees of mixture models remain unknown for several important settings.

In this thesis, we look at three groups of problems. The first part is aimed at estimating the parameters of a mixture of simple distributions. We ask the following question: How many samples are necessary and sufficient to learn the latent parameters? We propose several approaches for this problem that include complex analytic tools to connect statistical distances between pairs of mixtures with the characteristic

function. We show sufficient sample complexity guarantees for mixtures of popular distributions (including Gaussian, Poisson and Geometric). For many distributions, our results provide the first sample complexity guarantees for parameter estimation in the corresponding mixture. Using these techniques, we also provide improved lower bounds on the Total Variation distance between Gaussian mixtures with two components and demonstrate new results in some sequence reconstruction problems.

In the second part, we study Mixtures of Sparse Linear Regressions where the goal is to learn the best set of linear relationships between the scalar responses (i.e., labels) and the explanatory variables (i.e., features). We focus on a scenario where a learner is able to choose the features to get the labels. To tackle the high dimensionality of data, we further assume that the linear maps are also "sparse", i.e., have only few prominent features among many. For this setting, we devise algorithms with sub-linear (as a function of the dimension) sample complexity guarantees that are also robust to noise.

In the final part, we study Mixtures of Sparse Linear Classifiers in the same setting as above. Given a set of features and the binary labels, the objective of this task is to find a set of hyperplanes in the space of features such that for any (feature, label) pair, there exists a hyperplane in the set that justifies the mapping. We devise efficient algorithms with sub-linear sample complexity guarantees for learning the unknown hyperplanes under similar sparsity assumptions as above. To that end, we propose several novel techniques that include tensor decomposition methods and combinatorial designs.

TABLE OF CONTENTS

APPENDICES

LIST OF TABLES

Table **Page**

LIST OF FIGURES

Figure **Page**

NOTATION

The following notations are used throughout the proposal unless otherwise specified.

- $\mathbb{Z}$ is used to denote the set of integers.
- $\mathbb{R}$ denotes the set of real numbers.
- $\mathbb{N}$ denotes the natural numbers (positive integers).
- $[n]$ is used to denote the set $\{1, 2, \ldots, n\}$ for any $n \in \mathbb{N}$ unless otherwise specified.
- $\boldsymbol{A}$ denotes a matrix with rows $\boldsymbol{A}_i$ and entries $\boldsymbol{A}_{i,j}$.
- $\mathcal{O}$ denotes an oracle function.
- $\mathcal{G} = (\mathcal{V}, \mathcal{E})$ denotes a graph $\mathcal{G}$ with vertex set $\mathcal{V}$ and edge set $\mathcal{E}$.
- $\boldsymbol{v}$ denotes a vector with entries $\boldsymbol{v}_i$.
- $\mathcal{N}(\mu, \sigma^2)$ is the normal distribution with mean μ and variance σ^2.
- $\mathsf{Ber}(p)$ is the Bernoulli distribution with parameter p.
- $\triangleq$ reads "is defined to be equal to."
- $|S|$ is the cardinality of the set S.
- $\mathsf{supp}(\boldsymbol{v})$ is the support of $\boldsymbol{v}$, $\{i : \boldsymbol{v}_i \neq 0\}$.
- $\mathcal{M}$ denotes a mixture of distributions
- For any positive integer p, $||\boldsymbol{x}||_p \triangleq (\sum_{i=1}^{n} |\boldsymbol{x}_i^p|)^{1/p}$ denotes the ℓ_p norm. The ℓ_0 norm $||\boldsymbol{x}||_0$ denotes the number of non-zero entries in the vector $\boldsymbol{x}$.

CHAPTER 1

INTRODUCTION

In the modern world, data is ubiquitous and is becoming more easily available each day. A vast number of applications collect data to extract important insights and subsequently design systems that are based on such insights. Unfortunately, much of this data is unstructured, high dimensional, corrupted, or incomplete and it seems impossible to process such data without carefully designing rich and expressive models that incorporate prior structural assumptions and capture inherent properties of real-world datasets. One approach for designing efficient algorithms for high dimensional complex datasets is by designing generative models of different forms of data and exploiting many natural structural properties such as sparsity, presence of geometric motifs, and underlying similarity relations. Such structures in the data either occur naturally in many applications (for instance, image and speech signals are sparse) or are often embedded into data via pre-processing (mapping data points into a feature vector space induces a geometric structure). In this thesis, we model the observed data by a mixture of finite functions from a class of "simple" functions. Canonical examples of such models can be mixtures of some well-known distribution, mixtures of linear regressions, and mixture of linear classifiers. Such models are extremely rich and expressive albeit having a small number of parameters and allow for rigorous theoretical treatment. Finite mixture models have provided a sound mathematically rigorous approach to the statistical modeling of a wide variety of random phenomena. As finite mixture models are quite flexible and can be used to model complex data, they have been proved to be extremely useful and have continued to receive significant

attention over the last few years from both a theoretical and practical point of view. Finite mixture models have been successfully used for modeling data in many different fields including astrology, genetics, medicine, psychiatry, economics, marketing, and engineering among many others in the biological, physical, and social sciences. Finite mixture models are especially successful in modeling datasets having a group structure or the presence of a sub-population within the overall population. In this scenario, finite mixture models are useful in assessing the error rates (sensitivity and specificity) of diagnostic and screening procedures in the absence of a gold standard (see [115], Chapter 1). Furthermore, any continuous distribution can be approximated arbitrarily well by a finite mixture of normal densities with common variance (or co-variance matrix in the multivariate setting); mixture models provide a convenient framework for modeling datasets with unknown distributional shapes. A mixture model can model quite complex distributions through an appropriate choice of its components to represent accurately the local areas of support of the true distribution. Often, it can handle situations where a single parametric family cannot provide a satisfactory model for local variations in the observed data. Such flexibility of mixture models is useful for theoretical analysis of neural networks as well [20]. As an example, consider neural networks formed by using radial basis functions. In this setting, the input data can be modeled by a normal mixture. Subsequently, the second layer weights of the neural network can be estimated from the input data and their known outputs using maximum likelihood. In this thesis, we study many different statistical reconstruction problems under the big umbrella of *mixture models in machine learning* that can be categorized into two distinct frameworks. First, in the experimental design setting, we study the framework when we can carefully choose the data (design the covariates) or obtain side information by querying from an *oracle*. The presence of the *oracle* is justified in the recent surge of interest in crowdsourcing where it is possible to make queries to the crowd-worker and extract relevant responses. This framework is

extremely useful since it is very difficult to incorporate human analysis in an algorithm otherwise. The oracle can also be inherently present in a recommendation system or in an advertisement engine where the query corresponds to the recommended item or advertisement and the response corresponds to whether the user consumed the item or not. This framework, which is similar to the well-known *active learning* setting (see [126]), designs a model by utilizing responses from a small amount of carefully chosen data. For regression or classification problems, we can query for the label of a carefully designed feature vector. However, these queries are expensive since they cost time and money and therefore a natural objective is to minimize the number of queries to the oracle and still fulfill the objective. Secondly, in the unsupervised setting, we study the framework where we use a parameterized family of mixture models to fit the dataset under the assumption that each data point in the dataset is generated independently according to some distribution belonging to the parameterized family. This framework includes mixtures of simple distributions where, given a latent random variable denoting a particular component, each data point is sampled from a simple distribution. Furthermore, this framework also includes mixtures of linear regression and mixtures of linear classifiers where each data point consists of a (feature, label) pair; the feature is sampled from some multivariate distribution while the response given the feature is a mixture of simple functions of the feature. Finally, we also mention the supervised setting although it is not studied in this thesis. In this setting, the goal is to design efficient algorithms that minimize some loss function while fitting a parameterized family of mixture models to some dataset that is not necessarily generated according to some member of the family. The objective is to understand the performance of the fitted model to unseen data. We believe that the supervised setting for mixture models is an important direction to study in the future and can have many far-reaching practical applications.

1.1 A Brief History of Mixture models

It is believed that the first use and analysis of mixture models to fit a dataset was over a century ago by Karl Pearson, a luminary biometrician. In the now-classic paper [118], Pearson fitted a mixture of two normal probability density functions with different means, variances, and component weights to a dataset provided by his friend Weldon [143, 144]. The latter paper might have been the first-ever to advocate statistical analysis as a primary method for studying biological problems; see Stigler [132], Chapter 8 and Tarter and Lock [136] for a more detailed description. The data analyzed by Pearson [118] consisted of measurements on the ratio of the forehead to the body length of 1000 crabs sampled from the Bay of Naples. These measurements recorded in the form of 29 intervals are displayed in Figure 1.1 along with the plot of the density of a single normal distribution fitted to that dataset. Waldon [143] speculated that the asymmetry in the histogram of this dataset is an indicator of the fact that the Naples crab population contains two sub-populations. Subsequently, Weldon asked his friend Pearson for help as Pearson was trained more in statistics. Pearson's [118] mixture model approach suggested as well that there were two subspecies present. Figure 1.1 also plots the density of the two-component normal mixture as obtained by using maximum likelihood to fit this model to the data in their original interval form. Pearson [118] used the method of moments to fit the mixture model to the mid-points of the intervals which gives a fit very similar to the maximum likelihood approach. It can be seen from Figure 1.1 that a mixture of two normal distributions with components having unequal variances can be used to model the skewness of the data; on the other hand, a single normal distribution was inadequate for modeling such a dataset. Pearson [118] obtained his parameter estimates as the solution of ninth-degree polynomial which was truly a heroic task a century ago. Not surprisingly, various attempts were made over the ensuing years to

simplify Pearson's [118] moments-based approach to the fitting of a normal mixture model.

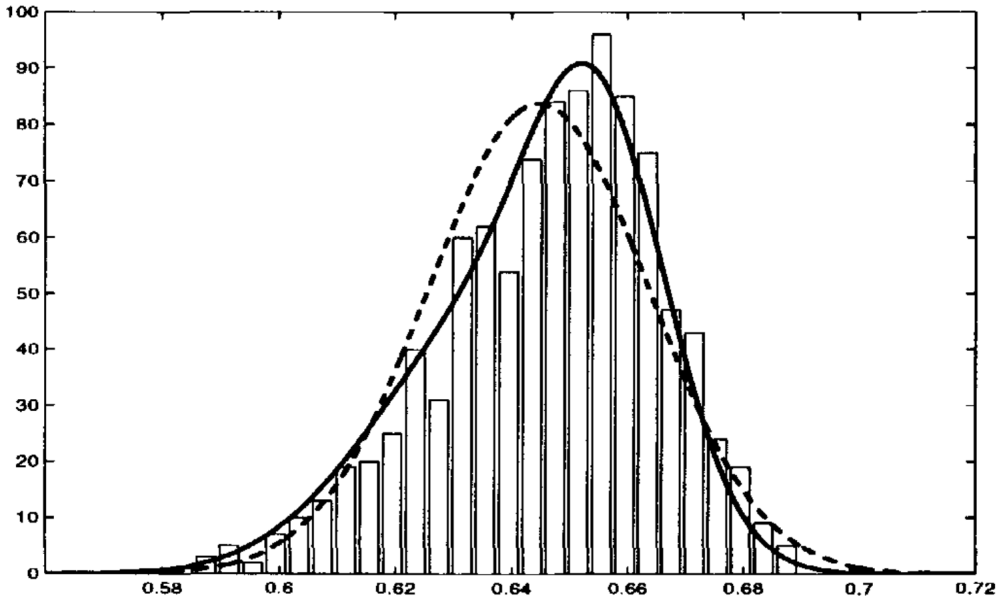


Figure 1.1: Plot of forehead to body length data on 1000 crabs and of the fitted one component normal distribution (dashed line) and two-component (solid line) normal mixture models (Image taken from [115]).

Although mixture models were first studied more than a century ago, considerable advances have been made only in the last couple of decades. Even with the advent of high-speed computers, there was significant reluctance in fitting mixture models to datasets comprising more than one dimension due to a lack of understanding of issues that arise with such fitting namely the presence of multiple maxima in the likelihood function and the likelihood function being unbounded (for example in a mixture of normals with unequal covariance matrices). However, an increased understanding over the years of these issues led to significantly higher adoption of mixture models in practice. In the 1960s, the fitting of mixture models by maximum likelihood have been studied in several papers including the seminal papers by Day [44] and Wolfe [145]. In 1977, the seminal paper of Dempster, Laird and Rubin [50] on the EM algorithm highly stimulated interest in the application of mixture models to fit heterogeneous

data. Since then, as Aitkin [5] noted, most applications of mixture models in practice were using the EM algorithm. Despite such a widescale adoption of the EM algorithm, very little was known about theoretical guarantees of fitting a mixture model until recently. Starting with the seminal work of [38], computational and statistical aspects of learning mixture models have been the subject of intense investigation in the theoretical computer science and statistics communities [3, 88, 18, 9, 114, 63, 34, 2, 81, 55, 95, 72]. Recently in [6, 107, 125, 32, 155], several papers used tensors to provide theoretical guarantees on parameter estimation in several latent variable models. Even more recently, in a parallel line of work, several papers [13, 40, 150, 101, 100] have sought to provide theoretical guarantees for the EM algorithm including local and global convergence results. Despite such advances, many theoretical questions on mixture models remain unanswered. Finally, in the statistical literature, for more details, we refer the reader to several well-written books on mixture models such as Everitt [61], Titterington et al. [137], McLachlan & Basford [111], Lindsay [105], Bohning [23] and McLachlan & Peel [115] among many others. In addition, mixture models are addressed in several books involving classification, machine learning, and other fields in multivariate analysis.

1.2 Mixtures of Distributions

1.2.1 Parameter learning in mixtures

In Chapter 2, we study sample complexity guarantees to reconstruct a mixture of simple canonical distributions. In this literature, there are two flavors of result: (1) *parameter estimation*, where the goal is to identify the mixing weights and the parameters of each component from samples, and (2) *density estimation* or PAC-learning, where the goal is simply to find a distribution that is close in some distance (e.g., TV distance) to the data-generating mechanism. Density estimation can be further subdivided into *proper* and *improper learning* approaches depending on whether

the algorithm outputs a distribution from the given mixture family or not. These three guarantees are quite different. Apart from Gaussian mixtures, where all types of results exist, prior work for other mixture families largely focuses on density estimation, and very little is known for parameter estimation outside of Gaussian mixture models. In the first part of this chapter, we focus on parameter estimation and provide two new approaches, both of which apply to several mixture families. We provide results on the exact estimation of the parameters under the assumption that the unknown parameters lie on an integer lattice. In the second part of this chapter, we specifically focus on mixtures of Gaussians with two components and shared component variance. Note that tight bounds are known for the total variation distance between single Gaussians; however, they have only recently been derived as closed-form functions of the distribution parameters [16, 52]. The functional form of the TV distance bound is often much more useful in practice because it can be directly evaluated based on only the means and covariances of the distribution. This has opened up the door for new applications to a variety of areas, such as analyzing ReLU networks [146], distribution learning [10, 12], private distribution testing [28, 31], and average-case reductions [27]. Inspired by the wealth of applications for single Gaussian total variation bounds, we investigate the possibility of deriving analogous results for mixtures with two components. As our next contribution, we complement the single Gaussian results and derive tight lower bounds for pairs of mixtures containing two equally weighted Gaussians with shared variance. We also present our results in a closed form in terms of the gap between the component means and certain statistics of the covariance matrix. The total variation distance between two distributions can be upper bounded by other distances/divergences (e.g., KL divergence, Hellinger distance) that are easier to analyze. In contrast, it is a key challenge to develop ways to lower bound the total variation distance. The shared variance case is important because it presents some of the key difficulties in parameter estimation and is widely studied [41, 147].

For example, mean estimation with shared variance serves as a model for the sensor location estimation problem in wireless or physical networks [94, 106, 138]. As a consequence of these new bounds, we study estimating the means of the underlying components with minimal samples in one dimension. We also design and analyze a new, polynomial-time estimation algorithm. Our sample complexity results match the guarantees provided by the spectral estimator [149] and is also robust to the assumption that the samples are generated from a member of the mixture model family. Finally, the literature on theoretical guarantees on mixtures of distributions is quite large, and we have just referred to a sample of the most relevant chapters here. A bigger overview on learning distributions can be found in the recent monographs such as [113, 53].

1.2.2 Support recovery in sparse mixtures

In Chapter 5, we study the sample complexity of learning the support of a mixture of distributions having sparse latent parameters. Most high dimensional datasets have sparse parameters and sparsity is a natural constraint on the set of parameters to reduce the effective dimension of the space of solutions to devise efficient algorithms. In such a setting, support recovery of the latent parameter vectors is often the first step towards parameter estimation. Sparsity has been considered in the context of mixtures in [139, 7, 11], where it is assumed only a few dimensions of the component means are relevant for de-mixing. In this paper, we considered a slightly different model. We assume the means themselves are sparse. The former problem can be reduced to our setting if one of the component means is known. Based on the method of moments, we provide a general framework to learn the supports of the latent parameter vectors in mixture of simple distributions that satisfies the following assumption: moments of the distribution of each coordinate can be described as a polynomial in the component parameters. The authors in [18] showed (see Table 2 in [18]) that most common

distributions including Gaussians, Uniform distribution, Poisson, Laplace satisfy this assumption. Therefore our results in this section are not only applicable to many canonical settings but also makes progress towards quantifying the sufficient number of moments in the following general problem:

Consider L random vectors $\boldsymbol{v}^1, \boldsymbol{v}^2, \ldots, \boldsymbol{v}^L \in \mathbb{R}^d$ such that each of them are k-sparse i.e. $||\boldsymbol{v}^i||_0 \leq k$ for all $i \in [L]$. Suppose we obtain samples of a random vector $\boldsymbol{x} \in \mathbb{R}^d$ such that given a random variable t sampled uniformly from the set $\{1, 2, \ldots, L\}$, each of its d co-ordinates $(\boldsymbol{x}_i \mid t = j)$ are independently distributed according to a distribution $\mathcal{D}(\theta)$ with parameter $\theta = \boldsymbol{v}_i^j$. Our objective is to use minimum number of samples to recover the support of all the unknown vectors. Below, we provide a few examples of the distribution $\mathcal{D}$ and the corresponding unknown parameter:

1. $\mathcal{D}(\theta)$ can be a Gaussian distribution with mean θ. This setting corresponds to a mixture of high-dimensional Gaussians with sparse means and identity covariance.

2. $\mathcal{D}(\theta)$ can be a uniform distribution with range $[\theta, b]$ for a fixed and known b.

3. $\mathcal{D}(\theta)$ can be a Poisson distribution with mean θ.

An alternate approach to the support recovery problem is to first recover the union of support of the unknown parameters and then apply known parameter estimation guarantees to identify the support of each of the unknown vectors after reducing the dimension of the problem. Note that this approach crucially requires parameter estimation guarantees for the corresponding family of mixtures which is a significantly more difficult objective. To the best of our knowledge, most constructive sample complexity guarantees for parameter estimation in mixture models without separability assumptions correspond to mixtures of Gaussians [88, 18, 114, 72, 64, 77, 108, 76]. However, most known results correspond to mixtures of Gaussians with two components. The only known results for parameter estimation in mixtures of Gaussians with more

than 2 components is [114] but as we describe later, using the alternate approach with the guarantees in [114] results in a logarithmic dependence on the latent space dimension and polynomial dependence on the sparsity which is undesirable in this setting. On the contrary, our sample complexity guarantees only scale logarithmically with the dimension and is essentially independent of the sparsity (for constant L) which is a significant improvement over the alternate approach. For other distributions, [18, 99] studied parameter estimation under the same moment-based assumption that we use. However, [18] uses non-constructive arguments from algebraic geometry because of which, their results did not include bounds on the sufficient number of moments for learning the parameters in a mixture model. In [99], the authors resolve this question to a certain extent for these aforementioned families of mixture models as they quantify the sufficient number of moments for parameter estimation under the restrictive assumption that the latent parameters lie on an integer lattice. Therefore, our results for these distributions form the first guarantees for support recovery.

1.2.3 Trace Reconstruction

We study the sparse trace reconstruction problem in Chapter 2. The trace reconstruction problem was first proposed by Batu et al. [17] where the goal is to reconstruct an unknown string $\boldsymbol{x} \in \{0,1\}^n$ given a set of random subsequences of $\boldsymbol{x}$. Each subsequence, or "trace", is generated by passing $\boldsymbol{x}$ through the *deletion channel* in which each entry of $\boldsymbol{x}$ is deleted independently with probability p. The locations of the deletions are not known; if they were, the channel would be an *erasure channel*. The central question is to find how many traces are required to exactly reconstruct $\boldsymbol{x}$ with high probability. This intriguing problem has attracted significant attention from a large number of researchers [89, 141, 17, 80, 79, 119, 75, 116, 45, 110, 43, 36]. In a recent breakthrough, De et al. [45] and Nazarov and Peres [116] independently showed that for the general problem i.e. when $k = n$, $\exp(O((n/q)^{1/3}))$ traces suffice

where $q = 1 - p$. This bound is achieved by a *mean-based* algorithm, which means that the only information used is the fraction of traces that have a `1` in each position. While $\exp(O((n/q)^{1/3}))$ is known to be optimal amongst mean-based algorithms, the best algorithm-independent lower bound is the much weaker $\Omega(n^{5/4}/\log n)$ [78]. Many variants of the problem have also been considered including: (1) larger alphabets and (2) an average case analysis where $\boldsymbol{x}$ is drawn uniformly from $\{\mathtt{0}, \mathtt{1}\}^n$. Larger alphabets are only easier than the binary case since we can encode the alphabet in binary, e.g., by mapping a single character to `1` and the rest to `0` and repeating for all characters. In the average case analysis, the state-of-the-art result is that $\exp(O(\log^{1/3}(n)))$ traces suffice*, whereas $\Omega(\log^{9/4} n/\sqrt{\log\log n})$ traces are necessary [75, 79, 78]. Very recently, and concurrent with our work, other variants have been studied including a) where the bits of x are associated with nodes of a tree whose topology determines the distribution of traces generated [43] and b) where x is a codeword from a code with $o(n)$ redundancy [36].

1.3 Sparse Mixture Models in Experimental Design setting

As discussed previously, in the experimental design setting, the learning algorithm has access to an oracle and can specify the covariates or the features for which the response is obtained from the oracle. The objective of this framework is to learn the latent parameters of the problem by using the minimum number of designed features (also called *sensing* vectors) and their corresponding responses (also called *samples*). Such a setting can arise naturally in recommendation systems where the recommending engine represents the learning algorithm and the tastes of any user can be modeled by an unknown parameter vector. The task of the recommendation engine is to learn the parameters representing the user's tastes with minimum number

*p is assumed to be constant in that work.

of recommendations so that it can provide personalized recommendations to the user. When the same account is used by a family or a group of users, the recommendation engine cannot tag the purchase history/ reviews to a particular user; subsequently, we can model this setting by a mixture model. Finally, we also impose the constraint of sparsity on the latent parameters as most high dimensional parameter vectors are naturally sparse. In a movie recommendation system, for example, it is expected that each user only likes a few particular genres among many available. It is interesting that Städler et al. [131] argued to impose sparsity on the solutions, implying that each linear function depends on only a small number of variables. In this paper we are concerned with exactly this same problem.

1.3.1 Mixtures of Linear Regressions

Learning mixtures of linear regressions is a natural generalization of the basic linear regression problem. In the first part of Chapter 3, the goal is to learn the best linear relationship between the scalar responses (i.e., labels) and the explanatory variables (i.e., features). In the generalization, each scalar response is stochastically generated by picking a function uniformly from a set of L unknown linear functions, evaluating this function on the explanatory variables, and possibly adding noise; the goal is to learn the set of L unknown linear functions. In the second part of Chapter 3, we study the problem of support recovery where the goal is to recover the support (positions of the non-zero coordinates) of the L unknown linear functions from minimum number of linear measurements. The problem was introduced by De Veaux [47] over thirty years ago and has recently attracted growing interest [32, 62, 131, 140, 151, 153]. Recent work focuses on a query-based scenario in which the input to the randomly chosen linear function can be specified by the learner. The *sparse* setting, in which each linear function depends on only a small number of variables, was recently considered by Yin et al. [153], and can be viewed as a generalization of the well-studied compressed sensing

problem [30, 57]. The problem has numerous applications in modeling heterogeneous data arising in medical applications, behavioral health, and music perception [153].

1.3.2 Mixtures of Linear Classifiers

One of the first and most basic tasks of machine learning is to train a binary linear classifier. Given a set of explanatory variables (features) and binary responses (labels), the objective of this task is to find the hyperplane in the space of features that best separates the variables according to their responses. In Chapter 4, we consider a natural generalization of this problem and model a classification task as a mixture of L components. In this generalization, each response is stochastically generated by picking a hyperplane uniformly from the set of L unknown hyperplanes and then returning the side of that hyperplane the feature vector lies. The goal is to learn all of these L hyperplanes as accurately as possible, using the least number of responses. As in Chapter 3, we also study the setting where the goal is to learn the supports of all the normals to these hyperplanes. This can be termed as a mixture of binary linear classifiers [134]. A similar mixture of simple machine learning models has been around for at least the last thirty years [47] with a mixture of linear regression models being the most studied ones [35, 83, 91, 127, 130, 142, 152, 157]. Models of this type are pretty good function approximators [19, 86] and have numerous applications in modeling heterogeneous settings such as machine translation [104], behavioral health [49], medicine [21], object recognition [121] etc. While algorithms for learning the parameters of a mixture of linear regressions are solidly grounded (such as tensor decomposition based learning algorithms of [32]), in many of the above applications the labels are discrete categorical data, and therefore a mixture of classifiers is a better model than a mixture of regressions. To the best of our knowledge, [134] first rigorously studied a mixture of linear classifiers and provided polynomial time algorithm to approximate the subspace spanned by the component

classifiers (hyperplane-normals) as well as a prediction algorithm that gives a feature and label, correctly predicts the component used. In this chapter we study a related but different problem: the sample complexity of learning *all* the component hyperplanes. Our model also differs from [134] where the component responses are 'smoothened out'. Here the term *sample complexity* is used with a slightly generalized meaning than traditional learning theory - as we explain next and then switch to the term *query complexity* instead. We assume while queried with a point (vector), an oracle randomly chooses one of the L binary classifiers and then returns an answer according to what was chosen. For most of this chapter, we concentrate on recovering 'sparse' linear classifiers, which implies that each of the classifiers uses only a few of the explanatory variables. This setting is in the spirit of the well-studied *1-bit compressed sensing* (1bCS) problem. In 1-bit compressed sensing, linear measurements of a sparse vector are quantized to only 1 bit, e.g. indicating whether the measurement outcome is positive or not, and the task is to recover the vector up to a prescribed Euclidean error with a minimum number of measurements. An overwhelming majority of the literature focuses on the nonadaptive setting for the problem [1, 4, 65, 71, 85, 120]. Also, a large portion of the literature concentrates on learning only the support of the sparse vector from the 1-bit measurements [1, 71].

It was shown in [85] that $O(\frac{k}{\epsilon}\log(\frac{n}{\epsilon}))$ Gaussian queries[†] suffice to approximately (to the Euclidean precision ϵ) recover an unknown k-sparse vector $\boldsymbol{\beta}$ using 1-bit measurements. Given the labels of the query vectors, one recovers $\boldsymbol{\beta}$ by finding a k-sparse vector that is consistent with all the labels. If we consider enough queries, then the obtained solution is guaranteed to be close to the actual underlying vector. [1] studied a two-step recovery process, where in the first step, they use queries corresponding to the rows of a special matrix, known as *Robust Union Free Family*

[†]all coordinates of the query vector are sampled independently from the standard Gaussian distribution

(RUFF), to recover the support of the unknown vector $\boldsymbol{\beta}$ and then use this support information to approximately recover $\boldsymbol{\beta}$ using an additional $\tilde{O}(\frac{k}{\epsilon})$ Gaussian queries. Although the recovery algorithm works in two steps, the queries are non-adaptive.

1.4 Sparse Mixture models in Unsupervised setting

In Chapter 5, we investigate the support recovery problem in the setting when the latent parameters of a mixture model are sparse. In the unsupervised setting, we assume that we obtain samples in the form of (feature, label) tuples where the features are generated according to a high dimensional Gaussian with zero mean and identity covariance, and the response given the feature is modeled by a mixture of simple functions with unknown sparse parameters. In this setting, we provide several techniques for obtaining support recovery guarantees including a general framework using low-rank tensor decomposition of integral tensors in different mixture models. While low-rank tensor decomposition is a popular method [6, 125, 32] for providing parameter estimation guarantees in mixture models, the theoretical results depend on the minimum eigenvalue of the set of the latent parameters; in other words, if the latent parameters are linearly dependent, vanilla low rank tensor decomposition fail to recover the parameters. For the support recovery problem, our tensor-based approach does not suffer from this approach as we can crucially use the property that the tensors have integral entries for low-rank decomposition. We believe that this framework is quite general and powerful and can be extended to many other mixture models as well.

1.4.1 Mixtures of Linear Regression

In the mixture of sparse linear regression (MLR) problem, the response given the feature is modeled by a mixture of linear functions of the feature with sparse parameters. For the support recovery problem in the sparse MLR setting, we provide a

suite of results under different assumptions namely 1) the unknown sparse parameters are binary 2) the unknown sparse parameters have the same sign 3) the unknown sparse parameters are distributed according to a Gaussian. MLR is a popular mixture model and has many applications due to its effectiveness in capturing non-linearity and its model simplicity [47, 62]. It has been a recent theoretical topic for analyzing benchmark algorithms for non-convex optimization [32, 92, 35]. There has been a large body of work studying the theoretical guarantees in this setting in recent years [6, 125, 152, 156, 13, 92] culminating with [103] where the authors proved sample complexity guarantees for parameter estimation which is linear is the dimension and is optimal in its dependence on the dimension. For the support recovery problem, one alternative approach is to first find the union of support of the unknown parameters and then apply the parameter estimation guarantees to recovery the support of each of the unknown linear functions. Compared to the alternate approach outlined above, we obtain sample complexity guarantees that have significantly better dependencies on many parameters.

1.4.2 Mixtures of Linear Classifiers

Unlike the MLR and MD (mixtures of distributions) setting, mixture of linear classifiers (MLC) is far less studied. In the sparse MLC setting, the response given the feature is modeled by the mixture of the sign of the linear functions of the feature where the unknown linear functions have sparse parameters. MLC is suitable for modeling datasets where the response is binary and the dataset contains sub-groups exhibiting different behavior. MLC is significantly more difficult to analyze than MLR since only the sign of the linear function of the feature is revealed. We study the support recovery problem in sparse MLC under the setting that all the parameters of the unknown vectors are either non-negative or they are non-positive. Although this assumption might seem restrictive, note that theoretical work in the MLC setting is extremely

limited. To the best of our knowledge, there are only two relevant papers [134, 125] that have studied this problem theoretically in our setting. [134] does not make any assumptions on sparsity and provides an algorithm for recovering the subspace in which the parameter vectors corresponding to the unknown linear functions lie. In contrast, support recovery is a different objective and hence is incomparable to the subspace recovery guarantees. The second work, [125] uses tensors to resolve this issue and provides sample complexity guarantees for learning the parameter vectors but their sample complexity is inversely proportional to the square of the minimum eigenvalue of the matrix comprising the unknown parameter vectors as columns. This is an unwanted dependence as it implies that if the parameter vectors are linearly dependent, then the algorithm will require infinite samples to recover the parameter vectors. Note that our support recovery guarantees do not have any such assumption on the parameters. Moreover, unlike the MD and MLR setting, it is not evident in MLC how to recover the union of support of the unknown sparse vectors. Hence the sample complexity obtained by applying the results in [125] directly will lead to a polynomial dependence on the dimension of the latent space which is undesirable (ideally, we require a logarithmic dependence on the latent space dimension). Our results exhibit such dependence on the dimension and also do not assume linear independence of the parameter vectors. Therefore, our results make important progress towards further understanding of theoretical properties of mixtures where the response is a mixture of non-linear functions of the feature.

CHAPTER 2

MIXTURES OF DISTRIBUTIONS

2.1 Introduction

In this chapter, we study sample complexity guarantees of mixtures of simple canonical distributions. Mixtures of distributions are a very rich and powerful model to fit the data. This content of this chapter is divided into two parts. In the first part, we demonstrate two different approaches to obtain our main results. Our first approach is analytic in nature and yields new sample complexity guarantees for univariate mixture models including Gaussian, Binomial, and Poisson. The variational distance (a.k.a., the total variation (TV) distance) between two distributions f, f' with same sample space Ω and sigma algebra $\mathcal{S}$ is defined as follows:

$$||f - f'||_{\mathsf{TV}} \triangleq \sup_{\mathcal{A} \in \mathcal{S}} \Big(f(\mathcal{A}) - f'(\mathcal{A}) \Big).$$

The minimum pairwise TV distance of a class of distribution appears naturally in the expressions of statistical error rates related to the class, most notably in the Neyman-Pearson approach to hypothesis testing [102, 117], as well as in the sample complexity results in density estimation [51]. In particular, in these applications, a lower bound on the total variation distance between two candidate distributions is an essential part of the algorithm design and analysis. Our key technical insight is that we can relate the total variation distance between two candidate mixtures to a certain Littlewood polynomial, and then use complex analytic techniques to establish separation in TV-distance. With this separation result, we can use density estimation

techniques (specifically proper learning techniques) to find a candidate mixture that is close in TV-distance to the data generating mechanism. The results we obtain via this approach are labeled as "analytic" in Table 2.1. This approach has recently led to important advances in the trace reconstruction and population recovery problems; see work by [45], [116], and [46].

Our second approach for parameter estimation in general mixtures is based on the *method of moments*, a popular approach for learning Gaussian mixtures, and is more algebraic. Roughly, these algorithms are based on expressing moments of the mixture model as polynomials of the component parameters, and then solving a polynomial system using estimated moments. This approach has been studied in some generality by [18] who show that it can succeed for a large class of mixture models. However, as their method uses non-constructive arguments from algebraic geometry it cannot be used to bound how many moments are required, which is essential in determining the sample complexity; see a discussion in [113, Section 7.6]. In contrast, our approach does yield bounds on how many moments suffice and can be seen as a quantified version of the results in [18]. The results we obtain via this approach are labeled as "algebraic" in Table 2.1. The first set of results of this chapter can be found in more details in [99].

In the second part of this chapter, we specifically focus on mixtures of Gaussians with two components and shared component variance. Again, our main approach considers obtaining lower bounds on the total variation distance by examining the characteristic function of the mixture. Although this connection has been shown in the first set of results in this chapter, it required strict assumptions on the mixtures having discrete parameter values, i.e., Gaussians with means that belong to a scaled integer lattice. The drawback here is that the results present in Table 2.1 only applies to a very restricted set of Gaussian mixtures, and it is not clear how to generalize the previous techniques to non-integer means. As a first step towards that generalization, we analyze

unrestricted two-component one-dimensional mixtures by applying a novel and more direct analysis of the characteristic function. Then, in the high-dimensional setting, we obtain a new total variation distance lower bound by projecting and then using our one-dimensional result. By carefully choosing and analyzing the one-dimensional projection (which depends on the mixtures), we exhibit nearly-tight bounds on the TV distance of d-dimensional mixtures for any $d \geq 1$. In one dimension, we also our lower bounds on total variation distance to provide sample complexity guarantees for parameter estimation.

In the third part of this chapter, we study the sparse trace reconstruction problem where the goal is to reconstruct a sparse unknown string $\boldsymbol{x} \in \{0,1\}^n$ given a set of random subsequences of $\boldsymbol{x}$. Each subsequence, or "trace", is generated by passing $\boldsymbol{x}$ through the *deletion channel* in which each entry of $\boldsymbol{x}$ is deleted independently with probability p. The locations of the deletions are not known; if they were, the channel would be an *erasure channel*. The central question is to find how many traces are required to exactly reconstruct $\boldsymbol{x}$ with high probability. Using results for parameter estimation in binomial mixtures proved in the first part of this chapter, we show new guarantees in this problem. The results in this chapter can be found in [99], [42] and [98].

Organization: The rest of the chapter is organized as follows: In Section 2.2, we provide an overview of the two different general frameworks that we propose namely 1) using complex analysis (Section 2.2.1) and 2) method of moments (Section 2.2.2) for estimating parameters for many different mixtures of canonical distributions. In Section 2.3, we focus on mixtures of Gaussians with two components and shared component co-variance. In particular, in Section 2.3.1, we provide an overview of the proofs. In Section 2.3.2 and 2.3.3, we provide detailed proofs of lower bounds on the total variation distance between two mixtures in the one-dimensional and high-dimensional setting respectively. In Section 2.3.4, we provide details on parameter

estimation using our TV distance lower bounds via the minimum distance estimator and in Section 2.3.5, we provide efficient algorithms for parameter estimation via the Scheffe estimator. Finally, in Section 2.4, we introduce and provide results for the problem of sparse trace reconstruction. All the missing proofs in this chapter can be found in the Appendix in Chapter A.

2.2 Our Techniques and Results

As mentioned, an overview of our sample complexity results are displayed in Table 2.1, where in all cases we consider a uniform mixture of k distributions. Our guarantees are for *exact* parameter estimation, under the assumption that the mixture parameters are discretized to a particular resolution, given in the third column of the table. Theorem statements are given in the sequel.

At first glance the guarantees seem weak, since they all involve exponential dependence in problem parameters. However, except for the Gaussian case, these results are the first guarantees for parameter estimation for these distributions. All prior results we are aware of consider density estimation [33, 63].

For the mixtures of discrete distributions, such as binomial and negative binomial with shared trial parameter, or Poisson/geometric/chi-squared mixtures with certain discretizations, it seems like the dependence of sample complexity on the number of components k is polynomial (see Table 2.1). Note that for these examples $k \leq N$, the upper bounds on parameter values. Therefore the actual dependence on k can still be interpreted as exponential. The results are especially interesting when k is large and possibly growing with N.

For Gaussian mixtures, the most interesting aspect of our bound is the polynomial dependence on the number of components k (first row of Table 2.1). In our setting and taking $\sigma = 1$, the result of [114] is applicable, and it yields $\epsilon^{-O(k)}$ sample complexity, which is incomparable to our $k^3 \exp(O(\epsilon^{-2/3}))$ bound. Note that our

result avoids an exponential dependence in k, trading this off for an exponential dependence on the discretization/accuracy parameter ϵ.* Other results for Gaussian mixtures either 1) consider density estimation [39, 63], which is qualitatively quite different from parameter estimation, 2) treat k as constant [72, 88], or 3) focus on the high dimensional setting and require separation assumptions (see for example [54] and [113]).

As such, our results reflect a new sample complexity tradeoff for parameter estimation in Gaussian mixtures.

As another note, using ideas from [116, 45], one can show that the analytic result for Binomial mixtures is optimal. This raises the question of whether the other results are also optimal or is learning a Binomial mixture intrinsically harder than learning, e.g., a Poisson or Gaussian mixture?

As a final remark, our assumption that parameters are discretized is related to separation conditions that appear in the literature on learning Gaussian mixtures. However, our approach does not seem to yield guarantees when the parameters do not exactly fall into the discretization. In the second part of this chapter, we resolve this shortcoming for the specific case of Gaussian mixtures with two components and share component co-variance matrix.

Our techniques: To establish these results, we take two loosely related approaches. In our analytic approach, the key structural result is to lower bound the total variation distance between two mixtures $\mathcal{M}, \mathcal{M}'$ by a certain Littlewood polynomial. For each distribution type, if the parameter is θ, we find a function $G_t : \mathbb{R} \to \mathbb{C}$ such that

$$\mathbb{E}[G_t(X)] = \exp(it\theta).$$

*Due to our discretization structure, our results do not contradict the lower bounds of [114, 72].

Distribution	Pdf/Pmf $f(x;\theta)$	Discretization	Sample Complexity	Approach
Gaussian	$\frac{1}{\sqrt{2\pi}\sigma}e^{-\frac{(x-\mu)^2}{\sigma^2}}$	$\mu_i \in \epsilon\mathbb{Z}$	$k^3\exp(O((\sigma/\epsilon)^{2/3}))$	Analytic
Binomial	$\binom{n}{x}p^x(1-p)^{n-x}$	$n_i \in \{1,2,\ldots,N\}$	$\exp(O(((N/p)^{1/3})))$	Analytic †
		$p_i \in \{0,\epsilon,\ldots,1\}$	$O(k^2(n/\epsilon)^{8/\sqrt{\epsilon}})$	Algebraic
Poisson	$\frac{\lambda^x e^{-\lambda}}{x!}$	$\lambda_i \in \{0,1,\ldots,N\}$	$\exp(O(N^{1/3}))$	Analytic
Geometric	$(1-p)^x p$	$1/p_i \in \{1,\ldots,N\}$	$O(k^2(\sqrt{N})^{8\sqrt{N}})$	Algebraic
		$p_i \in \{0,\epsilon,\ldots,1\}$	$O(\frac{k^2}{\epsilon^{8/\sqrt{\epsilon}+2}}\log\frac{1}{\epsilon})$	Algebraic
χ^2	$\frac{x^{n/2-1}e^{-x/2}}{2^{n/2}\Gamma(n/2)}$	$n_i \in \{0,1,\ldots,N\}$	$\exp(O(N^{1/3}))$	Analytic
Negative Binomial	$\binom{x+r-1}{x}(1-p)^r p^x$	$r_i \in \{1,2,\ldots,N\}$	$\exp(O((N/p)^{1/3}))$	Analytic

Table 2.1: Overview of our results. Results are given for uniform mixtures of k different components but some can be extended to non-uniform mixtures. Note that for rows 2, 4, 7, and 8, k does not appear. This is because $k \leq N$ and other terms dominate.

(For Gaussians, G_t is essentially the characteristic function). Such functions can be used to obtain Littlewood polynomials from the difference in expectation for two different mixtures, for example if the parameters θ are integral and the mixture weights are uniform. Applying complex analytic results on Littlewood polynomials, this characterization yields a lower bound on the total variation distance between mixtures, at which point we may use density estimation techniques for parameter learning. Specifically we use the minimum distance estimator (see, [51, Sec. 6.8]), which is based on the idea of Scheffe sets. Scheffe sets are building blocks of the Scheffe estimator, commonly used in density estimation, e.g. [135].

Our algebraic approach is based on the more classical method of moments. Our key innovation here is a combinatorial argument to bound the number of moments that we

†We obtained this result as a byproduct of sparse trace reconstruction [98]. In fact, the present work was motivated by the observation that the technique we were using there is much more general.

need to estimate in order to exactly identify the correct mixture parameters. In more detail, when the parameters belong to a discrete set, we show that the moments reveal various statistics about the multi-set of parameters in the mixture. Then, we adapt and extend classical combinatorics results on sequence reconstruction to argue that two distinct multi-sets must disagree on a low-order moment. These combinatorial results are related to the Prouhet-Tarry-Escott problem (see, e.g., [25]) which also has connections to Littlewood polynomials. To wrap up we use standard concentration arguments to estimate all the necessary moments, which yields the sample complexity guarantees.

We note that the complex analytic technique provides non-trivial result only for those mixtures for which an appropriate function G_t exists. On the other hand, the algebraic approach works for all mixtures whose ℓ^{th} moment can be described as a polynomial of degree exactly ℓ in its unknown parameters. In [18], it was shown that most distributions have this later property. In general, where both methods can be applied, the complex analytic techniques typically provide tighter sample complexity bounds than the algebraic ones.

2.2.1 Learning Mixtures via Characteristic Functions

In this section, we show how analysis of the characteristic function can yield sample complexity guarantees for learning mixtures. At a high level, the recipe we adopt is the following.

1. First, we show that, in a general sense, the total variation distance between two separated mixtures is lower bounded by the L_∞ norm of their characteristic functions.

2. Next, we use complex analytic methods and specialized arguments for each particular distribution to lower bound the latter norm.

3. Finally, we use the minimum distance estimator [51] to find a mixture that is close in total variation to the data generating distribution. Using uniform convergence arguments this yields exact parameter learning.

The two main results we prove in this section are listed below.

Theorem 2.1 (Learning Gaussian mixtures). *Let $\mathcal{M} = \frac{1}{k}\sum_{i=1}^{k}\mathcal{N}(\mu_i, \sigma^2)$ be a uniform mixture of k univariate Gaussians, with known shared covariance σ^2 and with distinct means $\mu_i \in \epsilon\mathbb{Z}$. Then there exists an algorithm that requires $k^3\exp(O((\sigma/\epsilon)^{2/3}))$ samples from $\mathcal{M}$ and exactly identifies the parameters $\{\mu_i\}_{i=1}^{k}$ with high probability.*

Theorem 2.2 (Learning Poisson mixtures). *Let $\mathcal{M} = \frac{1}{k}\sum_{i=1}^{k} Poi(\lambda_i)$ where $\lambda_i \in \{0, 1, \ldots, N\}$ for each i are distinct. Then there exists an algorithm that that requires $\exp(O(N^{1/3})))$ samples from $\mathcal{M}$ to exactly identify the parameters $\{\lambda_i\}_{i=1}^{k}$ with high probability.*

There are some technical differences in deriving the results for Gaussian vs Poisson mixtures. Namely, because of finite choice of parameters we can take a union bound over the all possible incorrect mixtures for the latter case, which is not possible for Gaussian. For Gaussian mixtures we instead use an approach based on VC dimension. The results for negative binomial mixtures and chi-squared mixtures (shown in Table 2.1) follow the same route as the Poisson mixture. As reported in Table 2.1, this approach also yields results for mixtures of binomial distributions that we obtained in a different context in our prior work [98].

Total Variation and Characteristic Functions: Let $\{f_\theta\}_{\theta\in\Theta}$ denote a parameterized family of distributions over a sample space $\Omega \subset \mathbb{R}$, where f_θ denotes either a pdf or pmf, depending on the context. We call $\mathcal{M}$ a (finite) Θ-*mixture* if $\mathcal{M}$ has pdf/pmf $\sum_{\theta\in\mathcal{A}}\alpha_\theta f_\theta$ and $\mathcal{A} \subset \Theta, |\mathcal{A}| = k$. For a distribution with density f (we use distribution and density interchangeably in the sequel), define the *characteristic function* $C_f(t) \equiv \mathbb{E}_{X\sim f}[e^{itX}]$. For any two distribution f, f' defined over a

sample space $\Omega \subseteq \mathbb{R}$ the variational distance (or the TV-distance) is defined to be $\|f - f'\|_{\mathrm{TV}} \equiv \frac{1}{2}\int_\Omega \left|\frac{df'}{df} - 1\right| df$. For a function $G : \Omega \to \mathbb{C}$ define the L_∞ norm to be $\|G\|_\infty = \sup_{\omega\in\Omega} |G(\omega)|$ where $|\cdot|$ denotes the modulus.

As a first step, our aim is to show that the total variation distance between $\mathcal{M} = \sum_{\theta\in\mathcal{A}} \alpha_\theta f_\theta$ and any other mixture $\mathcal{M}'$ given by $\sum_{\theta\in\mathcal{B}} \beta_\theta f_\theta, \mathcal{B} \subset \Theta, |\mathcal{B}| = k$ is lower bounded. The following elementary lemma completes the first step of the outlined approach.

Lemma 2.1. *For any two distributions f, f' defined over the same sample space $\Omega \subseteq \mathbb{R}$, we have*

$$\|f - f'\|_{TV} \geq \frac{1}{2} \sup_{t\in\mathbb{R}} |C_f(t) - C_{f'}(t)|.$$

More generally, for any $G : \Omega \to \mathbb{C}$ and $\Omega' \subset \Omega$ we have

$$\|f - f'\|_{TV} \geq \left(2 \sup_{x\in\Omega'} |G(x)|\right)^{-1} \Big(|\mathbb{E}_{X\sim f} G(X) - \mathbb{E}_{X\sim f'} G(X)| - \int_{x\in\Omega\setminus\Omega'} |G(x)| \cdot |df(x) - df'(x)| \Big).$$

Proof. We prove the latter statement, which implies the former since for the function $G(x) = e^{itx}$ we have $\sup_x |G(x)| = 1$. We have

$$\begin{aligned} |\mathbb{E}_{X\sim f} G(X) - \mathbb{E}_{X\sim f'} G(X)| &\leq \int_{x\in\Omega} |G(x)| \cdot |df(x) - df'(x)| \\ &\leq 2 \sup_{x\in\Omega'} |G(x)| \cdot \|f - f'\|_{\mathrm{TV}} + \int_{x\in\Omega\setminus\Omega'} |G(x)| \cdot |df(x) - df'(x)|. \end{aligned}$$ □

Equipped with the lower bound in Lemma 2.1, for each type of distribution, we set out to find a good function G to witness separation in total variation distance. As

we will see shortly, for a parametric family f_θ, it will be convenient to find a family of functions G_t such that

$$\mathbb{E}_{X\sim f_\theta}[G_t(X)] = \exp(it\theta).$$

Of course, to apply Lemma 2.1, it will also be important to understand $\|G_t\|_\infty$. While such functions are specific to the parametric model in question, the remaining analysis will be unified. We derive such functions and collect the relevant properties in the following lemma. At a high level, the calculations are based on reverse engineering from the characteristic function, e.g., finding a choice $t'(t)$ such that $C_f(t') = \exp(it\theta)$.

Lemma 2.2. *Let $z = \exp(it)$ where $t \in [-\pi/L, \pi/L]$.*

- *Gaussian. If $X \sim \mathcal{N}(\mu, \sigma)$ and $G_t(x) = e^{itx}$ then*

$$\mathbb{E}[G_t(X)] = \exp(-\sigma^2 t^2/2) z^\mu \text{ and } \|G_t\|_\infty = 1 \ .$$

- *Poisson. If $X \sim Poi(\lambda)$ and $G_t(x) = (1+it)^x$ then*

$$\mathbb{E}[G_t(X)] = z^\lambda \text{ and } |G_t(x)| \le (1+t^2)^{x/2} \ .$$

- *Chi-Squared. If $X \sim \chi^2(\ell)$ and $G_t(x) = \exp(x/2 - xe^{-2it}/2)$ then*

$$\mathbb{E}[G_t(X)] = z^\ell \text{ and } |G_t(x)| \le e^{cxt^2 + O(xt^4)} \ .$$

- *Negative Binomial. If $X \sim NB(r,p)$ and $G_t(x) = (1/p - (1/p-1)e^{-it})^x$ then*

$$\mathbb{E}[G_t(X)] = z^r \text{ and } |G_t(x)| \le e^{-cx\frac{(1-p)t^2}{p^2}} \ .$$

Proof. We consider each distribution in turn:

- *Poisson:* For Poisson random variables, if $G_t(x) = (1+it)^x$ then since $|1+it|^2 = 1+t^2$ the second claim follows. For the first:

$$\mathbb{E}[G_t(X)] = \exp(\lambda((1+it)-1)) = z^\lambda. \qquad \square$$

- *Gaussian:* Observe that $\mathbb{E}[G_t(X)]$ is precisely the characteristic function. Clearly we have $\|G_t\|_\infty = 1$ and further

$$\mathbb{E}[G_t(X)] = \exp(it\mu - \sigma^2 t^2/2) = \exp(-\sigma^2 t^2/2) z^\mu.$$

- *Chi-Squared:* Let $w_t = \exp(1/2 - e^{-2it}/2)$ then

$$|w_t|^2 = |e^{1-e^{-2it}}| = |e^{1-\cos 2t} e^{i \sin 2t}| \leq e^{ct^2 + O(t^4)}$$

  and

$$\mathbb{E}[G_t(X)] = (1 - 2\ln w_t)^{-\frac{\ell}{2}} = z^\ell.$$

- *Negative Binomial:* Let $w_t = 1/p - (1/p-1)e^{-it}$ then $|w_t|^2 = \frac{1+(1-p)^2-2(1-p)\cos t}{p^2} = \frac{p^2+4(1-p)\sin^2(t/2)}{p^2} \leq e^{\frac{(1-p)t^2}{p^2}}$ and

$$\mathbb{E}[G_t(X)] = \Big(\frac{1-p}{1-pw_t}\Big)^r = z^r.$$

Next, we crucially use the following lemma.

Lemma 2.3 ([24])**.** *Let $a_0, a_1, a_2, \cdots \in \{0, 1, -1\}$ be such that not all of them are zero. For any complex number z, let $A(z) \equiv \sum_k a_k z^k$. Then, for some absolute constant c,*

$$\max_{-\pi/L \leq t \leq \pi/L} |A(e^{it})| \geq e^{-cL} \ .$$

We will also need the following 'tail bound' lemma.

Lemma 2.4. *Suppose $a > 1$ is any real number and $r \in \mathbb{R}_+$. For any discrete random variable X with support $\mathbb{Z}$ and pmf f,*

$$\sum_{x \geq r} a^x f(x) \leq \frac{\mathbb{E}[a^{2X}]}{a^{r-1}}.$$

Proof. Note that, $\Pr(X \geq x) = \Pr(a^{2X-2x} \geq 1) \leq \mathbb{E}[a^{2X-2x}]$. We have,

$$\sum_{x \geq r} a^x \Pr(X = x) \leq \sum_{x \geq r} a^x \Pr(X \geq x) \leq \sum_{x \geq r} a^x \mathbb{E}[a^{2X-2x}] = \mathbb{E}[a^{2X}] \sum_{x \geq r} a^{-x} \leq \frac{\mathbb{E}[a^{2X}]}{a^{r-1}}.$$

□

Subsequently, we can show the following lower bound on the total variation distance:

Theorem 2.3 (TV Lower Bounds)**.** *The following bounds hold on distance between two different mixtures assuming all k parameters are distinct for each mixture.*

- Gaussian: $\mathcal{M} = \frac{1}{k}\sum_{i=1}^{k} \mathcal{N}(\mu_i, \sigma)$ *and* $\mathcal{M}' = \frac{1}{k}\sum_{i=1}^{k} \mathcal{N}(\mu'_i, \sigma)$ *where* $\mu_i, \mu'_i \in \epsilon\mathbb{Z}$. *Then*

$$\|\mathcal{M}' - \mathcal{M}\|_{TV} \geq k^{-1} \exp(-\Omega((\sigma/\epsilon)^{2/3})) \ .$$

- Poisson: $\mathcal{M} = \frac{1}{k}\sum_{i=1}^{k} Poi(\lambda_i)$ *and* $\mathcal{M}' = \frac{1}{k}\sum_{i=1}^{k} Poi(\lambda'_i)$ *where* $\lambda_i, \lambda'_i \in \{0, 1, \ldots, N\}$. *Then*

$$\|\mathcal{M}' - \mathcal{M}\|_{TV} \geq k^{-1} \exp(-\Omega(N^{1/3})) \ .$$

- Chi-Squared: $\mathcal{M} = \frac{1}{k}\sum_{i=1}^{k} \chi^2(\ell_i)$ *and* $\mathcal{M}' = \frac{1}{k}\sum_{i=1}^{k} \chi^2(\ell'_i)$ *where* $\ell_i, \ell'_i \in \{1, \ldots, N\}$. *Then*

$$\|\mathcal{M}' - \mathcal{M}\|_{TV} \geq k^{-1} \exp(-\Omega(N^{1/3})) \ .$$

- Negative Binomial: $\mathcal{M} = \frac{1}{k}\sum_{i=1}^{k} NB(r_i, p)$ *and* $\mathcal{M}' = \frac{1}{k}\sum_{i=1}^{k} NB(r'_i, p)$ *where* $r_i, r'_i \in \{1, 2, \ldots, N\}$. *Then*

$$\|\mathcal{M}' - \mathcal{M}\|_{TV} \geq k^{-1} \exp(-\Omega((N/p)^{1/3})) \ .$$

Proof. • *Poisson:*

Let $X \sim \mathcal{M}$ and $X' \sim \mathcal{M}'$. Then, for $w = 1 + it$, from Lemma 2.2,

$$\mathbb{E}(w^X) - \mathbb{E}(w^{X'}) = \frac{1}{k}\sum_{j=1}^{k}(e^{it\lambda_j} - e^{it\lambda'_j}).$$

Now we use Lemma 2.1 with $G(x) = w^x$, $\Omega' = \{0, 1, \ldots, 2N\}$ and $t \leq 1$, to have,

$$\begin{aligned}
|\mathbb{E}(w^X) - \mathbb{E}(w^{X'})| = \Big|\sum_x (w^x\mathcal{M}(x) - w^x\mathcal{M}'(x))\Big| &\leq \sum_x |w|^x|\mathcal{M}(x) - \mathcal{M}'(x)| \\
&\leq (1+t^2)^{2N}\sum_x |\mathcal{M}(x) - \mathcal{M}'(x)| + \sum_{x>4N}(1+t^2)^{x/2}e^{-N}\frac{N^x}{x!} \\
&\leq (1+t^2)^{2N}\sum_x |\mathcal{M}(x) - \mathcal{M}'(x)| + \sum_{x>4N}2^{x/2}e^{-N}\frac{N^x}{x!}.
\end{aligned}$$

Now using Lemma 2.4,

$$\begin{aligned}
&|\mathbb{E}(w^X) - \mathbb{E}(w^{X'})| \leq 2(1+t^2)^{2N}\left\|\mathcal{M} - \mathcal{M}'\right\|_{\mathrm{TV}} + \frac{\mathbb{E}[2^X]}{2^{2N-1/2}} \\
&\leq 2(1+t^2)^{N}\left\|\mathcal{M} - \mathcal{M}'\right\|_{\mathrm{TV}} + e^{N(\sqrt{1+t^2}-1)}\frac{e^{-N\sqrt{1+t^2}}(en\sqrt{1+t^2})^{2N}}{(2N)^{2N}} \\
&\leq 2e^{2t^2N}\left\|\mathcal{M} - \mathcal{M}'\right\|_{\mathrm{TV}} + \frac{e^N}{2^{2N-1/2}} \\
&= 2e^{2\pi^2N/L^2}\left\|\mathcal{M} - \mathcal{M}'\right\|_{\mathrm{TV}} + \exp(-\Omega(N)),
\end{aligned}$$

by taking $|t| \leq \frac{\pi}{L}$. Now using Lemma 2.3, there exist an absolute constant c such that,

$$\max_{-\frac{\pi}{L}\leq t\leq\frac{\pi}{L}}\Big|\sum_{j=1}^{k}(e^{it\lambda_j} - e^{it\lambda'_j})\Big| \geq e^{-cL}.$$

Therefore by setting $L = N^{1/3}$,

$$\left\|\mathcal{M} - \mathcal{M}'\right\|_{\mathrm{TV}} \geq (2k)^{-1}e^{-cL-2\pi^2N/L^2} - \exp(-\Omega(N)) \geq k^{-1}\exp(-\Omega(N^{1/3})).$$

□

- *Gaussian:* The characteristic function of a Gaussian $X \sim \mathcal{N}(\mu, \sigma^2)$ is

$$C_{\mathcal{N}}(t) = \mathbb{E}e^{itX} = e^{it\mu - \frac{t^2\sigma^2}{2}}.$$

Therefore we have that

$$C_{\mathcal{M}}(t) - C_{\mathcal{M}'}(t) \geq \frac{e^{-\frac{t^2\sigma^2}{2}}}{k} \sum_{j=1}^{k} (e^{it\mu_j} - e^{it\mu'_j}).$$

Now, using Lemma 2.3, there exist an absolute constant c such that,

$$\max_{-\frac{\pi}{\epsilon L} \leq t \leq \frac{\pi}{\epsilon L}} \Big| \sum_{j=1}^{k} (e^{it\mu_j} - e^{it\mu'_j}) \Big| \geq e^{-cL}.$$

Also, for $t \in (-\frac{\pi}{\epsilon L}, \frac{\pi}{\epsilon L})$, $e^{-\frac{t^2\sigma^2}{2}} \geq e^{-\frac{\sigma^2\pi^2}{2\epsilon^2 L^2}}$. And therefore,

$$\Big| C_{\mathcal{M}}(t) - C_{\mathcal{M}'}(t) \Big| \geq \frac{1}{k} e^{-\frac{\sigma^2\pi^2}{2\epsilon^2 L^2} - cL}.$$

By substituting $L = \frac{(\pi\sigma)^{2/3}}{(\epsilon^2 c)^{1/3}}$ above we conclude that there exists t such that

$$\Big| C_{\mathcal{M}}(t) - C_{\mathcal{M}'}(t) \Big| \geq \frac{1}{k} e^{-\frac{3}{2}(c\pi\sigma/\epsilon)^{2/3}}.$$

Now using Lemma 2.1, we have $\|\mathcal{M}' - \mathcal{M}\|_{\mathrm{TV}} \geq k^{-1} \exp(-\Omega((\sigma/\epsilon)^{2/3}))$.

- *Chi-Squared:* Let $X \sim \mathcal{M}$ and $X' \sim \mathcal{M}'$. Then, for $w = \exp(1/2 - e^{-2it}/2)$, from Lemma 2.2,

$$\mathbb{E}(w^X) - \mathbb{E}(w^{X'}) = \frac{1}{k} \sum_{j=1}^{k} (e^{it\ell_j} - e^{it\ell'_j}).$$

Now we use Lemma 2.1, with $\Omega' = [0, 2N]$ we have,

$$\|\mathcal{M} - \mathcal{M}'\|_{TV} \geq e^{-2ct^2N} \Big(\Big| \mathbb{E}(w^X) - \mathbb{E}(w^{X'}) \Big| - \int_{x>2N} \exp(ct^2x) f(x) dx \Big),$$

where $f \sim \chi^2(N)$. We have,

$$\int_{x>2N} \exp(ct^2x)f(x)dx = \frac{1}{(1-2ct^2)^{N/2-1}} \int_{y>2N(1-2ct^2)} f(y)dy \leq \frac{e^{-N(1-4ct^2)^2/8}}{(1-2ct^2)^{N/2-1}}$$
$$\leq \exp(-\Omega(N)),$$

where we have used the pdf of chi-squared distribution and the tail bounds for chi-squared. Now using Lemma 2.3, and taking $|t| \leq \frac{\pi}{L}$,

$$\|\mathcal{M} - \mathcal{M}'\|_{TV} \geq k^{-1}e^{-c'L-2ct^2N} - \exp(-\Omega(n))$$
$$\geq k^{-1}\exp(-c'L - 2\pi^2N/L^2) - \exp(-\Omega(N)).$$

Again setting, $L = N^{1/3}$,

$$\|\mathcal{M} - \mathcal{M}'\|_{TV} \geq k^{-1}\exp(-\Omega(N^{1/3})).$$

- *Negative-Binomial:* Let $X \sim \mathcal{M}$ and $X' \sim \mathcal{M}'$. Then, for $w = 1/p-(1/p-1)e^{-it}$, from Lemma 2.2, taking $G(x) = w^x$,

$$\mathbb{E}(w^X) - \mathbb{E}(w^{X'}) = \frac{1}{k}\sum_{j=1}^{k}(e^{itr_j} - e^{itr'_j}).$$

Now we use Lemma 2.1, with $\Omega' = [0, 6pN/(1-p)]$ we have,

$$\|\mathcal{M} - \mathcal{M}'\|_{TV} \geq e^{-12ct^2N/p}\Big(\Big|\mathbb{E}(w^X) - \mathbb{E}(w^{X'})\Big| - \sum_{x>\frac{6Np}{1-p}} |w|^x u(x)\Big),$$

where $u(x) = \binom{x+N-1}{x}(1-p)^N p^x$. We have $|w| \leq e^{c(1-p)t^2/p^2} \leq e^{c(1-p)/p^2}$ for $t < 1$. Using Lemma 2.4, with $X \sim NB(N,p)$, we have,

$$\sum_{x>\frac{6Np}{1-p}} \exp(cx(1-p)/p^2)u(x) \leq a^{1-\frac{6Np}{1-p}}\mathbb{E}[a^{2X}] = a^{1-\frac{6Np}{1-p}}\Big(\frac{1-p}{1-pa^2}\Big)^N = \exp(-\Omega(N)),$$

where, $a = \exp(c(1-p)/p^2) > 1$. Now using Lemma 2.3, and taking $|t| \leq \frac{\pi}{L}$,

$$\begin{aligned}\|\mathcal{M} - \mathcal{M}'\|_{TV} &\geq k^{-1} e^{-c'L - 12ct^2 N/p} - \exp(-\Omega(n)) \\ &\geq k^{-1} \exp(-c'L - 12\pi^2 N/(pL^2)) - \exp(-\Omega(N)).\end{aligned}$$

Setting $L = (N/p)^{1/3}$,

$$\|\mathcal{M} - \mathcal{M}'\|_{TV} \geq k^{-1} \exp(-\Omega((N/p)^{1/3})).$$

Parameter Learning:

Union Bound Approach for Discrete Distributions: We begin with the following proposition which follows from Theorem 7.1 of [51].

Lemma 2.5. *Suppose $F = \{f_\nu\}_{\nu \in \Theta}$ is a class of distribution such that for any $\nu, \nu' \in \Theta$, $\|f_\nu - f_{\nu'}\|_{TV} \geq \delta$. Then $O(\log|\Theta|/\delta^2)$ samples from a distribution f in F suffice to distinguish it from all other distributions in F with high probability.*

For the mixture of Poissons, $\mathcal{M} = \frac{1}{k}\sum_{i=1}^{k} \text{Poi}(\lambda_i)$ where $\lambda_i \in \{0, 1, \ldots, N\}$, the number of choices for parameters in the mixture is $(N+1)^k$. Now using Lemmas 2.3 and 2.5, $\exp(O(N^{1/3}))$ samples are sufficient to learn the parameters of the mixture.

Exactly the same argument applies to mixtures of Chi-Squared and Negative-Binomial distributions, yielding $\exp(O(N^{1/3}))$ and $\exp(O((N/p)^{1/3}))$ samples suffice, respectively. However, for Gaussians we need a more intricate approach.

VC Approach for Gaussians: To learn the parameters of a Gaussian mixture

$$\mathcal{M} = \frac{1}{k}\sum_{i=1}^{k}\mathcal{N}(\mu_i, \sigma) \quad \text{where} \quad \mu_i \in \{\ldots, -2\epsilon, -\epsilon, 0, \epsilon, 2\epsilon \ldots\}$$

we use the minimum distance estimator precisely defined in [51, Section 6.8]. Let $\mathcal{A} \equiv \{\{x : \mathcal{M}(x) \geq \mathcal{M}'(x)\} : \text{ for any two mixtures } \mathcal{M} \neq \mathcal{M}'\}$ be a collection of subsets. Let P_m denote the empirical probability measure induced by the m samples. Then, choose a mixture $\hat{\mathcal{M}}$ for which the quantity $\sup_{A\in\mathcal{A}} |\Pr_{\sim\hat{\mathcal{M}}}(A) - P_m(A)|$ is minimum (or within $1/m$ of the infimum). This is the minimum distance estimator, whose performance is guaranteed by the following proposition [51, Thm. 6.4].

Proposition 2.1. *Given m samples from $\mathcal{M}$ and with $\Delta = \sup_{A\in\mathcal{A}} |\Pr_{\sim\mathcal{M}}(A) - P_m(A)|$, we have*

$$\left\|\hat{\mathcal{M}} - \mathcal{M}\right\|_{TV} \leq 4\Delta + \frac{3}{m}.$$

We now upper bound the right-hand side of the above inequality. Via McDiarmid's inequality and a standard symmetrization argument, Δ is concentrated around its mean which is a function of $VC(\mathcal{A})$, the VC dimension of the class $\mathcal{A}$, see [51, Section 4.3]:

$$\left\|\hat{\mathcal{M}} - \mathcal{M}\right\|_{\text{TV}} \leq 4\Delta + O(1/m) \leq 4\mathbb{E}_{\sim\mathcal{M}}\Delta + O(1/\sqrt{m}) \leq c\sqrt{\frac{VC(\mathcal{A})}{m}},$$

with high probability, for an absolute constant c. This latter term is bounded by the following.

Lemma 2.6. *For the class $\mathcal{A}$ defined above, the VC dimension is given by $VC(\mathcal{A}) = O(k)$.*

Proof. First of all we show that any element of the set $\mathcal{A}$ can be written as union of at most $4k-1$ intervals in $\mathbb{R}$. For this we use the fact that a linear combination of k Gaussian pdfs $f(x) = \sum_{i=1}^{k}\alpha_i f_i(x)$ where f_is normal pdf $\mathcal{N}(\mu_i, \sigma_i^2)$ and $\alpha_i \in$

$\mathbb{R}, 1 \leq i \leq k$ has at most $2k-2$ zero-crossings [87]. Therefore, for any two mixtures of interest $\mathcal{M}(x) - \mathcal{M}'(x)$ has at most $4k-2$ zero-crossings. Therefore any $A \in \mathcal{A}$ must be a union of at most $4k-1$ contiguous regions in $\mathbb{R}$. It is now an easy exercise to see that the VC dimension of such a class is $\Theta(k)$. □

As a result the error of the minimum distance estimator is $O(\sqrt{k/m})$ with high probability. But from Theorem 2.3, notice that for any other mixture $\mathcal{M}'$ we must have,

$$\|\mathcal{M} - \mathcal{M}'\|_{\text{TV}} \geq k^{-1} \exp(-\Omega((\sigma/\epsilon)^{2/3})).$$

As long as $\left\|\hat{\mathcal{M}} - \mathcal{M}\right\|_{\text{TV}} \leq \frac{1}{2} \|\mathcal{M} - \mathcal{M}'\|_{\text{TV}}$ we will exactly identify the parameters. Therefore $m = k^3 \exp(O((\sigma/\epsilon)^{2/3}))$ samples suffice to exactly learn the parameters with high probability.

Extension to Non-Uniform Mixtures: The above results extend to non-uniform mixtures, where the main change is that we require a generalization of complex analytic tools. The result, also proved by [24], states that if $a_0, a_1, a_2, \ldots \in [-1, 1]$ with poly(n) precision then $\max_{-\pi/L \leq \theta \leq \pi/L} |A(e^{i\theta})| \geq e^{-cL \log n}$, for an absolute constant c. This weaker bound yields an extra poly(n) factor in the sample complexity.

2.2.2 Learning Mixtures via Moments

There are some mixtures where the problem of learning parameters is not amenable to the approach in the previous section. A simple motivating example is learning the parameters $p_i \in \{0, \epsilon, 2\epsilon, 3\epsilon, \ldots, 1\}$ values[‡] in the mixture $\mathcal{M} = \frac{1}{k} \sum_{i=1}^{k} \text{Bin}(n, p_i)$. In this section, we present an alternative procedure for learning such mixtures. The basic idea is as follows:

[‡]Note that we are implicitly assuming $1/\epsilon$ is integral here and henceforth.

- We compute moments $\mathbb{E}X^\ell$ exactly for $\ell = 0, 1, \ldots, T$ by taking sufficiently many samples. The number of samples will depend on T and the precision of the parameters of the mixture.

- We argue that if T is sufficiently large, then these moments uniquely define the parameters of the mixture. To do this we use a combinatorial result due to [96].

In this section, it will be convenient to define a function m_ℓ on multi-sets where

$$m_\ell(A) := \sum_{a \in A} a^\ell \ .$$

Our main result is as follows:

Theorem 2.4 (Learning Binomial mixtures)**.** *Let $\mathcal{M} = \frac{1}{k}\sum_{i=1}^{k} Bin(n, p_i)$ be a uniform mixture of k binomials, with known shared number of trials n and unknown probabilities $p_1, \ldots, p_k \in \{0, \epsilon, 2\epsilon, \ldots, 1\}$. Then, provided $n \geq 4/\sqrt{\epsilon}$, the first $4/\sqrt{\epsilon}$ moments suffice to learn the parameters p_i and there exists an algorithm that, when given $O(k^2(n/\epsilon)^{8/\sqrt{\epsilon}})$ samples from $\mathcal{M}$, exactly identifies the parameters $\{p_i\}_{i=1}^k$ with high probability.*

Computing the Moments: We compute the ℓth moment as $S_{\ell,t} = \sum Y_i^\ell / t$ where $Y_1, \ldots, Y_t \sim X$.

Lemma 2.7. $\Pr[|S_{\ell,t} - \mathbb{E}X^\ell| \geq \gamma] \leq \frac{\mathbb{E}X^{2\ell}}{t\gamma^2} \leq \frac{(2\ell)!}{\gamma^2 t} \inf_\alpha \left(\frac{\mathbb{E}e^{\alpha X}}{\alpha^{2\ell}}\right)$ *where the last inequality assumes the all the moments of X are non-negative.*

Proof. By the Chebyshev bound,

$$\Pr[|S_{\ell,t} - \mathbb{E}X^\ell| \geq \gamma] \leq \frac{Var(S_{\ell,t})}{\gamma^2} = \frac{Var(X^\ell)}{t\gamma^2} \leq \frac{\mathbb{E}X^{2\ell}}{t\gamma^2}.$$

We then use the moment generating function: for all $\alpha > 0$, $\mathbb{E}X^{2\ell} \leq (2\ell)!\mathbb{E}e^{\alpha X}/\alpha^{2\ell}$. $\square$

The following corollary, tailors the above lemma for a mixture of binomial distributions.

Corollary 2.1. *If* $X \sim \sum_{i=1}^{k} \mathrm{Bin}(n, p_i)/k$ *then* $\Pr[|S_{\ell,t} - \mathbb{E}X^\ell| \geq \gamma] = \gamma^{-2} n^{2\ell}/t$.

Fixing n, the ℓ^{th} moment of a mixture of binomial distributions $X \sim \sum_{i=1}^{k} \mathrm{Bin}(n, p_i)/k$ is

$$\mathbb{E}X^\ell = \sum_{i=1}^{k} f(p_i)/k$$

where f is a polynomial of degree at most ℓ with integer coefficients [18]. If p_i is an integer multiple of ϵ then this implies $k(\mathbb{E}X^\ell)/\epsilon^\ell$ is integral and therefore any mixture with a different ℓth moment differs by at least ϵ^ℓ/k. Hence, learning the ℓth moment up to $\gamma_\ell < \epsilon^\ell/(2k)$ implies learning the moment exactly.

Lemma 2.8. *For* $X \sim \mathrm{Bin}(n, p)$, $\mathbb{E}X^\ell$ *is a polynomial in* p *of degree exactly* ℓ *if* $n \geq \ell$.

Proof. We will prove that for $X \sim \mathrm{Bin}(n, p)$, the leading term of $\mathbb{E}X^\ell$ is $\prod_{i=0}^{\ell-1}(n-i)p^\ell$. Since for $n \geq \ell$, $\prod_{i=0}^{\ell-1}(n-i) \neq 0$, this implies that $\mathbb{E}X^\ell$ is a polynomial of degree exactly ℓ. We will prove this by induction. Since $X \sim \mathrm{Bin}(n, p)$, we know that $\mathbb{E}X = np$. This verifies the base case. Now, in the induction step, let us assume that the leading term of $\mathbb{E}X^k$ is $\prod_{i=0}^{k-1}(n-i)p^k$. It is known that (see [18])

$$\mathbb{E}X^{k+1} = np\mathbb{E}X^k + p(1-p)\frac{d\mathbb{E}X^k}{dp}.$$

Therefore it follows that the leading term of $\mathbb{E}X^{k+1}$ is

$$np\prod_{i=0}^{k-1}(n-i)p^k - kp^2\prod_{i=0}^{k-1}(n-i)p^{k-1} = \prod_{i=0}^{k}(n-i)p^{k+1}.$$

This proves the induction step and the lemma.

□

Theorem 2.5. *$O(k^2(n/\epsilon)^{8/\sqrt{\epsilon}})$ samples are sufficient to exactly learn the first $4/\sqrt{\epsilon}$ moments of a uniform mixture of k binomial distributions $\sum_{i=1}^{k} \mathrm{Bin}(n, p_i)/k$ with probability at least $7/8$ where each $p_i \in \{0, \epsilon, 2\epsilon, \ldots, 1\}$.*

Proof. Let $T = 4/\sqrt{\epsilon}$. From Corollary 2.1 and the preceding discussion, learning the ℓth moment exactly with failure probability $1/9^{1+T-\ell}$ requires

$$t = \gamma_\ell^{-2} n^{2\ell} 9^{1+T-\ell} = O(k^2 9^{1+T-\ell} n^{2\ell}/\epsilon^{2\ell}) = O(k^2 9^T (n/3\epsilon)^{2\ell})$$

samples. And hence, we can compute all ℓth moments exactly for $1 \le \ell \le 4/\sqrt{\epsilon}$ using

$$\sum_{\ell=1}^{T} O(k^2 9^T (n/3\epsilon)^{2\ell}) = O(k^2 (n/\epsilon)^{2T})$$

samples with failure probability $\sum_{\ell=1}^{T} 1/9^{1+T-\ell} < \sum_{i=1}^{\infty} 1/9^i = 1/8$. □

How many moments determine the parameters: It remains to show the first $4/\sqrt{\epsilon}$ moments suffice to determine the p_i values in the mixture $X \sim \sum_{i=1}^{k} \mathrm{Bin}(n, p_i)/k$ provided $n \ge \frac{4}{\epsilon}$. To do this suppose there exists another mixture $Y \sim \sum_{i=1}^{k} \mathrm{Bin}(n, q_i)/k$ and we will argue that

$$\mathbb{E}X^\ell = \mathbb{E}Y^\ell \text{ for } \ell = 0, 1, \ldots, 4\sqrt{1/\epsilon}$$

implies $\{p_i\}_{i\in[k]} = \{q_i\}_{i\in[k]}$. To argue this, define integers $\alpha_i, \beta_i \in \{0, 1, \ldots, 1/\epsilon\}$ such at that $p_i = \alpha_i\epsilon$ and $q_i = \beta_i\epsilon$. Let $\mathcal{A} = \{\alpha_1, \ldots, \alpha_k\}$ and $\mathcal{B} = \{\beta_1, \ldots, \beta_k\}$. Then,

$$\mathbb{E}X = \mathbb{E}Y \Longrightarrow \sum_i \alpha_i = \sum_i \beta_i \Longrightarrow m_1(\mathcal{A}) = m_1(\mathcal{B})$$

and, after some algebraic manipulation, it can be shown that for all $\ell \in \{2, 3, \ldots\}$,

$$\left(\forall \ell' \in \{0, 1, \ldots, \ell-1\}\ ,\ \sum_i \alpha_i^{\ell'} = \sum_i \beta_i^{\ell'}\right) \text{ and } \mathbb{E}X^\ell = \mathbb{E}Y^\ell$$

$$\implies \left(\sum_i \alpha_i^\ell = \sum_i \beta_i^\ell \right) \implies m_\ell(\mathcal{A}) = m_\ell(\mathcal{B}) \ .$$

Hence, if the first T moments match $m_\ell(\mathcal{A}) = m_\ell(\mathcal{B})$ for all $\ell = 0, 1, \ldots, T$. But the following theorem establishes that if $T = 4\sqrt{1/\epsilon}$ then this implies $\mathcal{A} = \mathcal{B}$.

Theorem 2.6 ([96]). *For any two subsets S, T of $\{0, 1, \ldots, n-1\}$, then*

$$S = T \text{ iff } \left(m_k(S) = m_k(T) \text{ for all } k = 0, 1, \ldots, 4\sqrt{n} \right) \ .$$

We note that the above theorem is essentially tight. Specifically, there exists $S \neq T$ with $m_k(S) = m_k(T)$ for $k = 0, 1, \ldots, cn/\log n$ for some c. As a consequence of this, we note that even the exact values of the $c\sqrt{n}/\log n$ moments are insufficient to learn the parameters of the distribution. For an example in terms of Gaussian mixtures, even given the promise $\mu_i \in \{0, 1, \ldots, n-1\}$ are distinct, then the first $c\sqrt{n}/\log n$ moments of $\sum_i \mathcal{N}(\mu_i, 1)$ are insufficient to uniquely determine μ_i whereas the first $4\sqrt{n}$ moments are sufficient.

Extension to Non-Uniform Distributions: We now consider extending the framework to non-uniform distributions. In this case, the method of computing the moments is identical to the uniform case. However, when arguing that a small number of moments suffices we can no longer appeal to the Theorem 2.6.

To handle non-uniform distribution we introduce a precision variable q and assume that the weights of the component distributions $\omega_1, \omega_2, \ldots, \omega_k$ are of the form:

$$\omega_i = \frac{w_i}{\sum_{i=1}^k w_i}$$

where $w_i \in \{0, 1, \ldots, q-1\}$. Then, in the above framework if we are trying to learn parameters $\alpha_1, \ldots, \alpha_k$ then the moments are going to define a multi-set consisting of

w_i copies of α_i for each $i \in [k]$. We can quantify how many moments suffice in this case by a relatively straight-forward generalization of proof by [124].

Theorem 2.7. *For any two multi-sets S, T where each element is in $\{0, 1, \ldots, n-1\}$ and the multiplicity of each element is at most $q-1$, then $S = T$ if and only if $m_k(S) = m_k(T)$ for all $k = 0, 1, \ldots, 2\sqrt{qn \log qn}$.*

The proof of this theorem can be found in Chapter A, Section A.1.

2.3 Mixtures of Gaussians with 2 components

Let $\mathcal{N}(\boldsymbol{\mu}, \boldsymbol{\Sigma})$ denote the d-dimensional Gaussian distribution with mean $\boldsymbol{\mu} \in \mathbb{R}^d$ and positive definite covariance matrix $\boldsymbol{\Sigma} \in \mathbb{R}^{d \times d}$. A k-component mixture of d-dimensional Gaussian distributions is a distribution of the form $f = \sum_{i=1}^{k} w_i \cdot \mathcal{N}(\boldsymbol{\mu}_i, \boldsymbol{\Sigma}_i)$. Such a mixture is defined by k triples $\{(w_i, \boldsymbol{\mu}_i, \boldsymbol{\Sigma}_i)\}_{i=1}^{k}$, where $w_i \in \mathbb{R}^+$ with $\sum_{i=1}^{k} w_i = 1$ are the mixing weights, $\boldsymbol{\mu}_i \in \mathbb{R}^d$ are the means, and $\boldsymbol{\Sigma}_i \in \mathbb{R}^{d \times d}$ are the covariance matrices.

Let $\mathcal{F}$ be the set of all d-dimensional, two-component, equally weighted Gaussian mixtures

$$\mathcal{F} = \left\{ f_{\boldsymbol{\mu}_0, \boldsymbol{\mu}_1} = \frac{1}{2}\mathcal{N}(\boldsymbol{\mu}_0, \boldsymbol{\Sigma}) + \frac{1}{2}\mathcal{N}(\boldsymbol{\mu}_1, \boldsymbol{\Sigma}) \mid \boldsymbol{\mu}_0, \boldsymbol{\mu}_1 \in \mathbb{R}^d, \boldsymbol{\Sigma} \in \mathbb{R}^{d \times d} \right\},$$

where $\boldsymbol{\Sigma} \in \mathbb{R}^{d \times d}$ is a positive definite matrix. When $d = 1$, we use the notation $f_{\mu_0, \mu_1} \in \mathcal{F}$ and simply denote the variance as $\sigma^2 \in \mathbb{R}$. We also define the following:

Definition 2.1. *For any $\delta' > 0$ and $0 < \eta < 1$, we define a (δ', η)-estimate of the unknown distribution f to be another mixture of two component Gaussians*

$$\frac{\mathcal{N}(\hat{\boldsymbol{\mu}}_0, \sigma^2)}{2} + \frac{\mathcal{N}(\hat{\boldsymbol{\mu}}_1, \sigma^2)}{2}$$

such that for some permutation $\pi : \{0,1\} \to \{0,1\}$*, for* $i = 0, 1$

$$\left|\left|\hat{\boldsymbol{\mu}}_i - \boldsymbol{\mu}_{\pi(i)}\right|\right|_2 \leq \delta'$$

with probability $1 - \eta$*.*

Our main result is the following nearly-tight lower bound on the TV distance between pairs of d-dimensional two-component mixtures with shared covariance.

Theorem 2.8. *For* $f_{\boldsymbol{\mu}_0,\boldsymbol{\mu}_1}, f_{\boldsymbol{\mu}'_0,\boldsymbol{\mu}'_1} \in \mathcal{F}$*, define sets* $S_1 = \{\boldsymbol{\mu}_1 - \boldsymbol{\mu}_0, \boldsymbol{\mu}'_1 - \boldsymbol{\mu}'_0\}$*,* $S_2 = \{\boldsymbol{\mu}'_0 - \boldsymbol{\mu}_0, \boldsymbol{\mu}'_1 - \boldsymbol{\mu}_1\}$*,* $S_3 = \{\boldsymbol{\mu}'_0 - \boldsymbol{\mu}_1, \boldsymbol{\mu}'_1 - \boldsymbol{\mu}_0\}$ *and vectors* $\boldsymbol{v}_1 = \mathsf{argmax}_{s \in S_1} ||\boldsymbol{s}||_2$*,* $\boldsymbol{v}_2 = \mathsf{argmax}_{s \in S_2} ||\boldsymbol{s}||_2$ *and* $\boldsymbol{v}_3 = \mathsf{argmax}_{s \in S_3} ||\boldsymbol{s}||_2$*. Let* $\lambda_{\Sigma,\mathcal{U}} \triangleq \max_{\boldsymbol{u}:||\boldsymbol{u}||_2=1, \boldsymbol{u} \in \mathcal{U}} \boldsymbol{u}^T \Sigma \boldsymbol{u}$ *with* $\mathcal{U}$ *being the span of the vectors* $\boldsymbol{v}_1, \boldsymbol{v}_2, \boldsymbol{v}_3$*. If* $\|\boldsymbol{v}_1\|_2 \geq \min(\|\boldsymbol{v}_2\|_2, \|\boldsymbol{v}_3\|_2)/2$ *and* $\sqrt{\lambda_{\Sigma,\mathcal{U}}} = \Omega(||\boldsymbol{v}_1||_2)$*, then*

$$\left|\left|f_{\boldsymbol{\mu}_0,\boldsymbol{\mu}_1} - f_{\boldsymbol{\mu}'_0,\boldsymbol{\mu}'_1}\right|\right|_{\mathsf{TV}} = \Omega\Big(\min\Big(1, \frac{\|\boldsymbol{v}_1\|_2 \min(\|\boldsymbol{v}_2\|_2, \|\boldsymbol{v}_3\|_2)}{\lambda_{\Sigma,\mathcal{U}}}\Big)\Big),$$

and otherwise, we have that

$$\left|\left|f_{\boldsymbol{\mu}_0,\boldsymbol{\mu}_1} - f_{\boldsymbol{\mu}'_0,\boldsymbol{\mu}'_1}\right|\right|_{\mathsf{TV}} = \Omega\Big(\min\Big(1, \frac{\min(\|\boldsymbol{v}_2\|_2, \|\boldsymbol{v}_3\|_2)}{\sqrt{\lambda_{\Sigma,\mathcal{U}}}}\Big)\Big).$$

Notice from the definitions of $\boldsymbol{v}_1, \boldsymbol{v}_2, \boldsymbol{v}_3$ that $\mathcal{U}$ is contained within the subspace spanned by the unknown mean vectors $\boldsymbol{\mu}_0, \boldsymbol{\mu}_1, \boldsymbol{\mu}'_0, \boldsymbol{\mu}'_1$. Furthermore, $\lambda_{\Sigma,\mathcal{U}}$ as defined in Theorem 2.8 can always be bounded from above by the largest eigenvalue of the matrix $\boldsymbol{\Sigma}$, and as we will show, this upper bound characterizes the total variation distance between mixtures in several instances.

In the special case of one component Gaussians, i.e., $\boldsymbol{\mu}_0 = \boldsymbol{\mu}_1$ and $\boldsymbol{\mu}'_0 = \boldsymbol{\mu}'_1$, we recover a result by Devroye et al. (see the lower bound in [52, Theorem 1.2], setting $\Sigma_1 = \Sigma_2$). In the one-dimensional setting, our next theorem shows a novel

lower bound on the total variation distance between any two distinct two-component one-dimensional Gaussian mixtures from $\mathcal{F}$.

Theorem 2.9. *Without loss of generality, for $f_{\mu_0,\mu_1}, f_{\mu_0',\mu_1'} \in \mathcal{F}$, suppose $\mu_0 \leq \min(\mu_1, \mu_0', \mu_1')$ and $\mu_0' \leq \mu_1'$. Further, let $\delta_1 = \max\{|\mu_0 - \mu_1|, |\mu_0' - \mu_1'|\}$ and $\delta_2 = \max\{|\mu_0' - \mu_0|, |\mu_1 - \mu_1'|\}$. If $[\mu_0', \mu_1'] \subseteq [\mu_0, \mu_1]$ and $\sigma = \Omega(\delta_1)$, then we have that*

$$||f_{\mu_0,\mu_1} - f_{\mu_0',\mu_1'}||_{\mathsf{TV}} \geq \Omega(\min(1, \delta_1\delta_2/\sigma^2))$$

and otherwise, $||f_{\mu_0,\mu_1} - f_{\mu_0',\mu_1'}||_{\mathsf{TV}} \geq \Omega(\min(1, \delta_2/\sigma))$.

Subsequently, we show the sufficient number of samples to estimate the parameters of an unknown two component one-dimensional mixture f_{μ_0,μ_1} up to a specified precision with high probability. We use our results on lower bounds of total variation distance between mixtures in conjunction with well known theoretical tools such as the Minimum Distance estimator and Scheffe Estimator (we refer the reader to Chapter D for detailed introduction on these tools). First, we characterize the statistical properties of the Minimum Distance Estimator in the following theorem:

Theorem 2.10. *Fix $0 < \delta'$ and $0 < \eta < C$, for C a small absolute constant. For $f_{\mu_0,\mu_1} \in \mathcal{F}$ with separation δ, if $\delta \geq \delta'$, then we can (δ', η)-learn f_{μ_0,μ_1} from the minimum distance estimator over $\mathcal{F}$ using $s(\delta', \eta) = O(\max(1, \sigma^4/(\delta^2\delta'^2)) \log^2 \eta^{-1})$ samples. If $\delta' \geq \delta$, then we only need $s(\delta', \eta) = O(\max(1, \sigma^2/\delta'^2) \log^2 \eta^{-1})$ samples.*

Next, we demonstrate a polynomial time algorithm that uses the Scheffe estimator to achieve the same sample complexity guarantee upto log factors.

Theorem 2.11. *Let $\delta' > 0$ and $0 < \eta < C$ be fixed, for C a small absolute constant. For $f_{\mu_0,\mu_1} \in \mathcal{F}$ with separation δ, if $\delta \geq \delta'$, then we can (δ', η)-learn f_{μ_0,μ_1} using*

$$s(\delta', \eta) = O(\max(1, \sigma^4 \log(\sigma/\delta')/(\delta^2\delta'^2)) \log^3 \eta^{-1})$$

samples with an algorithm that uses the Scheffe estimator and runs in time

$$O\Big(\frac{\sigma^6}{\delta'^2\delta^4}\log\frac{\sigma}{\delta'}\log^3\eta^{-1}\Big).$$

If $\delta' \geq \delta$, then we only need $s(\delta',\eta) = O(\max(1,\sigma^2/\delta'^2)\log(\sigma/\delta)\log^3\eta^{-1})$ samples.

Prior Work: Let $\boldsymbol{I}$ denote the d-dimensional identity matrix. Statistical distances between a pair of k-component d-dimensional Gaussian mixtures $f = \sum_{i=1}^{k} k^{-1}\mathcal{N}(\boldsymbol{\mu}_i, \boldsymbol{I})$ and $f' = \sum_{i=1}^{k} k^{-1}\mathcal{N}(\boldsymbol{\mu}'_i, \boldsymbol{I})$ with shared and known component covariance $\boldsymbol{I}$ have been studied in [58, 147]. For a k-component Gaussian mixture $f = \sum_{i=1}^{k} k^{-1}\mathcal{N}(\boldsymbol{\mu}_i, \boldsymbol{I})$, let $M_\ell(f) = \sum_{i=1}^{\ell} k^{-1}\boldsymbol{\mu}_i^{\otimes\ell}$ where $\boldsymbol{x}^{\otimes\ell}$ is the ℓ-wise tensor product of $\boldsymbol{x}$. We denote the Kullback-Leibler divergence, Squared Hellinger divergence and χ^2-divergence of f, f' by $||f-f'||_{\mathsf{KL}}, ||f-f'||_{\mathsf{H}^2}$ and $||f-f'||_{\chi^2}$ respectively. We will write $||M||_F$ to denote the Frobenius norm of the matrix M. Prior work shows the following bounds.

Theorem 2.12 (Theorem 4.2 in [58])**.** *Consider a pair of k-component mixtures $f = \sum_{i=1}^{k} k^{-1}\mathcal{N}(\boldsymbol{\mu}_i, \boldsymbol{I})$ and $f' = \sum_{i=1}^{k} k^{-1}\mathcal{N}(\boldsymbol{\mu}'_i, \boldsymbol{I})$ where $||\boldsymbol{\mu}_i||_2 \leq R, ||\boldsymbol{\mu}'_i||_2 \leq R$, for all $i \in [k]$ and constant $R \geq 0$. For any distance $D \in \{\mathsf{H}^2, \mathsf{KL}, \chi^2\}$, we have $\|f-f'\|_D = \Theta\Big(\max_{\ell\leq 2k-1}\|M_\ell(f) - M_\ell(f')\|_F^2\Big)$.*

This bound is does not give a guarantee for the TV distance. However it is well-known that,

$$\big|\big|f_{\boldsymbol{\mu}_0,\boldsymbol{\mu}_1} - f_{\boldsymbol{\mu}'_0,\boldsymbol{\mu}'_1}\big|\big|_{\mathsf{TV}} \geq \big|\big|f_{\boldsymbol{\mu}_0,\boldsymbol{\mu}_1} - f_{\boldsymbol{\mu}'_0,\boldsymbol{\mu}'_1}\big|\big|_{\mathsf{H}^2}. \tag{2.1}$$

We can use this in conjunction with Theorem 2.12 to get a lower bound on TV distance. That bound turns out to be suboptimal for many canonical instances. As a simple example, consider a pair of one-dimensional Gaussian mixtures

$$f = 0.5\mathcal{N}(u,1) + 0.5\mathcal{N}(-u,1) \quad \text{and} \quad f' = 0.5\mathcal{N}(2u,1) + 0.5\mathcal{N}(-2u,1). \tag{2.2}$$

Using Eq. (2.1) and Theorem 2.12, we have that $||f - f'||_{\mathsf{TV}} = \Omega(u^4)$. On the other hand, by using our result (Theorem 2.9), we obtain the improved bound $||f - f'||_{\mathsf{TV}} = \Omega(u^2)$. The improvement becomes more significant as u becomes smaller. Also, the prior result in Theorem 2.12 assumes that the means of the two mixtures f, f' are contained in a ball of constant radius, limiting its applicability.

The TV distance between Gaussian mixtures with two components for the special case of $d = 1$ has been recently studied in the context of parameter estimation [64, 77, 108, 76]. The TV distance guarantees in these papers are more general in the sense that they do not need the component covariances to be same. However, the results (and proofs) are tailored towards the case when both the mixtures and have zero mean. Hence, the results do not apply when we are considering the TV distance between two mixtures with distinct means (since it is no longer possible to subtract a single number to make both mixtures zero mean). In comparison, our results (Theorems 2.8 and 2.9) hold for all mixtures with shared component variances, and we not need any assumptions on the means.

We also note that our bound can be tighter than these prior results, even in the special case when the mixtures have zero mean. Consider again the pair of mixtures f, f' defined in Eq. (2.2) above. In [108, 77], the authors show that $||f - f'||_{\mathsf{TV}} = \Omega(u^4)$; see, e.g., Eq. (2.7) in [108]. Notice that this is the same bound that can be recovered from Theorem 2.12, and as we mentioned before, this bound is loose. Indeed, by using Theorem 2.9, we obtain the improved bound $||f - f'||_{\mathsf{TV}} = \Omega(u^2)$. Now consider a more general pair of mixtures, where for $u, v \geq 0$, we define

$$f = 0.5\mathcal{N}(u, 1) + 0.5\mathcal{N}(-u, 1) \text{ and } f' = 0.5\mathcal{N}(v, 1) + 0.5\mathcal{N}(-v, 1). \tag{2.3}$$

In [64] (see the proof of Lemma G.1 part (b)), the authors have shown that $||f - f'||_{\mathsf{TV}} = \Omega((u - v)^2)$. Notice that for the previous example in Eq. (2.2) with $v = 2u$, the result in [64] leads to the bound $||f - f'||_{\mathsf{TV}} = \Omega(u^2)$, which is the same bound that can be

obtained from Theorem 2.9. However, for any small $\epsilon > 0$, by setting $v = u + \epsilon$, we see that the bound in [64] reduces to $||f - f'||_{\mathsf{TV}} = \Omega(\epsilon^2)$. On the other hand, by using Theorem 2.9, we obtain the bound $||f - f'||_{\mathsf{TV}} = \Omega(u \cdot \epsilon)$. In particular, whenever $u \gg \epsilon$, our result provides a much larger and tighter lower bound.

Tightness of the TV distance bound: Our bounds on the TV distance are tight up to constant factors. For example, let $\boldsymbol{u} \in \mathbb{R}^d$ be a d-dimensional vector satisfying $||\boldsymbol{u}||_2 < 1$. Consider the mixtures

$$f = 0.5\mathcal{N}(\boldsymbol{u}, \boldsymbol{I}) + 0.5\mathcal{N}(-\boldsymbol{u}, \boldsymbol{I}) \quad \text{and} \quad f' = 0.5\mathcal{N}(2\boldsymbol{u}, \boldsymbol{I}) + 0.5\mathcal{N}(-2\boldsymbol{u}, \boldsymbol{I})$$

where $\boldsymbol{I}$ is the d-dimensional identity matrix. Considering the notation of Theorem 2.8, we have $\boldsymbol{v}_1 = 2\boldsymbol{u}$ and $\boldsymbol{v}_2 = \boldsymbol{u}$, and the first bound in the theorem implies that $||f - f'||_{\mathsf{TV}} \geq \Omega(||\boldsymbol{u}||_2^2)$. On the other hand, we use the inequality $||f - f'||_{\mathsf{TV}} \leq \sqrt{2\,||f - f'||_{\mathsf{H}^2}}$ in conjunction with Theorem 2.12. In the notation of Theorem 2.12, note that $M_1(f) - M_1(f') = 0$, and we can upper bound the max over $\ell \in \{2, 3\}$ by the sum of the two terms to say that

$$\begin{aligned} ||f - f'||_{\mathsf{TV}} &\leq O\Big(\max_{\ell \in \{2,3\}} \|M_\ell(f) - M_\ell(f')\|_F^2\Big) \\ &\leq O(||\boldsymbol{v} \otimes \boldsymbol{u}||_F + ||\boldsymbol{u} \otimes \boldsymbol{u} \otimes \boldsymbol{u}||_F) = O(||\boldsymbol{u}||_2^2 + ||\boldsymbol{u}||_2^3). \end{aligned}$$

Since $||\boldsymbol{u}||_2 < 1$, we see that $||\boldsymbol{u}||_2^2$ is the dominating term on the RHS, and $||f - f'||_{\mathsf{TV}} = \Theta(||\boldsymbol{u}||_2^2)$. As a result, our TV distance bound in Theorem 2.8 is tight as a function of the means for this example.

2.3.1 Overview of Proofs

One dimension: In one dimension, we lower bound the TV distance as follows. For $f_{\mu_0,\mu_1}, f_{\mu_0',\mu_1'} \in \mathcal{F}$, suppose μ_0 is the smallest mean. Recall that $\delta_1 = \max\{|\mu_0 -$

$\mu_1|, |\mu_0' - \mu_1'|\}$ and $\delta_2 = \max\{|\mu_0' - \mu_0|, |\mu_1 - \mu_1'|\}$. If $[\mu_0', \mu_1'] \subseteq [\mu_0, \mu_1]$, then

$$||f_{\mu_0,\mu_1} - f_{\mu_0',\mu_1'}||_{\mathsf{TV}} \geq \Omega(\min(1, \delta_1\delta_2/\sigma^2))$$

and otherwise,

$$||f_{\mu_0,\mu_1} - f_{\mu_0',\mu_1'}||_{\mathsf{TV}} \geq \Omega(\min(1, \delta_2/\sigma)).$$

The latter case corresponds to when either both means from one mixture are smaller than another, i.e., $\mu_0 \leq \mu_1 \leq \mu_0', \mu_1'$, or the mixtures' means are interlaced, i.e., $\mu_0 \leq \mu_0' \leq \mu_1 \leq \mu_1'$.

We use Lemma 2.1 to lower bound the TV distance between mixtures $f_{\mu_0,\mu_1}, f_{\mu_0',\mu_1'} \in \mathcal{F}$ by the modulus of a complex analytic function:

$$4\left|\left|f_{\mu_0,\mu_1} - f_{\mu_0',\mu_1'}\right|\right|_{\mathsf{TV}} \geq \sup_t e^{-\frac{\sigma^2t^2}{2}}\left|e^{it\mu_0} + e^{it\mu_1} - e^{it\mu_0'} - e^{it\mu_1'}\right|. \tag{2.4}$$

Let $h(t) = e^{it\mu_0} + e^{it\mu_1} - e^{it\mu_0'} - e^{it\mu_1'}$. A lower bound on $||f_{\mu_0,\mu_1} - f_{\mu_0',\mu_1'}||_{\mathsf{TV}}$ can be obtained by taking $t = 1/(c\sigma)$ for c constant, so that $e^{-\sigma^2t^2/2}$ is not too small. Then, it remains to bound $|h(t)|$ at the chosen value of t. In some cases, we will have to choose the constant c very carefully, as terms in $h(t)$ can cancel out due to the periodicity of the complex exponential function. For instance, if $\mu_0 = 0$, $\mu_1 = 200\sigma$, $\mu_0' = \sigma$, and $\mu_1' = 201\sigma$ with $\sigma = 2\pi$, then $|h(1)| = 0$.

It is reasonable to wonder whether there is a simple, global way to lower bound Eq. (2.4). We could reparameterize the function $h(t)$ as the complex function $g(z) = z^{\mu_0} + z^{\mu_1} - z^{\mu_0'} - z^{\mu_1'}$, where $z = e^{it}$, then study $|g(z)|$, for z in the disc with center 0 and radius 1 in the complex plane. However, we are unaware of a global way to bound $|g(z)|$ here due to the fact that (i) $g(z)$ is not analytic at 0 when the means are non-integral and (ii) there is not a clear, large lower bound for $g(z)$ anywhere inside the unit disc. These two facts obstruct the use of either the Maximum Modulus

Principle or tools from harmonic measure to obtain lower bounds. Instead, we use a series of lemmas to handle the different ways that $|h(t)|$ can behave. The techniques include basic complex analysis and Taylor series approximations of order at most three.

High-Dimensional Overview: Let $f^{\boldsymbol{t}}_{\boldsymbol{\mu}_0,\boldsymbol{\mu}_1}$ be the distribution of the samples obtained according to $f_{\boldsymbol{\mu}_0,\boldsymbol{\mu}_1}$ and projected onto the direction $\boldsymbol{t} \in \mathbb{R}^d$. We have (see Lemma 2.12 for a proof)

$$f^{\boldsymbol{t}}_{\boldsymbol{\mu}_0,\boldsymbol{\mu}_1} \equiv \frac{\mathcal{N}(\boldsymbol{\mu}_0^T\boldsymbol{t}, \boldsymbol{t}^T\boldsymbol{\Sigma}\boldsymbol{t})}{2} + \frac{\mathcal{N}(\boldsymbol{\mu}_1^T\boldsymbol{t}, \boldsymbol{t}^T\boldsymbol{\Sigma}\boldsymbol{t})}{2}$$
$$f^{\boldsymbol{t}}_{\boldsymbol{\mu}_0',\boldsymbol{\mu}_1'} \equiv \frac{\mathcal{N}(\boldsymbol{\mu}_0'^T\boldsymbol{t}, \boldsymbol{t}^T\boldsymbol{\Sigma}\boldsymbol{t})}{2} + \frac{\mathcal{N}(\boldsymbol{\mu}_1'^T\boldsymbol{t}, \boldsymbol{t}^T\boldsymbol{\Sigma}\boldsymbol{t})}{2}.$$

By the data processing inequality for f-divergences (see Theorem 5.2 in [51]), we have

$$\|f_{\boldsymbol{\mu}_0,\boldsymbol{\mu}_1} - f_{\boldsymbol{\mu}_0',\boldsymbol{\mu}_1'}\|_{\mathsf{TV}} \geq \sup_{\boldsymbol{t}\in\mathbb{R}^d} \|f^{\boldsymbol{t}}_{\boldsymbol{\mu}_0,\boldsymbol{\mu}_1} - f^{\boldsymbol{t}}_{\boldsymbol{\mu}_0',\boldsymbol{\mu}_1'}\|_{\mathsf{TV}}.$$

Using our lower bound on the TV distance between one-dimensional mixtures (Theorem 2.9), we obtain a lower bound on $\|f_{\boldsymbol{\mu}_0,\boldsymbol{\mu}_1} - f_{\boldsymbol{\mu}_0',\boldsymbol{\mu}_1'}\|_{\mathsf{TV}}$ by choosing $\boldsymbol{t} \in \mathbb{R}^d$ carefully. This leads to Theorem 2.8.

Parameter estimation in one dimension: Recall that our objective is to learn f_{μ_0,μ_1} with unknown means $\mu_0, \mu_1 \in \mathbb{R}$ using samples from f_{μ_0,μ_1}. As we saw in Theorem 2.10, the minimum distance estimator recovers μ_0, μ_1 upto a precision of δ' using $\tilde{O}(d\sigma^2/\delta^2\delta'^2)$ samples with high probability probability where $\delta' = |\mu_0 - \mu_1|$. The minimum distance estimator chooses the best distribution from $\mathcal{F} \equiv \{f_{\theta_0,\theta_1} := 0.5\mathcal{N}(\theta_0, \sigma^2) + 0.5\mathcal{N}(\theta_1, \sigma^2), \theta_0, \theta_1 \in \mathbb{R}\}$ by evaluating every candidate distribution. As $|\mathcal{F}|$ is infinite, the running time of the minimum distance estimator is infinite as well. Our approach to resolve this issue is to design a finite cover of $\mathcal{F}$ i.e. design a

finite subset of candidates $\mathcal{F}' \subset \mathcal{F}$ and subsequently, use the Scheffe estimator. The running time of the Scheffe estimator is $O(n|\mathcal{F}'|^2)$ which is small if $|\mathcal{F}'|$ is small.

In order to design a finite cover, our first step is to use a statistical test powered by the Method of Moments with $\tilde{O}(\max(1, \sigma^2))$ samples from f_{μ_0,μ_1} to test if the gap $|\mu_0 - \mu_1|$ is larger than $3\sigma/4 + o(\sigma)$ with high probability. If the gap is indeed larger than $3\sigma/4$, then we can simply run EM Algorithm and using the results in [41], we will obtain a (δ', η)-estimate of f_{μ_0,μ_1} with sample/time complexity $\tilde{O}\Big(\frac{\sigma^2}{\delta'^2} \log \frac{\sigma}{\delta'}\Big)$. On the other hand, if the gap is $|\mu_0 - \mu_1|$ is smaller than $3\sigma/4 + o(\sigma)$, then we first compute an estimate $\hat{M}$ of the mean $(\mu_0 + \mu_1)/2$ of the mixture f_{μ_0,μ_1} upto a precision of ϵ. Now, we can simply substract $\hat{M}$ from every sample and instead learn $f_{\mu_0-\hat{M},\mu_1-\hat{M}}$. In order to do so, we design the following set of candidate distributions

$$\mathcal{F}' \equiv \left\{ \frac{\mathcal{N}(a\rho, \sigma^2)}{2} + \frac{\mathcal{N}(-a\rho, \sigma^2)}{2}; -\frac{5\sigma}{6\rho} \leq a \leq \frac{5\sigma}{6\rho}, a \in \mathbb{Z} \right\}.$$

for the Scheffe estimator. Clearly $|\mathcal{F}| \leq |5\sigma/3\rho|$ is finite. Notice from equation D.2 that the Scheffe estimator has similar guarantees to the Minimum distance estimator.

The main roadblock in following this approach is the fact that f_{μ_0,μ_1} does not belong to $\mathcal{F}'$ unlike the other learning settings in this work where the unknown distribution also belongs to the set of candidates. Therefore, in this case $\inf_{f \in \mathcal{F}'} ||f_{\mu_0,\mu_1} - f||_{\mathsf{TV}} \geq 0$ and in order to get a good theoretical guarantee for the Scheffe estimator we must bound the quantity $\inf_{f \in \mathcal{F}'} ||f_{\mu_0,\mu_1} - f||_{\mathsf{TV}}$ from above. In order to do so, we can use Moment matching techniques to show that there exists a candidate distribution in $\mathcal{F}'$ (with an appropriate choice of ρ) whose TV distance with f_{μ_0,μ_1} must be $o(\delta\delta'/\sigma^2)$ and further the parameters of this candidate are very close to the unknown parameters of f_{μ_0,μ_1}. On the other hand, we show that for any candidate distribution $f_{\mu_0',\mu_1'}$ in $\mathcal{F}'$ such that

$$\min_{\pi:\{0,1\}\to\{0,1\}} \max\{\left|\mu_0' - \mu_{\pi(0)}\right|, \left|\mu_1' - \mu_{\pi(1)}\right|\} \geq \delta',$$

we must have $\left|\left|f_{\mu_0',\mu_1'} - f_{\mu_0,\mu_1}\right|\right|_{\mathsf{TV}} \geq \Omega(\delta\delta'/\sigma^2)$. Putting everything together with an appropriate choice of ϵ, ρ, we get that the Scheffe estimator returns a (δ', η)-estimate of f_{μ_0,μ_1} and with a polynomial running time (see Theorem 2.10).

2.3.2 Lower Bounds on TV Distance of 1-Dimensional Mixtures

Consider distinct Gaussian mixtures $f_{\mu_0,\mu_1}, f_{\mu_0',\mu_1'} \in \mathcal{F}$. Without loss of generality we will also let $\mu_0 \leq \min(\mu_1, \mu_0', \mu_1')$ be the smallest unknown parameter, and let $\mu_1' \geq \mu_0'$. We maintain these assumptions throughout this section, and we will prove Theorem 2.9.

Eq. (2.4) implies that we can lower bound $||f_{\mu_0,\mu_1} - f_{\mu_0',\mu_1'}||_{\mathsf{TV}}$ by the modulus of a complex analytic function with parameter t. Then, we can optimize the bound by choosing $t = \Theta(1/\sigma)$ and lower bounding the term in the absolute value signs.

We define the following parameters relative to the means to simplify some bounds:

$$\delta_1 = \max(|\mu_0 - \mu_1|, |\mu_0' - \mu_1'|) \qquad \delta_2 = \max(|\mu_0' - \mu_0|, |\mu_1 - \mu_1'|)$$
$$\delta_3 = |\mu_0 + \mu_1 - \mu_0' - \mu_1'| \qquad \delta_4 = \min(|\mu_0' - \mu_0|, |\mu_1' - \mu_1|).$$

We first consider t such that $t(\mu_1 - \mu_0), t(\mu_1' - \mu_0), t(\mu_1' - \mu_0) \leq \frac{\pi}{4}$, which is covered in Lemma 2.9.

Lemma 2.9. *For $t > 0$ with $t(\mu_1 - \mu_0), t(\mu_1' - \mu_0), t(\mu_1' - \mu_0) \in [0, \frac{\pi}{4}]$, if $\mu_0', \mu_1' \in [\mu_0, \mu_1]$, then*

$$\left|e^{it\mu_0} + e^{it\mu_1} - e^{it\mu_0'} - e^{it\mu_1'}\right| \geq \max\left(\frac{t^2(\delta_1 - \delta_4)\delta_4}{2}, \frac{t\delta_3}{4\sqrt{2}}\right)$$

and otherwise, when $\mu_1' > \mu_1$, $\left|e^{it\mu_0} + e^{it\mu_1} - e^{it\mu_0'} - e^{it\mu_1'}\right| \geq t\delta_2/(2\sqrt{2})$.

See Figure 2.1 for an illustration of the different ways that the means can be ordered. The lemma follows from straightforward calculations that only use Taylor series approximations, trigonometric identities, and basic facts about complex numbers. We include the proof in Chapter A, Section A.2.

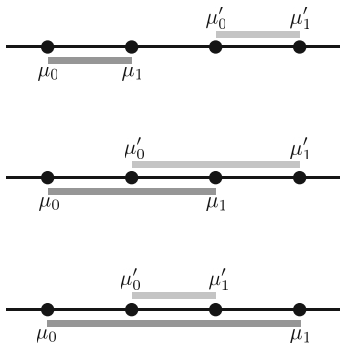


Figure 2.1: Layout of the means for Theorem 2.9. The means can be ordered in different ways, which affects the analysis of lower bounding $|e^{it\mu_0} + e^{it\mu_1} - e^{it\mu_0'} - e^{it\mu_1'}|$ in Lemma 2.9. For a fixed t, the order affects (i) whether the real or imaginary part of $e^{it\mu_0} + e^{it\mu_1} - e^{it\mu_0'} - e^{it\mu_1'}$ has large modulus and (ii) whether the terms from μ_0 and μ_1 or μ_0' and μ_1' dominate.

Recall that we will choose $t = \Theta(1/\sigma)$ to cancel the exponential term in Eq. (2.4). Therefore, Lemma 2.9 handles the case when all the means are within some interval of size $\Theta(\sigma)$.

Next, we prove that when the separation between the mixtures is substantially fair apart—when either $|\mu_0 - \mu_0'|$ or $|\mu_1 - \mu_1'|$ is at least 2σ—we have a constant lower bound on the TV distance. Recall that it is without loss of generality to assume that μ_0 is the smallest parameter and $\mu_1' > \mu_0'$. A similar result as the following two lemmas has been observed previously (e.g., [72]) but we provide a simple and self-contained proof.

Lemma 2.10. *If* $\max(|\mu_0 - \mu_0'|, |\mu_1 - \mu_1'|) \geq 2\sigma$, *then it follows that* $||f_{\mu_0,\mu_1} - f_{\mu_0',\mu_1'}||_{\mathsf{TV}} \geq \Omega(1)$.

Proof. Assume that $|\mu_0 - \mu_0'| \geq 2\sigma$, where the case $|\mu_1 - \mu_1'| \geq 2\sigma$ is analogous. Recall from the definition of TV distance that

$$||f_{\mu_0,\mu_1} - f_{\mu_0',\mu_1'}||_{\mathsf{TV}} \triangleq \sup_{\mathcal{A}\subseteq\Omega} \left(f_{\mu_0,\mu_1}(\mathcal{A}) - f'_{\mu_0',\mu_1'}(\mathcal{A}) \right)$$

$$\geq \Pr_{X\sim f_{\mu_0,\mu_1}}[X \leq \mu_0 + \sigma] - \Pr_{Y\sim f_{\mu_0',\mu_1'}}[Y \leq \mu_0 + \sigma].$$

For a random variable $X \sim f_{\mu_0,\mu_1}$, let $\mathcal{E}$ denote the event that we choose the component with mean μ_0, i.e., if $\widetilde{X}$ denotes X conditioned on $\mathcal{E}$, then we have $\widetilde{X} \sim \mathcal{N}(\mu_0, \sigma^2)$. Since the mixing weights are equal, we have $\Pr(\mathcal{E}) = \Pr(\mathcal{E}^c) = 1/2$, where $\mathcal{E}^c$ is the complement of $\mathcal{E}$. Therefore,

$$\begin{aligned}\Pr(X \geq \mu_0 + \sigma) &\leq \Pr(\mathcal{E}) \Pr(X \geq \mu_0 + \sigma \mid \mathcal{E}) + \Pr(\mathcal{E}^c) = \frac{1}{2} \int_{\mu_0+\sigma}^{\infty} \frac{e^{-\frac{(t-\mu_0)^2}{2\sigma^2}}}{\sqrt{2\pi}\sigma} dt + \frac{1}{2} \\ &\leq \frac{1}{2} \int_{\mu_0+\sigma}^{\infty} \Big(\frac{t-\mu_0}{\sigma}\Big) \frac{e^{-\frac{(t-\mu_0)^2}{2\sigma^2}}}{\sqrt{2\pi}\sigma} dt + \frac{1}{2} \leq \frac{1}{2} \cdot \frac{e^{-\frac{1}{2}}}{\sqrt{2\pi}} + \frac{1}{2}. \end{aligned} \tag{2.5}$$

Recall that $\mu_0 \leq \mu_0'$, $\mu_0' \leq \mu_1'$ and $|\mu_0 - \mu_0'| \geq 2\sigma$. Again, for a random variable $Y \sim f_{\mu_0',\mu_1'}$, let $\mathcal{E}'$ denote the event that the component with mean μ_0' is chosen (and $\mathcal{E}'^c$ denotes μ_1' is chosen). Then,

$$\begin{aligned}\Pr(Y \leq \mu_0 + \sigma) &= \Pr(\mathcal{E}') \Pr(Y \leq \mu_0 + \sigma \mid \mathcal{E}') + \Pr(\mathcal{E}'^c) \Pr(Y \leq \mu_0 + \sigma \mid \mathcal{E}'^c) \\ &\overset{a}{=} \Pr(\mathcal{E}') \Pr(Y \leq \mu_0' - \sigma \mid \mathcal{E}') + \Pr(\mathcal{E}'^c) \Pr(Y \leq \mu_1' - \sigma \mid \mathcal{E}'^c) \\ &\overset{b}{=} \Pr(\mathcal{E}') \Pr(Y \geq \mu_0' + \sigma \mid \mathcal{E}') + \Pr(\mathcal{E}'^c) \Pr(Y \geq \mu_1' + \sigma \mid \mathcal{E}'^c) \\ &\overset{c}{\leq} \frac{1}{2} \cdot \frac{e^{-\frac{1}{2}}}{\sqrt{2\pi}} + \frac{1}{2} \cdot \frac{e^{-\frac{1}{2}}}{\sqrt{2\pi}} = \frac{e^{-\frac{1}{2}}}{\sqrt{2\pi}}\end{aligned}$$

where in step (a), we used the fact that $\mu_0' - \sigma \geq \mu_0 + \sigma$ and $\mu_1' - \sigma \geq \mu_0 + \sigma$; in step (b), we used the symmetry of Gaussian distributions; in step (c), we used the same analysis as in (2.5). By plugging this in the definition of TV distance, we have

$$\begin{aligned}||f_{\mu_0,\mu_1} - f_{\mu_0',\mu_1'}||_{\mathsf{TV}} &\geq \Pr_{X \sim f_{\mu_0,\mu_1}}[X \leq \mu_0 + \sigma] - \Pr_{Y \sim f_{\mu_0',\mu_1'}}[Y \leq \mu_0 + \sigma] \\ &\geq \frac{1}{2} - \sqrt{\frac{9}{8\pi e}} \geq 0.137.\end{aligned}$$

□

If Lemma 2.10 does not apply, then we case on whether $\max(|\mu_0 - \mu_1|, |\mu_0' - \mu_1'|)$ is large or not. If $\max(|\mu_0 - \mu_1|, |\mu_0' - \mu_1'|) < 100\sigma$, we use Lemma 2.9—exactly how will be explained later—and otherwise we use the following lemma. Recall that $\delta_2 = \max(|\mu_0' - \mu_0|, |\mu_1 - \mu_1'|)$.

Lemma 2.11. *If* $\max(|\mu_0 - \mu_1|, |\mu_0' - \mu_1'|) \geq 100\sigma$ *and* $\max(|\mu_0 - \mu_0'|, |\mu_1 - \mu_1'|) \leq 2\sigma$*, then*

$$\sup_t e^{-\frac{\sigma^2 t^2}{2}} \left| e^{it\mu_0} + e^{it\mu_1} - e^{it\mu_0'} - e^{it\mu_1'} \right| \geq \frac{\pi^2 \delta_2}{240 e \sigma}.$$

We defer the proof of Lemma 2.11 to Chapter A, Section A.2. Using Lemmas 2.9, 2.10, and 2.11, we prove Theorem 2.9.

Proof of Theorem 2.9. Using Lemma 2.1, we see that

$$2 \left|\left| f_{\mu_0,\mu_1} - f_{\mu_0',\mu_1'} \right|\right|_{\mathsf{TV}} \geq \sup_t \frac{e^{-\frac{\sigma^2 t^2}{2}}}{2} \left| e^{it\mu_0} + e^{it\mu_1} - e^{it\mu_0'} - e^{it\mu_1'} \right|.$$

Case 1: Consider the case when $\mu_0, \mu_1, \mu_0', \mu_1'$ are in an interval of size at most 100σ, i.e.,

$$\max\Big(|\mu_1' - \mu_0|, |\mu_1 - \mu_0|, |\mu_0' - \mu_0| \Big) \leq 100\sigma. \tag{2.6}$$

Recall that $\delta_1 = \max\{|\mu_0 - \mu_1|, |\mu_0' - \mu_1'|\}$, $\delta_2 = \max(|\mu_0' - \mu_0|, |\mu_1 - \mu_1'|)$, $\delta_3 = |\mu_0 + \mu_1 - \mu_0' - \mu_1'|$, $\delta_4 = \min(|\mu_0' - \mu_0|, |\mu_1' - \mu_1|)$. For $t = \pi/400\sigma$, $0 \leq t \max\Big(|\mu_1' - \mu_0|, |\mu_1 - \mu_0|, |\mu_0' - \mu_0| \Big) \leq \frac{\pi}{4}$. We have assumed that $\mu_0 \leq \min(\mu_1, \mu_0', \mu_1')$ and $\mu_0' \leq \mu_1'$. This implies that $\mu_0 \leq \mu_0' \leq \mu_1' \leq \mu_1$ in the subcase when $\mu_0', \mu_1' \in [\mu_0, \mu_1]$. This also implies that $\delta_1 = |\mu_1 - \mu_0| \geq 2\delta_4$, a fact we will use later. The inequality in Eq. (2.6) implies that the above value of $t = \pi/400\sigma$ satisfies

the conditions of Lemma 2.9. Then, when $\mu_0', \mu_1' \in [\mu_0, \mu_1]$, the first part of the lemma implies that

$$2\left|\left|f_{\mu_0,\mu_1} - f_{\mu_0',\mu_1'}\right|\right|_{\mathsf{TV}} \geq \max\left(\frac{\pi^2(\delta_1 - \delta_4)\delta_4}{640000e\sigma^2}, \frac{\pi\delta_3}{3200\sqrt{2}e\sigma}\right)$$

Now we observe that $\delta_3 \geq \delta_2 - \delta_4$. To see this, assume without loss of generality that $\delta_2 = |\mu_0' - \mu_0|$ and $\delta_4 = |\mu_1' - \mu_1|$. By the triangle inequality, we have that $\delta_3 = |\mu_0 + \mu_1 - \mu_0' - \mu_1'| \geq |\mu_0' - \mu_0| - |\mu_1' - \mu_1| = \delta_2 - \delta_4$. We split up the calculations based on the value of δ_3. If $\delta_3 \geq \frac{\delta_2}{2}$, then $||f_{\mu_0,\mu_1} - f_{\mu_0',\mu_1'}||_{\mathsf{TV}} \geq \pi\delta_2/(12800\sqrt{2}e\sigma)$. On the other hand, if $\delta_3 \leq \frac{\delta_2}{2}$, then since $\delta_3 \geq \delta_2 - \delta_4$, we have that $\delta_4 \geq \frac{\delta_2}{2}$. Coupled with the fact that $\delta_1 \geq 2\delta_4$ (hence $\delta_4 \leq \delta_1/2$ implying $\delta_1 - \delta_4 \geq \delta_1/2$), we have that $||f_{\mu_0,\mu_1} - f_{\mu_0',\mu_1'}||_{\mathsf{TV}} \geq \pi^2\delta_1\delta_2/(5120000e\sigma^2)$. Putting these together, we have

$$\left|\left|f_{\mu_0,\mu_1} - f_{\mu_0',\mu_1'}\right|\right|_{\mathsf{TV}} \geq \min\left(\frac{\pi^2\delta_1\delta_2}{5120000e\sigma^2}, \frac{\pi\delta_2}{12800\sqrt{2}e\sigma}\right) = \frac{\pi^2\delta_1\delta_2}{5120000e\sigma^2}$$

For the case when both of μ_0', μ_1' are not in $[\mu_0, \mu_1]$, we have $\mu_1' > \mu_1$ (recall that μ_0 is the smallest mean and $\mu_0' \leq \mu_1'$), and we can use the second part of Lemma 2.9 to conclude that

$$2\left|\left|f_{\mu_0,\mu_1} - f_{\mu_0',\mu_1'}\right|\right|_{\mathsf{TV}} \geq \sup_t \frac{e^{-\frac{\sigma^2t^2}{2}}}{2}\left|e^{it\mu_0} + e^{it\mu_1} - e^{it\mu_0'} - e^{it\mu_1'}\right| \geq \frac{\pi\delta_2}{1600\sqrt{2}e\sigma}.$$

Case 2: Next, consider when $\delta_2 = \max(|\mu_0' - \mu_0|, |\mu_1 - \mu_1'|) \geq 2\sigma$. Lemma 2.10 implies that

$$\left|\left|f_{\mu_0,\mu_1} - f_{\mu_0',\mu_1'}\right|\right|_{\mathsf{TV}} \geq \Omega(1).$$

Case 3: Now, we consider the only remaining case, when

$$\delta_1 = \max(|\mu_0 - \mu_1|, |\mu_0' - \mu_1'|) \geq 100\sigma$$

and $\delta_2 \leq \max(|\mu_0 - \mu_0'|, |\mu_1 - \mu_1'|) \leq 2\sigma$. This case satisfies the conditions of Lemma 2.11, and therefore, we have that

$$2\left|\left|f_{\mu_0,\mu_1} - f_{\mu_0',\mu_1'}\right|\right|_{\mathsf{TV}} \geq \sup_t \frac{e^{-\frac{\sigma^2 t^2}{2}}}{2}\left|e^{it\mu_0} + e^{it\mu_1} - e^{it\mu_0'} - e^{it\mu_1'}\right| \geq \frac{\pi^2\delta_2}{240e\sigma},$$

thus proving the theorem. □

2.3.3 Lower Bounds on TV Distance of d-Dimensional Mixtures

We lower bound the TV distance of high-dimensional mixtures in $\mathcal{F}$ and prove Theorem 2.8. For any direction $\boldsymbol{t} \in \mathbb{R}^d$, we denote the projection of the distributions $f_{\boldsymbol{\mu}_0,\boldsymbol{\mu}_1}$ and $f_{\boldsymbol{\mu}_0',\boldsymbol{\mu}_1'}$ on $\boldsymbol{t}$ by $f^{\boldsymbol{t}}_{\boldsymbol{\mu}_0,\boldsymbol{\mu}_1}$ and $f^{\boldsymbol{t}}_{\boldsymbol{\mu}_0',\boldsymbol{\mu}_1'}$, respectively. The next lemma allows us to precisely define the projected mixtures.

Lemma 2.12. *For a random variable $\boldsymbol{x} \sim \frac{1}{2}\mathcal{N}(\boldsymbol{\mu}_0, \boldsymbol{\Sigma}) + \frac{1}{2}\mathcal{N}(\boldsymbol{\mu}_1, \boldsymbol{\Sigma})$, for any $\boldsymbol{t} \in \mathbb{R}^d$,*

$$\boldsymbol{t}^T\boldsymbol{x} \sim \frac{\mathcal{N}(\langle\boldsymbol{\mu}_0, \boldsymbol{t}\rangle, \boldsymbol{t}^T\boldsymbol{\Sigma}\boldsymbol{t})}{2} + \frac{\mathcal{N}(\langle\boldsymbol{\mu}_1, \boldsymbol{t}\rangle, \boldsymbol{t}^T\boldsymbol{\Sigma}\boldsymbol{t})}{2}.$$

Proof. A linear transformation of a multivariate Gaussian is also a Gaussian. For $\boldsymbol{x} \sim \mathcal{N}(\boldsymbol{\mu}_0, \boldsymbol{\Sigma})$, we see that $\langle\boldsymbol{t}, \boldsymbol{x}\rangle \sim \mathcal{N}(\langle\boldsymbol{\mu}_0, \boldsymbol{t}\rangle, \boldsymbol{t}^T\boldsymbol{\Sigma}\boldsymbol{t})$ by a computation of the mean and variance. Similarly, for $\boldsymbol{x} \sim \mathcal{N}(\boldsymbol{\mu}_1, \boldsymbol{\Sigma})$, we have $\langle\boldsymbol{t}, \boldsymbol{x}\rangle \sim \mathcal{N}(\langle\boldsymbol{\mu}_1, \boldsymbol{t}\rangle, \boldsymbol{t}^T\boldsymbol{\Sigma}\boldsymbol{t})$. Putting these together, the claim follows. □

From Lemma 2.12, we can exactly define the one-dimensional mixtures

$$f^{\boldsymbol{t}}_{\boldsymbol{\mu}_0,\boldsymbol{\mu}_1} = \frac{\mathcal{N}(\langle\boldsymbol{\mu}_0, \boldsymbol{t}\rangle, \boldsymbol{t}^T\boldsymbol{\Sigma}\boldsymbol{t})}{2} + \frac{\mathcal{N}(\langle\boldsymbol{\mu}_1, \boldsymbol{t}\rangle, \boldsymbol{t}^T\boldsymbol{\Sigma}\boldsymbol{t})}{2}$$

$$f^{\boldsymbol{t}}_{\boldsymbol{\mu}'_0,\boldsymbol{\mu}'_1} = \frac{\mathcal{N}(\langle \boldsymbol{\mu}'_0, \boldsymbol{t}\rangle, \boldsymbol{t}^T\boldsymbol{\Sigma}\boldsymbol{t})}{2} + \frac{\mathcal{N}(\langle \boldsymbol{\mu}'_1, \boldsymbol{t}\rangle, \boldsymbol{t}^T\boldsymbol{\Sigma}\boldsymbol{t})}{2}.$$

By using the data processing inequality, or the fact that variational distance is non-increasing under all mappings (see, for instance, Theorem 5.2 in [51]), it follows that

$$\left|\left|f_{\boldsymbol{\mu}_0,\boldsymbol{\mu}_1} - f_{\boldsymbol{\mu}'_0,\boldsymbol{\mu}'_1}\right|\right|_{\mathsf{TV}} \geq \sup_{\boldsymbol{t}\in\mathbb{R}^d} \left|\left|f^{\boldsymbol{t}}_{\boldsymbol{\mu}_0,\boldsymbol{\mu}_1} - f^{\boldsymbol{t}}_{\boldsymbol{\mu}'_0,\boldsymbol{\mu}'_1}\right|\right|_{\mathsf{TV}}.$$

Let $\mathcal{H}$ be the set of permutations on $\{0,1\}$. The following lemma has two cases based on whether the interval defined by one pair of mean's projections is contained in the interval defined by the other pair's projections.

Lemma 2.13. *Let $\boldsymbol{t} \in \mathbb{R}^d$ be any vector. If*

$$\sqrt{\boldsymbol{t}^T\boldsymbol{\Sigma}\boldsymbol{t}} = \Omega\Big(\max\left(|\langle \boldsymbol{t}, \boldsymbol{\mu}_0 - \boldsymbol{\mu}_1\rangle|, |\langle \boldsymbol{t}, \boldsymbol{\mu}'_0 - \boldsymbol{\mu}'_1\rangle|\right)\Big),$$

and either $\langle \boldsymbol{\mu}'_0, \boldsymbol{t}\rangle, \langle \boldsymbol{\mu}'_1, \boldsymbol{t}\rangle \in [\langle \boldsymbol{\mu}_0, \boldsymbol{t}\rangle, \langle \boldsymbol{\mu}_1, \boldsymbol{t}\rangle]$ or $\langle \boldsymbol{\mu}_0, \boldsymbol{t}\rangle, \langle \boldsymbol{\mu}_1, \boldsymbol{t}\rangle \in [\langle \boldsymbol{\mu}'_0, \boldsymbol{t}\rangle, \langle \boldsymbol{\mu}'_1, \boldsymbol{t}\rangle]$, then $||f^{\boldsymbol{t}}_{\boldsymbol{\mu}_0,\boldsymbol{\mu}_1} - f^{\boldsymbol{t}}_{\boldsymbol{\mu}'_0,\boldsymbol{\mu}'_1}||_{\mathsf{TV}}$ is at least

$$\Omega\Big(\min\Big(1, \frac{\max\left(|\langle \boldsymbol{t}, \boldsymbol{\mu}_0 - \boldsymbol{\mu}_1\rangle|, |\langle \boldsymbol{t}, \boldsymbol{\mu}'_0 - \boldsymbol{\mu}'_1\rangle|\right)}{\boldsymbol{t}^T\boldsymbol{\Sigma}\boldsymbol{t}} \times \min_{\sigma\in\mathcal{H}} \max\left(|\langle \boldsymbol{t}, (\boldsymbol{\mu}_0 - \boldsymbol{\mu}'_{\sigma(0)})\rangle|, |\langle \boldsymbol{t}, (\boldsymbol{\mu}_1 - \boldsymbol{\mu}'_{\sigma(1)})\rangle|\right)\Big)\Big).$$

Otherwise, we have that $||f^{\boldsymbol{t}}_{\boldsymbol{\mu}_0,\boldsymbol{\mu}_1} - f^{\boldsymbol{t}}_{\boldsymbol{\mu}'_0,\boldsymbol{\mu}'_1}||_{\mathsf{TV}}$ is at least

$$\Omega\Big(\min\Big(1, \min_{\sigma\in\mathcal{H}} \frac{\max(|\langle \boldsymbol{t}, (\boldsymbol{\mu}_0 - \boldsymbol{\mu}'_{\sigma(0)})\rangle|, |\boldsymbol{t}^T(\boldsymbol{\mu}_1 - \boldsymbol{\mu}'_{\sigma(1)})|)}{\sqrt{\boldsymbol{t}^T\boldsymbol{\Sigma}\boldsymbol{t}}}\Big)\Big).$$

Proof. The proof follows directly from Theorem 2.9. Note that in Theorem 2.9, we assumed the ordering of the means without loss of generality, i.e., $\mu_0 \leq \min(\mu_1, \mu'_0, \mu'_1)$

and $\mu'_0 < \mu'_1$. However, taking a minimum over the set of permutations in $\mathcal{H}$ allows us to restate the theorem in its full generality. □

Now we are ready to provide the proof of Theorem 2.8.

Proof of Theorem 2.8. Let

$$S_1 = \{\boldsymbol{\mu}_1 - \boldsymbol{\mu}_0, \boldsymbol{\mu}'_1 - \boldsymbol{\mu}'_0\}, \quad S_2 = \{\boldsymbol{\mu}'_0 - \boldsymbol{\mu}_0, \boldsymbol{\mu}'_1 - \boldsymbol{\mu}_1\}, \quad S_3 = \{\boldsymbol{\mu}'_0 - \boldsymbol{\mu}_1, \boldsymbol{\mu}'_1 - \boldsymbol{\mu}_0\},$$

and

$$\boldsymbol{v}_1 = \mathsf{argmax}_{s \in S_1} ||\boldsymbol{s}||_2, \quad \boldsymbol{v}_2 = \mathsf{argmax}_{s \in S_2} ||\boldsymbol{s}||_2, \quad \boldsymbol{v}_3 = \mathsf{argmax}_{s \in S_3} ||\boldsymbol{s}||_2.$$

We consider two cases below. Depending on the norm of $\boldsymbol{v}_1$, we modify our choice of projection direction. In the first case, we do not have a guarantee on the ordering of the means, so we use the first part of Lemma 2.13. In the second case, we can use the better bound in the second part of the lemma after arguing about the arrangement of the means.

Case 1 ($2\,||\boldsymbol{v}_1||_2 \geq \min(||\boldsymbol{v}_2||_2, ||\boldsymbol{v}_3||_2)$ and $\sqrt{\lambda_{\Sigma,\mathcal{U}}} = \Omega(||\boldsymbol{v}_1||_2)$): We start with a lemma that shows the existence of a vector $\boldsymbol{z}$ that is correlated with $\{\boldsymbol{v}_1, \boldsymbol{v}_2, \boldsymbol{v}_3\}$. We use $\boldsymbol{z}$ to define the direction $\boldsymbol{t}$ to project the means on, while roughly preserving their pairwise distances.

Lemma 2.14. *For $\boldsymbol{v}_1, \boldsymbol{v}_2, \boldsymbol{v}_3$ defined above, there exists a vector $\boldsymbol{z} \in \mathbb{R}^d$ such that $||\boldsymbol{z}||_2 \leq 10$, $\boldsymbol{z}$ belongs to the subspace spanned by $\boldsymbol{v}_1, \boldsymbol{v}_2, \boldsymbol{v}_3$, and $|\langle \boldsymbol{z}, \boldsymbol{v} \rangle| \geq \frac{||\boldsymbol{v}||_2}{6}$ for all $\boldsymbol{v} \in \{\boldsymbol{v}_1, \boldsymbol{v}_2, \boldsymbol{v}_3\}$.*

Proof. We use the probabilistic method. Let $\boldsymbol{u}_1, \boldsymbol{u}_2, \boldsymbol{u}_3$ be orthonormal vectors forming a basis of the subspace spanned by $\boldsymbol{v}_1, \boldsymbol{v}_2$ and $\boldsymbol{v}_3$; hence, we can write the vectors $\boldsymbol{v}_1, \boldsymbol{v}_2$ and $\boldsymbol{v}_3$ as a linear combination of $\boldsymbol{u}_1, \boldsymbol{u}_2, \boldsymbol{u}_3$. Let us define a vector $\boldsymbol{z}$ randomly

generated from the subspace spanned by $\boldsymbol{v}_1, \boldsymbol{v}_2, \boldsymbol{v}_3$ as follows. Let p, q, r be independently sampled according to $\mathcal{N}(0,1)$. Then, define $\boldsymbol{z} = p\boldsymbol{u}_1 + q\boldsymbol{u}_2 + r\boldsymbol{u}_3$. By this construction, we have that $\langle \boldsymbol{z}, \boldsymbol{v} \rangle \sim \mathcal{N}(0, ||\boldsymbol{v}||_2^2)$ for all vectors $\boldsymbol{v} \in \{\boldsymbol{v}_1, \boldsymbol{v}_2, \boldsymbol{v}_3\}$, and further, $||\boldsymbol{z}||_2^2 = p^2 + q^2 + r^2$. Hence, for any $\boldsymbol{v} \in \{\boldsymbol{v}_1, \boldsymbol{v}_2, \boldsymbol{v}_3\}$, we have

$$\begin{aligned}\Pr\left(|\langle \boldsymbol{z}, \boldsymbol{v} \rangle| \leq ||\boldsymbol{v}||_2/6\right) &\leq \int_{-\frac{||\boldsymbol{v}||_2}{6}}^{\frac{||\boldsymbol{v}||_2}{6}} \frac{e^{-x^2/2||\boldsymbol{v}||_2^2}}{\sqrt{2\pi||\boldsymbol{v}||_2^2}} dx \\ &\leq \int_{-\frac{||\boldsymbol{v}||_2}{6}}^{\frac{||\boldsymbol{v}||_2}{6}} \frac{1}{\sqrt{2\pi||\boldsymbol{v}||_2^2}} dx \leq \frac{||\boldsymbol{v}||_2}{3\sqrt{2\pi||\boldsymbol{v}||_2^2}} \leq \frac{1}{3\sqrt{2\pi}}.\end{aligned}$$

Also, we can bound the norm of z by bounding p, q, r. We see that

$$\Pr(p > 5) \leq \int_5^\infty \frac{e^{-x^2/2}}{\sqrt{2\pi}} dx \leq \frac{1}{5} \int_5^\infty \frac{x e^{-x^2/2}}{\sqrt{2\pi}} dx \leq \frac{e^{-12.5}}{5}.$$

Similarly, $\Pr(p < -5) \leq e^{-12.5}/5$. Applying the same calculations to q and r and taking a union bound, we must have that with positive probability $||\boldsymbol{z}||_2 \leq \sqrt{p^2 + q^2 + r^2} \leq \sqrt{75} \leq 10$ and $|\langle \boldsymbol{z}, \boldsymbol{v} \rangle| \geq ||\boldsymbol{v}||_2/6$ for all $\boldsymbol{v} \in \{\boldsymbol{v}_1, \boldsymbol{v}_2, \boldsymbol{v}_3\}$, implying there exists a vector $\boldsymbol{z}$ that satisfies the claim. □

For this case, we will use the first part of Lemma 2.13. Let $\boldsymbol{z}$ be the vector guaranteed by Lemma 2.14. Setting $\boldsymbol{t} = \frac{\boldsymbol{z}}{\sqrt{\boldsymbol{z}^T \boldsymbol{\Sigma} \boldsymbol{z}}}$, then Lemma 2.14 implies that

$$|\langle \boldsymbol{t}, \boldsymbol{v} \rangle| = \frac{|\langle \boldsymbol{z}, \boldsymbol{v} \rangle|}{\sqrt{\boldsymbol{z}^T \boldsymbol{\Sigma} \boldsymbol{z}}} \geq \frac{||\boldsymbol{v}||_2}{6\sqrt{\boldsymbol{z}^T \boldsymbol{\Sigma} \boldsymbol{z}}} \qquad \text{for all } \boldsymbol{v} \in \{\boldsymbol{v}_1, \boldsymbol{v}_2, \boldsymbol{v}_3\}.$$

Recall that we defined $\lambda_{\Sigma,\mathcal{U}} \triangleq \max_{\substack{||\boldsymbol{u}||_2 = 1 \\ \boldsymbol{u} \in \mathsf{span}(\boldsymbol{v}_1, \boldsymbol{v}_2, \boldsymbol{v}_3)}} \boldsymbol{u}^T \Sigma \boldsymbol{u}$ to be the maximum amount a unit norm vector $\boldsymbol{u}$ belonging to the span of the vectors $\boldsymbol{v}_1, \boldsymbol{v}_2, \boldsymbol{v}_3$ is stretched by the matrix $\boldsymbol{\Sigma}$. Note that $\lambda_{\Sigma,\mathcal{U}}$ is also upper bounded by the maximum eigenvalue of $\boldsymbol{\Sigma}$. Now, using the fact that $||\boldsymbol{z}||_2 \leq 10$ and $\sqrt{\boldsymbol{z}^T \boldsymbol{\Sigma} \boldsymbol{z}} \leq \sqrt{\lambda_{\Sigma,\mathcal{U}}}\, ||\boldsymbol{z}||_2 \leq 10\sqrt{\lambda_{\Sigma,\mathcal{U}}}$, we obtain

$$|\langle \boldsymbol{t}, \boldsymbol{v} \rangle| \geq \frac{||\boldsymbol{v}||_2}{60\sqrt{\lambda_{\Sigma,\mathcal{U}}}} \qquad \text{for all } \boldsymbol{v} \in \{\boldsymbol{v}_1, \boldsymbol{v}_2, \boldsymbol{v}_3\}. \tag{2.7}$$

The part of Lemma 2.13 that we use depends on whether $\sqrt{\boldsymbol{t}^T\boldsymbol{\Sigma}\boldsymbol{t}} = \Omega\Big(\max\left(|\langle \boldsymbol{t}, \boldsymbol{v}_1\rangle|\right)\Big)$ or not. However, the second part of the lemma is stronger and implies the first part. Therefore, we simply use the lower bound in the first part of the lemma, and we see that

$$\begin{aligned}
&\left|\left| f^{\boldsymbol{t}}_{\boldsymbol{\mu}_0,\boldsymbol{\mu}_1} - f^{\boldsymbol{t}}_{\boldsymbol{\mu}'_0,\boldsymbol{\mu}'_1} \right|\right|_{\mathsf{TV}} \\
&= \Omega\Big(\min\Big(1, \max\left(|\langle \boldsymbol{t}, \boldsymbol{\mu}_0 - \boldsymbol{\mu}_1\rangle|, |\langle \boldsymbol{t}, \boldsymbol{\mu}'_0 - \boldsymbol{\mu}'_1\rangle|\right) \\
&\times \min_{\sigma\in\mathcal{H}} \max\left(|\langle \boldsymbol{t}, (\boldsymbol{\mu}_0 - \boldsymbol{\mu}'_{\sigma(0)})\rangle|, |\langle \boldsymbol{t}, (\boldsymbol{\mu}_1 - \boldsymbol{\mu}'_{\sigma(1)})\rangle|\right)\Big)\Big) \\
&\overset{(a)}{=} \Omega\Big(\min\Big(1, |\langle \boldsymbol{t}, \boldsymbol{v}_1\rangle| \min(|\langle \boldsymbol{t}, \boldsymbol{v}_2\rangle|, |\langle \boldsymbol{t}, \boldsymbol{v}_3\rangle|)\Big)\Big) \\
&\overset{(b)}{=} \Omega\Big(\min\Big(1, \frac{\|\boldsymbol{v}_1\|_2 \min(\|\boldsymbol{v}_2\|_2, \|\boldsymbol{v}_3\|_2)}{\lambda_{\Sigma,\mathcal{U}}}\Big)\Big),
\end{aligned}$$

wherein step (a), we used the following facts (from definitions):

$$\max\left(|\langle \boldsymbol{t}, \boldsymbol{\mu}_0 - \boldsymbol{\mu}_1\rangle|, |\langle \boldsymbol{t}, \boldsymbol{\mu}'_0 - \boldsymbol{\mu}'_1\rangle|\right) \geq |\langle \boldsymbol{t}, \boldsymbol{v}_1\rangle| \tag{2.8}$$

$$\max\left(|\langle \boldsymbol{t}, (\boldsymbol{\mu}_0 - \boldsymbol{\mu}'_0)\rangle|, |\langle \boldsymbol{t}, (\boldsymbol{\mu}_1 - \boldsymbol{\mu}'_1)\rangle|\right) \geq |\langle \boldsymbol{t}, \boldsymbol{v}_2\rangle| \tag{2.9}$$

$$\max\left(|\langle \boldsymbol{t}, (\boldsymbol{\mu}_0 - \boldsymbol{\mu}'_1)\rangle|, |\langle \boldsymbol{t}, (\boldsymbol{\mu}_1 - \boldsymbol{\mu}'_0)\rangle|\right) \geq |\langle \boldsymbol{t}, \boldsymbol{v}_3\rangle| \tag{2.10}$$

and in step (b), we used Eq. (2.7) for each $\boldsymbol{v} \in \{\boldsymbol{v}_1, \boldsymbol{v}_2, \boldsymbol{v}_3\}$.

Case 2 ($2\left|\left|\boldsymbol{v}_1\right|\right|_2 \leq \min(\left|\left|\boldsymbol{v}_2\right|\right|_2, \left|\left|\boldsymbol{v}_3\right|\right|_2)$ or $\sqrt{\lambda_{\Sigma,\mathcal{U}}} = O(\left|\left|\boldsymbol{v}_1\right|\right|_2)$): For this case, we will use the second part of Lemma 2.13. The random choice of $\boldsymbol{t}$ in Case 1 would have been sufficient for using the second part of Lemma 2.13 when $c\left|\left|\boldsymbol{v}_1\right|\right|_2 \leq \min(\left|\left|\boldsymbol{v}_2\right|\right|_2, \left|\left|\boldsymbol{v}_3\right|\right|_2)$ or $\sqrt{\lambda_{\Sigma,\mathcal{U}}} = O(\left|\left|\boldsymbol{v}_1\right|\right|_2)$ for some large constant c but with a deterministic choice of $\boldsymbol{t}$ that is described below, we can show that $c = 2$ is sufficient. Let $\boldsymbol{t} = \frac{\boldsymbol{v}}{\sqrt{\boldsymbol{v}^T\boldsymbol{\Sigma}\boldsymbol{v}}}$, where

$$\boldsymbol{v} = \frac{\boldsymbol{v}_2}{\left|\left|\boldsymbol{v}_2\right|\right|_2} + \frac{s\boldsymbol{v}_3}{\left|\left|\boldsymbol{v}_3\right|\right|_2} \text{ with } s = \mathsf{argmax}_{u\in\{-1,+1\}}\langle \boldsymbol{v}_2, u\boldsymbol{v}_3\rangle.$$

Notice that we must have $s\langle \boldsymbol{v}_2, \boldsymbol{v}_3\rangle > 0$ from the definition of s. Then we see that

$$
\begin{aligned}
|\langle \boldsymbol{v}, \boldsymbol{v}_1\rangle| &= \left| \frac{\langle \boldsymbol{v}_2, \boldsymbol{v}_1\rangle}{||\boldsymbol{v}_2||_2} + \frac{s\langle \boldsymbol{v}_3, \boldsymbol{v}_1\rangle}{||\boldsymbol{v}_3||_2} \right| \leq 2\,||\boldsymbol{v}_1||_2 \leq \min(||\boldsymbol{v}_2||_2\,, ||\boldsymbol{v}_3||_3) \\
|\langle \boldsymbol{v}, \boldsymbol{v}_2\rangle| &= \left| ||\boldsymbol{v}_2||_2 + \frac{s\langle \boldsymbol{v}_2, \boldsymbol{v}_3\rangle}{||\boldsymbol{v}_3||_2} \right| \geq ||\boldsymbol{v}_2||_2 \qquad (2.11) \\
|\langle \boldsymbol{v}, \boldsymbol{v}_3\rangle| &= \left| \frac{\langle \boldsymbol{v}_2, \boldsymbol{v}_3\rangle}{||\boldsymbol{v}_2||_2} + s\,||\boldsymbol{v}_3||_2 \right| = \left| \frac{s\langle \boldsymbol{v}_2, \boldsymbol{v}_3\rangle}{||\boldsymbol{v}_2||_2} + ||\boldsymbol{v}_3||_2 \right| \geq ||\boldsymbol{v}_3||_2\,. \qquad (2.12)
\end{aligned}
$$

The first inequality follows the norm bound on $\boldsymbol{v}_1$ for this case, the second inequality uses that the definition of $\boldsymbol{v}$ and s imply that the second term in the sum is non-negative, and the third inequality uses the same logic and the fact that $s \in \{-1, 1\}$.

We just showed that $|\langle \boldsymbol{v}, \boldsymbol{v}_1\rangle| \leq \min(|\langle \boldsymbol{v}, \boldsymbol{v}_2\rangle|\,, |\langle \boldsymbol{v}, \boldsymbol{v}_3\rangle|)$, and hence $\langle \boldsymbol{t}, \boldsymbol{v}_1\rangle \leq \min(\langle \boldsymbol{t}, \boldsymbol{v}_2\rangle, \langle \boldsymbol{t}, \boldsymbol{v}_3\rangle)$. This implies that the interval defined by one pair of projected means is not contained within the interval defined by the other pair of projected means. This means we can use the second part of Lemma 2.13. Furthermore, we also have $\boldsymbol{t}^T\Sigma\boldsymbol{t} = 1$. Finally, since $||\boldsymbol{v}||_2 \leq 2$, note that $\sqrt{\boldsymbol{v}^T\Sigma\boldsymbol{v}} \leq 2\sqrt{\lambda_{\Sigma,\mathcal{U}}}$. Using Lemma 2.13 with our choice of $\boldsymbol{t}$, we see that

$$
\begin{aligned}
&\left|\left| f^{\boldsymbol{t}}_{\boldsymbol{\mu}_0,\boldsymbol{\mu}_1} - f^{\boldsymbol{t}}_{\boldsymbol{\mu}'_0,\boldsymbol{\mu}'_1} \right|\right|_{\mathsf{TV}} \\
&= \Omega\Big(\min\Big(1, \min_{\sigma\in\mathcal{H}} \max(|\langle \boldsymbol{t}, (\boldsymbol{\mu}_0 - \boldsymbol{\mu}'_{\sigma(0)})\rangle|, |\langle \boldsymbol{t}, (\boldsymbol{\mu}_1 - \boldsymbol{\mu}'_{\sigma(1)})\rangle|)\Big)\Big) \\
&\overset{(a)}{=} \Omega\Big(\min\Big(1, \min\Big(|\langle \boldsymbol{t}, \boldsymbol{v}_2\rangle|\,, |\langle \boldsymbol{t}, \boldsymbol{v}_3\rangle| \Big)\Big)\Big) \overset{(b)}{=} \Omega\Big(\min\Big(1, \frac{\min(||\boldsymbol{v}_2||_2\,, ||\boldsymbol{v}_3||_2)}{\sqrt{\lambda_{\Sigma,\mathcal{U}}}}\Big)\Big).
\end{aligned}
$$

In step (a), we used Eq. (2.9) and (2.10), while in step (b), we used Eq. (2.11) and (2.12). The remaining case is when $\sqrt{\lambda_{\Sigma,\mathcal{U}}} = O(||\boldsymbol{v}_1||_2)$. The second part of Lemma 2.13 applies because we observe that $\sqrt{\boldsymbol{t}^T\boldsymbol{\Sigma}\boldsymbol{t}} = O\Big(\max\left(|\langle \boldsymbol{t}, \boldsymbol{\mu}_0 - \boldsymbol{\mu}_1\rangle|, |\langle \boldsymbol{t}, \boldsymbol{\mu}'_0 - \boldsymbol{\mu}'_1\rangle|\right) \Big)$. To see this, recall that $\boldsymbol{t}^T\boldsymbol{\Sigma}\boldsymbol{t} = 1$, and hence,

$$
\max\left(|\langle \boldsymbol{t}, \boldsymbol{\mu}_0 - \boldsymbol{\mu}_1\rangle|, |\langle \boldsymbol{t}, \boldsymbol{\mu}'_0 - \boldsymbol{\mu}'_1\rangle|\right)
$$

$$\geq |\langle \boldsymbol{t}, \boldsymbol{v}_1 \rangle| = \left| \frac{\boldsymbol{z}^T \boldsymbol{v}_1}{\sqrt{\boldsymbol{z}^T \boldsymbol{\Sigma} \boldsymbol{z}}} \right| \geq \frac{||\boldsymbol{v}_1||_2}{6\sqrt{\boldsymbol{z}^T \boldsymbol{\Sigma} \boldsymbol{z}}} \geq \frac{||\boldsymbol{v}_1||_2}{6\sqrt{\lambda_{\Sigma,\mathcal{U}}}} = \Omega(1) = \Omega\left(\sqrt{\boldsymbol{t}^T \boldsymbol{\Sigma} \boldsymbol{t}}\right).$$

Next, recall that Lemma 2.14 implies that $|\langle \boldsymbol{t}, \boldsymbol{v}_2 \rangle| \geq ||\boldsymbol{v}_2||/6$ and $|\langle \boldsymbol{t}, \boldsymbol{v}_3 \rangle| \geq ||\boldsymbol{v}_3||/6$. Then, using the second part of Lemma 2.13, we have that

$$\begin{aligned}
&\left|\left| f^{\boldsymbol{t}}_{\boldsymbol{\mu}_0, \boldsymbol{\mu}_1} - f^{\boldsymbol{t}}_{\boldsymbol{\mu}'_0, \boldsymbol{\mu}'_1} \right|\right|_{\mathsf{TV}} \\
&= \Omega\Big(\min\Big(1, \min_{\sigma \in \mathcal{H}} \max(|\langle \boldsymbol{t}, (\boldsymbol{\mu}_0 - \boldsymbol{\mu}'_{\sigma(0)}) \rangle|, |\boldsymbol{t}^T (\boldsymbol{\mu}_1 - \boldsymbol{\mu}'_{\sigma(1)})|)\Big)\Big) \\
&\overset{(a)}{=} \Omega\Big(\min\Big(1, \min\Big(|\langle \boldsymbol{t}, \boldsymbol{v}_2 \rangle|, |\langle \boldsymbol{t}, \boldsymbol{v}_3 \rangle| \Big)\Big)\Big) \overset{(b)}{=} \Omega\Big(\min\Big(1, \frac{\min(||\boldsymbol{v}_2||_2, ||\boldsymbol{v}_3||_2)}{\sqrt{\lambda_{\Sigma,\mathcal{U}}}}\Big)\Big).
\end{aligned}$$

Again in step (a), we used Eq. (2.9) and (2.10) while in step (b), we used Eq. (2.11) and (2.12). This completes the proof of Theorem 2.8.

□

2.3.4 Learning 1-dimensional mixture with Minimum Distance estimator

Here, we will prove Theorem 2.10 assuming Theorem 2.9, where the theorem lower bounds the TV distance of the difference between a candidate distribution and the true distribution.

Proof of Theorem 2.10. We will use the minimum distance estimator on the class of distributions $\mathcal{F} = \{f_{\mu_0,\mu_1} := 0.5\mathcal{N}(\mu_0, \sigma^2) + 0.5\mathcal{N}(\mu_1, \sigma^2) : \mu_0, \mu_1 \in \mathbb{R}\}$. Suppose for sake of deriving a contradiction that the minimum distance estimator returns a candidate mixture $f_{\widehat{\mu}_0,\widehat{\mu}_1} \in \mathcal{F}$ such $\widehat{\mu}_0, \widehat{\mu}_1$ is not a $\delta'-$estimate of f_{μ_0,μ_1} with probability at least $1 - \eta$. From Theorem 2.9, for any such $f_{\widehat{\mu}_0,\widehat{\mu}_1}$, if $\delta > \delta'$, then

$$||f_{\mu_0,\mu_1} - f_{\widehat{\mu}_0,\widehat{\mu}_1}||_{\mathsf{TV}} \geq \Omega\Big(\min\Big(1, \frac{\delta\delta'}{\sigma^2}\Big)\Big),$$

and if $\delta' \geq \delta$, then

$$||f_{\mu_0,\mu_1} - f_{\widehat{\mu}_0,\widehat{\mu}_1}||_{\mathsf{TV}} \geq \Omega\Big(\min\Big(1, \frac{\delta'}{\sigma}\Big)\Big).$$

Recall that $\mathcal{A} = \{\{x : f(x) > g(x)\} | f, g \in \mathcal{F}, f \neq g\}$ is the union of sets of intervals in the sample space where one of the distribution has larger density than the other. From a result of Moitra and Valiant, $\mathsf{VC}(\mathcal{A}) = O(1)$ [114, 99]. Consider the case when $\delta > \delta'$ and $n \geq C \cdot \max\left(1, \frac{\sigma^4}{\delta^2 \delta'^2}\right) \log^2 \eta^{-1}$, for C a sufficiently large universal constant. We have the following guarantee for the minimum distance estimator (see Chapter D for details).

Proposition 2.2. *Let $VC(\mathcal{A})$ denote the Vapnik-Chervonenkis dimension of $\mathcal{A}$. Given n samples $x^1, x^2, \ldots, x^n$ distributed according to $f \in \mathcal{F}$, with probability at least $1 - \eta$, the minimum distance estimator returns $f_{\theta^\star} \in \mathcal{F}$ satisfying*

$$||f_{\theta^\star} - f||_{\mathsf{TV}} \leq 3 \inf_{\theta \in \Theta} ||f_\theta - f||_{\mathsf{TV}} + 2\sqrt{\frac{2VC(\mathcal{A}) + \log \eta^{-1}}{2n}} + \frac{3}{2n}. \tag{2.13}$$

By substituting into the above proposition, where we use $\inf_{f \in \mathcal{F}} ||f - f_{\mu_0, \mu_1}|| = 0$, with probability at least $1 - \eta$, we have that

$$||f_{\widehat{\mu}_0, \widehat{\mu}_1} - f_{\mu_0, \mu_1}||_{\mathsf{TV}} \leq O\Big(\min\Big(1, \frac{\delta\delta'}{\sigma^2}\Big)\Big).$$

Similarly, when $\delta < \delta'$ and $n \geq C' \cdot \max\left(1, \frac{\sigma^2}{\delta'^2}\right) \log^2 \eta^{-1}$, for C' a sufficiently large universal constant, with probability at least $1 - \eta$

$$||f_{\widehat{\mu}_0, \widehat{\mu}_1} - f_{\mu_0, \mu_1}||_{\mathsf{TV}} \leq O\Big(\min\Big(1, \frac{\delta'}{\sigma}\Big)\Big).$$

In both cases, this leads to a contradiction when we take η sufficiently small. □

We can extend this result to the setting when the distribution according to which the samples are generated is not a mixture of Gaussians but is close in Total Variation distance to one of them. More formally, suppose that we obtain samples from a distribution $\mathcal{D}$ such that $\inf_{f \in \mathcal{F}} ||\mathcal{D} - f||_{\mathsf{TV}} \leq \gamma$ and further let $f_{\mu_0, \mu_1} =$

$\mathsf{argmin}_{f\in\mathcal{F}} ||\mathcal{D} - f||_{\mathsf{TV}}$. As usual, we will denote by δ the separation of f_{μ_0,μ_1}. Now, we have the following result:

Lemma 2.15. *Let δ' and $0 < \eta < C$ be fixed, for C a small absolute constant. If $\delta > \delta' > \gamma\sigma^2/\delta$, then we can (δ', η)-learn f_{μ_0,μ_1} from the minimum distance estimator over $\mathcal{F}$ using $s(\delta', \eta) = O(\max(1, \sigma^4/(\delta^2\delta'^2)) \log^2 \eta^{-1})$ samples. If $\delta' \geq \min(\delta, \gamma\sigma)$, then we only need $s(\delta', \eta) = O(\max(1, \sigma^2/\delta'^2) \log^2 \eta^{-1})$ samples.*

Proof. The proof is very similar to the proof of Theorem 2.10 except in the use of Proposition 6, we need to substitute $\inf_{f\in\mathcal{F}} ||\mathcal{D} - f||_{\mathsf{TV}} \leq O(\delta\delta'/\sigma^2)$ when $\delta > \delta'$ and $\inf_{f\in\mathcal{F}} ||\mathcal{D} - f||_{\mathsf{TV}} \leq O(\delta'/\sigma)$ when $\delta < \delta'$. □

2.3.5 Learning 1-dimensional mixture with Scheffe estimator

In this section, we analyze the running time and the sample complexity of Algorithm 2.1 (stated below), a polynomial time algorithm that learns f_{μ_0,μ_1} and prove Theorem 2.11.

Algorithm 2.1 Estimate the means μ_0, μ_1 of f_{μ_0,μ_1} in polynomial time

Require: $\delta', \eta, C > 0$ such that $|\mu_0 - \mu_1| \geq C$. Access to samples from $f_{\mu_0,\mu_1} \in \mathcal{F}$.

1: Set $\rho = \min\{\delta'/\log \eta^{-1}, \sigma/12\}$ and $\epsilon = 2\rho C/\sigma$
2: Compute $\widehat{\phi}_0, \widehat{\phi}_1$ (mean estimates) using Method of Moments$(\sigma, O(\log \eta^{-1}), 36 \log \eta^{-1})$; Algorithm 3.6.
3: **if** $|\widehat{\phi}_0 - \widehat{\phi}_1| \geq 3\sigma/4$ **then**
4: Compute $\widehat{\mu}_0, \widehat{\mu}_1$ using EM(σ); Algorithm 3.4. Return $\widehat{\mu}_0, \widehat{\mu}_1$.
5: **else**
6: Compute $\widehat{M}$ using Fit a Single Gaussian$(O(\sigma^2 \log \eta^{-1}/\epsilon^2))$; Algorithm 3.7.
7: Set $\widetilde{\mathcal{F}} = \left\{\frac{\mathcal{N}(a\rho,\sigma^2)}{2} + \frac{\mathcal{N}(-a\rho,\sigma^2)}{2}; |a| \leq \frac{5\sigma}{6\rho}, a \in \mathbb{Z}\right\}$,
$T = O(\max(1, \frac{\sigma^4}{\delta^2\delta'^2} \log \frac{\sigma}{\delta'}) \log^3 \eta^{-1})$.
8: **for** $i = 1, 2, \ldots, T$ **do**
9: Obtain sample y^i from f_{μ_0,μ_1}.
10: **end for**
11: Compute the Scheffe estimate $f_{\widehat{\mu}_0,\widehat{\mu}_1}$ from $\widetilde{\mathcal{F}}$ using $y^1, \ldots, y^T$.
12: Return $\widehat{\mu}_0, \widehat{\mu}_1$.
13: **end if**

Estimate the separation of f_{μ_0,μ_1} using the Method of Moments: We use the Method of Moments (see Step 3 in Algorithm 2.1) to compute course mean estimates $\widehat{\phi}_0, \widehat{\phi}_1$ of μ_0, μ_1. For ϵ a small constant, which we will choose soon, suppose we use $O(\lceil \epsilon^{-4} \rceil \log \eta^{-1})$ samples to recover $\widehat{\phi}_0$ and $\widehat{\phi}_1$ using the Method of Moments. According to the guarantee provided in Theorem 3.8 for $\delta' = 2\epsilon\sqrt{\sigma^2 + (\mu_1 - \mu_2)^2}$, with probability at least $1 - \eta$,

$$\left|\widehat{\phi}_i - \mu_i\right| \leq 2\epsilon\sqrt{\sigma^2 + (\mu_1 - \mu_2)^2} \quad \text{for } i = 0, 1.$$

Therefore, we have

$$|\mu_0 - \mu_1| - \left|\mu_0 - \widehat{\phi}_0\right| - \left|\mu_1 - \widehat{\phi}_1\right| \leq \left|\widehat{\phi}_0 - \widehat{\phi}_1\right| \leq |\mu_0 - \mu_1| + \left|\mu_0 - \widehat{\phi}_0\right| + \left|\mu_1 - \widehat{\phi}_1\right|$$
$$\left|\left|\widehat{\phi}_0 - \widehat{\phi}_1\right| - |\mu_0 - \mu_1|\right| \leq 4\epsilon\sqrt{\sigma^2 + (\mu_0 - \mu_1)^2}.$$

We choose $\epsilon = 1/256$, in which case the number of samples is $O(\log \eta^{-1})$ and

$$\left|\left|\widehat{\phi}_0 - \widehat{\phi}_1\right| - |\mu_0 - \mu_1|\right| \leq \frac{1}{64}\sqrt{\sigma^2 + (\mu_0 - \mu_1)^2} \leq \frac{\sigma}{64} + \frac{|\mu_0 - \mu_1|}{64}.$$

Therefore, if $|\widehat{\phi}_0 - \widehat{\phi}_1| \geq \frac{3\sigma}{4}$, then we must have that $|\mu_0 - \mu_1| \geq \frac{47\sigma}{65}$. On the other hand, if $|\widehat{\phi}_0 - \widehat{\phi}_1| \leq \frac{3\sigma}{4}$, then $|\mu_0 - \mu_1| \leq \frac{7\sigma}{9}$.

If the separation is large, i.e. $|\widehat{\phi}_0 - \widehat{\phi}_1| \geq 3\sigma/4$, run EM: If the test using Method of Moments returns $|\widehat{\phi}_0 - \widehat{\phi}_1| \geq 3\sigma/4$, then we use the EM algorithm to compute $\widehat{\mu}_0, \widehat{\mu}_1$, estimates of μ_0, μ_1. From the guarantees of Theorem 3.7, we can conclude that $O((\sigma^2/\delta'^2) \log \eta^{-1})$ samples are sufficient to (δ', η)-learn f_{μ_0,μ_1} when $|\mu_0 - \mu_1| \geq 47\sigma/65$. Further, from Theorem 3.7, the run-time of the EM algorithm is $\widetilde{O}(\frac{\sigma^2}{\delta'^2} \log \frac{\sigma}{\delta'})$, where the $\widetilde{O}(\cdot)$ notation hides logarithmic factors in η.

Estimate the mean of f_{μ_0,μ_1} up to precision ϵ: In Step 2 of Algorithm 2.1, we compute $\widehat{M}$, an estimate of the mean, $(\mu_0+\mu_1)/2$, of the mixture f_{μ_0,μ_1}. Using Theorem 3.5 directly with $\delta' = \epsilon/\sigma$, we can conclude that $O(\max(1, \sigma^2 \log \eta^{-1}/\epsilon^2)$ samples are sufficient to compute $\widehat{M}$ such that with probability at least $1-\eta$

$$\left|\widehat{M} - (\mu_0+\mu_1)/2\right| \leq \epsilon.$$

Further, the run-time of Step 2 in Algorithm 2.1 is $O(\sigma^2 \log \eta^{-1}/\epsilon^2)$.

If the estimated separation is small, i.e. $|\widehat{\phi}_0 - \widehat{\phi}_1| < 3\sigma/4$, use the Scheffe estimator: In this case, we obtain samples from f_{μ_0,μ_1} and subtract $\widehat{M}$ from each of them. The modified—or rather, approximately centered—samples are distributed according to

$$f_{\mu_0-\widehat{M},\mu_1-\widehat{M}} = \frac{1}{2}\mathcal{N}(\mu_0 - \widehat{M}, \sigma^2) + \frac{1}{2}\mathcal{N}(\mu_1 - \widehat{M}, \sigma^2).$$

The mean of the modified distribution $f_{\mu_0-\widehat{M},\mu_1-\widehat{M}}$ has magnitude at most ϵ. Next, we design the following set of candidate distributions:

$$\widetilde{\mathcal{F}} = \left\{ \frac{1}{2}\mathcal{N}(a\rho, \sigma^2) + \frac{1}{2}\mathcal{N}(-a\rho, \sigma^2); -\frac{5\sigma}{6\rho} \leq a \leq \frac{5\sigma}{6\rho}, a \in \mathbb{Z} \right\}.$$

$|\widetilde{\mathcal{F}}| \leq 5\sigma/3\rho$ and thus, we have a finite set of candidate distributions. Let $f^\star \in \widetilde{\mathcal{F}}$ be the distribution $f^\star = \frac{1}{2}\left(\mathcal{N}(a^\star\rho, \sigma^2) + \mathcal{N}(-a^\star\rho, \sigma^2)\right)$, such that $a^\star = \mathsf{argmin}_a |a\rho - \mu_0 - \widehat{M}| + |a\rho + \mu_1 + \widehat{M}|$. The following lemma says that f^* is not too far from $f_{\mu_0-\widehat{M},\mu_1-\widehat{M}}$ in TV distance.

Lemma 2.16. *Let $\delta' > 0$, $0 < \eta < 1$, $0 < \rho < \sigma/12$, and $\epsilon = \frac{2\rho\delta}{\sigma}$ be fixed. Further, let $f_{\mu_0,\mu_1} \in \mathcal{F}$ with separation δ and let $\widehat{M}$ be such that $|\widehat{M} - (\mu_0+\mu_1)/2| \leq \epsilon$. For*

$f^\star \in \widetilde{\mathcal{F}}$ *the distribution* $f^\star = \frac{1}{2}\left(\mathcal{N}(a^\star\rho, \sigma^2) + \mathcal{N}(-a^\star\rho, \sigma^2)\right)$ *with* $a^\star = \mathsf{argmin}_a |a\rho - \mu_0 - \widehat{M}| + |a\rho + \mu_1 + \widehat{M}|$ *satisfies*

$$\left|\left| f_{\mu_0 - \widehat{M}, \mu_1 - \widehat{M}} - f^\star \right|\right|_{\mathsf{TV}} \leq \frac{16\rho\delta}{\sigma^2}.$$

Proof. We will use the invariance property of TV distance (see [51]), which says that for any pair of random variables X and Y on a sample space Ω with densities f and g, respectively, for any bijection $T : \mathbb{R} \to \mathbb{R}$

$$\begin{aligned} ||f - g|||_{\mathsf{TV}} &= \sup_{B \in \Omega} |\Pr(X \in B) - \Pr(X \in B)| \\ &= \sup_{B \in \Omega} |\Pr(T(X) \in T(B)) - \Pr(T(X) \in T(B))| \,. \end{aligned}$$

For simplicity of calculations, we consider the bijective mapping $T(x) = x/\sigma$ and subsequently divide the samples by σ, after subtracting $\widehat{M}$. Again for simplicity of notation, we denote $b = \max(|\mu_0 - \widehat{M}|, |\mu_1 - \widehat{M}|)$ and $b + \zeta = \min(|\mu_0 - \widehat{M}|, |\mu_1 - \widehat{M}|)$. Because of our guarantees on $\widehat{M}$, we have $0 \geq \zeta \geq -2\epsilon$ and $2b + \zeta = \delta$. Now, consider $U, U^\star$ to be two random variables as defined below,

$$U = \begin{cases} \frac{b}{\sigma} & \text{with probability} \frac{1}{2} \\ -\frac{b+\zeta}{\sigma} & \text{with probability} \frac{1}{2}. \end{cases}$$

$$U^\star = \begin{cases} \frac{a^\star\rho}{\sigma} & \text{with probability} \frac{1}{2} \\ -\frac{a^\star\rho}{\sigma} & \text{with probability} \frac{1}{2}. \end{cases}$$

We now use the moment matching results on the TV distance between two Gaussian mixtures (see Chapter 3, [148]) to say that

$$\left|\left| f_{\mu_0 - \widehat{M}, \mu_1 - \widehat{M}} - f^\star \right|\right|_{\mathsf{TV}} \leq \left(\sum_{m=0}^{\infty} \frac{|\mathbb{E}[U^m] - \mathbb{E}[U^{\star\, m}]|^2}{m!} \right)^{1/2}.$$

For odd m, we have $\mathbb{E}[U^{\star\, m}] = 0$. We also have that

$$\mathbb{E}[U^m] = \frac{1}{2}\Big(\Big(\frac{b}{\sigma}\Big)^m - \Big(\frac{b+\zeta}{\sigma}\Big)^m\Big) = \frac{b^m}{2\sigma^m}\Big(1 - \Big(1+\frac{\zeta}{b}\Big)^m\Big).$$

As $\epsilon \le \delta/6$, we must have that $|\zeta| \le 2\epsilon \le \delta/3 < b$. Therefore, we can invoke Bernoulli's inequality ($(1+x)^r \ge 1 + rx$ for any $x \ge -1$ and any integer $r \ge 0$) and conclude that $\mathbb{E}[U^m] \le \frac{|\zeta| m}{2b}\Big(\frac{b}{\sigma}\Big)^m$. For odd m, it follows that

$$|\mathbb{E}[U^m] - \mathbb{E}[U^{\star\, m}]| \le \frac{|\zeta| m}{2b}\Big(\frac{b}{\sigma}\Big)^m \le \frac{\epsilon m}{b}\Big(\frac{b}{\sigma}\Big)^m$$

For even $m > 0$, $\mathbb{E}[U^{\star\, m}]$ no longer cancels out to 0. We use that $\max(|a^\star\rho - b|, |a^\star\rho - b - \zeta|) \le \rho + |\zeta|$ and Bernoulli's inequality to see the following:

$$\begin{aligned}
|\mathbb{E}[U^m] - \mathbb{E}[U^{\star\, m}]| &= \frac{1}{2}\left|\Big(\Big(\frac{b}{\sigma}\Big)^m + \Big(\frac{b+\zeta}{\sigma}\Big)^m - 2\Big(\frac{a^\star\rho}{\sigma}\Big)^m\Big)\right| \\
&\le \left|\Big(\frac{b}{\sigma}\Big)^m - \Big(\frac{b-\rho-|\zeta|}{\sigma}\Big)^m\right| \\
&\le \frac{(|\zeta|+\rho)m}{b}\Big(\frac{b}{\sigma}\Big)^m \le \frac{(2\epsilon+\rho)m}{b}\Big(\frac{b}{\sigma}\Big)^m.
\end{aligned}$$

Combining the cases of odd and even m, we see that

$$\Big|\Big|f_{\mu_0-\widehat{M},\mu_1-\widehat{M}} - f^\star\Big|\Big|_{\mathsf{TV}}^2 \le \sum_{m>0 \text{ odd}} \Big(\frac{\epsilon m}{b}\Big)^2 \frac{\Big(\frac{b}{\sigma}\Big)^{2m}}{m!} + \sum_{m>0 \text{ even}} \Big(\frac{(2\epsilon+\rho)m}{b}\Big)^2 \frac{\Big(\frac{b}{\sigma}\Big)^{2m}}{m!}.$$

As m^2 grows much slower than $m!$ and $b < \delta < \sigma$, we can upper bound the sums by constant factors times their first terms to see that

$$\Big|\Big|f_{\mu_0-\widehat{M},\mu_1-\widehat{M}} - f^\star\Big|\Big|_{\mathsf{TV}}^2 \le \Big(\Big(\frac{\epsilon}{b}\Big)^2\Big(\frac{b}{\sigma}\Big)^2 + 2\cdot\Big(\frac{(2\epsilon+\rho)}{b}\Big)^2\Big(\frac{b}{\sigma}\Big)^4\Big)(1+o(1)).$$

Since we chose $\epsilon = \frac{2\rho\delta}{\sigma}$ and $\rho \le \sigma/12$

$$\Big(\frac{\epsilon}{b}\Big)^2\Big(\frac{b}{\sigma}\Big)^2 = \epsilon^2/\sigma^2 = 4\rho^2\delta^2/\sigma^4 \le 16\rho\delta/\sigma^2.$$

□

From Theorem 2.9, for any candidate distribution $f_{c\eta,-c\eta} \in \widetilde{\mathcal{F}}$ with $\max(|c\eta - \mu_0 + \widehat{M}|, |c\eta + \mu_1 + \widehat{M}|) \geq \delta'$, it holds that $||f_{\mu_0,\mu_1} - f_{c\eta,-c\eta}||_{\mathsf{TV}} \geq \Omega\Big(\min\Big(1, \frac{\delta\delta'}{\sigma^2}\Big)\Big)$.

Now, we provide the proof of Theorem 2.11.

Proof of Theorem 2.11. We assume that some lower bound C on δ is known i.e. $\delta > C$. In Algorithm 2.1, we choose $\rho = \min\{\delta'/\log\eta^{-1}, \sigma/12\}$ and $\epsilon = 2\rho C/\sigma$. By using Theorem 3.5, the number of samples sufficient to compute $\widehat{M}$ such that $|\widehat{M} - (\mu_0 + \mu_1)/2| \leq \epsilon$ is at most

$$O(\sigma^2 \log\eta^{-1}/\epsilon^2) = O\left(\frac{\sigma^4 \log^3\eta^{-1}}{\delta^2\delta'^2}\right)$$

when $\rho = \delta'/\log\eta^{-1}$ and at most

$$O(\sigma^2 \log\eta^{-1}/\epsilon^2) = O\left(\sigma^2 \log\eta^{-1}/\delta^2\right)$$

when $\rho = \sigma/12$; so overall we can use the former bound. Let $\mathcal{A} = \{\{x : f(x) > g(x)\} | f, g \in \widetilde{\mathcal{F}}, f \neq g\}$ and $\mu_{n,A} = \frac{1}{n}\sum_{i=1}^{n} \mathbb{1}[x^i \in A]$. Given n samples $x^1, x^2, \ldots, x^n$ from a distribution $f_{\mu_0,\mu_1} \in \mathcal{F}$, the Scheffe estimator returns a distribution $f_{a\eta,-a\eta} \in \widetilde{\mathcal{F}}$ such that

$$||f_{a\eta,-a\eta} - f_{\mu_0,\mu_1}||_{\mathsf{TV}} \leq 9 \min_{f\in\mathcal{F}} ||f - f_{\mu_0,\mu_1}||_{\mathsf{TV}} + 8\sqrt{\frac{2\log|\widetilde{F}| + \log\eta^{-1}}{2n}}$$

From Lemma 2.16, there exists $a^\star$ such that $f_{a^\star\eta,-a^\star\eta} \in \widetilde{\mathcal{F}}$ and $||f_{\mu_0,\mu_1} - f_{a^\star\eta,-a^\star\eta}||_{\mathsf{TV}} \leq \frac{4\rho\delta}{\sigma^2}$. Using this and the fact that $|\widetilde{F}| \leq 5\sigma/3\rho$, the distribution $f_{a\eta,-a\eta} \in \widetilde{\mathcal{F}}$ returned by the Scheffe estimator satisfies

$$||f_{a\eta,-a\eta} - f_{\mu_0,\mu_1}||_{\mathsf{TV}} \leq \frac{36\rho\delta}{\sigma^2} + 8\sqrt{\frac{2\log(5\sigma/(3\rho)) + \log\eta^{-1}}{2n}}.$$

If $\rho = \delta'/\log\eta^{-1}$, then $\frac{36\rho\delta}{\sigma^2} = O(\delta\delta'/\sigma^2)$, and similarly if $\rho = \sigma/12$, then $\frac{36\rho\delta}{\sigma^2} = O(\delta\delta'/\sigma^2)$. Since we want the TV distance $||f_{a\eta,-a\eta} - f_{\mu_0,\mu_1}||_{\mathsf{TV}}$ to be sufficiently small so that the first term $36\rho\delta/\sigma^2$ dominates, we note that for any n such that (recall that $\delta > C$ for known C)

$$n = \Omega\Big(\max\Big(1, \frac{\sigma^4\log^3\eta^{-1}}{C^2\delta'^2}\log\frac{\sigma}{\delta'}\Big)\Big)$$

the Scheffe estimator is close to the unknown mixture, $||f_{a\eta,-a\eta} - f_{\mu_0,\mu_1}||_{\mathsf{TV}} \le O(\delta\delta'/\sigma^2)$. The mixture $f_{a\eta,-a\eta} \in \widetilde{\mathcal{F}}$ satisfies $\max(|a\eta - \mu_0 + \widehat{M}|, |a\eta + \mu_1 + \widehat{M}|) < \delta'$, as if it did not satisfy this property, then by Theorem 2.9, $||f_{\mu_0,\mu_1} - f_{a\eta,-a\eta}||_{\mathsf{TV}} \ge \Omega\Big(\min\Big(1, \frac{\delta\delta'}{\sigma^2}\Big)\Big)$, contradicting our upper bound on the TV distance for η sufficiently small. Finally, using that $|\widetilde{\mathcal{F}}| \le 5\sigma/3\rho$, the running time of the Scheffe estimator (see Chapter 6, [51]) is at most

$$n|\widetilde{\mathcal{F}}|^2 = O\Big(\min\Big(1, \frac{\sigma^6\log^3\eta^{-1}}{C^2\delta'^4}\log\frac{\sigma}{\delta'}\Big)\Big).$$

□

2.4 Trace Reconstruction

In this section, we study the sparse trace reconstruction problem, where we assume that the unknown string $\boldsymbol{x} \in \{0,1\}^n$ has at most k 1s. Our analysis for this setting is based on a simple reduction from trace reconstruction to learning a mixture of binomial distributions, followed by a new sample complexity guarantee for the latter problem. We introduce additional notations to study this problem:

Notation: Throughout, n is the length of the binary string being reconstructed, n_0 is the number of 0s, k is the number of 1s, i.e., the *sparsity* or *weight*. For matrices, n is the total number of entries, and we focus on square $\sqrt{n} \times \sqrt{n}$ matrices. For most of our results, we assume that n, n_0, k are known since, if not, they can easily be estimated using a polynomial number of traces. Let p denote the deletion probability

when the 1s and 0s are deleted with the same probability. We also study a channel where the 1s and 0s are deleted with different probabilities; in this case, p_0 is the deletion probability of a 0 and p_1 is the deletion probability of a 1. We refer to the corresponding channel as the (p_0, p_1)-Deletion Channel or the asymmetric deletion channel. It will also be convenient to define $q = 1 - p, q_0 = 1 - p_0$ and $q_1 = 1 - p_1$ as the corresponding retention probabilities. Throughout, m denotes the number of traces. For a natural number w we use the notation $[w] = \{1, \ldots, w\}$.

Our main result in this section is the following:

Theorem 2.13. *Let $q \equiv 1 - p$ be the retention probability and assume that $q = \Omega(k^{-1/2} \log^{1/2} n)$. If $\boldsymbol{x} \in \{0,1\}^n$ has at most k non-zeros, $\exp(O((k/q)^{1/3} \log^{2/3} n))$ traces suffice to recover $\boldsymbol{x}$ exactly, with high probability.*

As some points of comparison, note that there is a trivial $\exp(O(k/q + \log n))$ upper bound, which our result improves on with a polynomially better dependence on k/q in the exponent. The trivial bound is obtained by getting enough samples so that it is possible to obtain $\text{poly}(n)$ samples where none of the 1s are deleted. The best known result for the general case is $\exp(O((n/q)^{1/3}))$ [116, 45] and our result is a strict improvement when $k = o(n/\log^2 n)$. Note that since we have no restrictions on k in the statement, improving upon $\exp(O((k/q)^{1/3}))$ would imply an improved bound in the general setting.

Somewhat surprisingly, our actual result is considerably stronger (See 2.2 for a precise statement). We also obtain $\exp(O((k/q)^{1/3} \log^{2/3} n))$ sample complexity in an asymmetric deletion channel, where each 0 is deleted with probability extremely close to 1, but each 1 is deleted with probability $p = 1 - q$. With such a channel, all but a vanishingly small fraction of the traces contain only 1s, yet we are still able to exactly identify the location of every 0. Since we can accommodate $k = \Theta(n)$ this result also applies to the general case with an asymmetric channel, yielding improvements over De et al. [45] and Nazarov and Peres [116].

Our approach yields two new results: first, we obtain an $\exp(O((k/q_1)^{1/3}\log^{2/3} n))$ sample complexity bound for sparse trace reconstruction, and second, we show that this guarantee applies even if the deletion probability for 0s is very close to 1.

To establish our results, we introduce a slightly more challenging channel which we refer to as the *Austere Deletion Channel*. The bulk of the proof analyzes this channel, and we obtain results for the (p_0, p_1) channel via a simple reduction.

Theorem 2.14 (Austere Deletion Channel Trace Reconstruction)**.** *In the Austere Deletion Channel, all but exactly one 0 are deleted (the choice of which 0 to retain is made uniformly at random) and each 1 is deleted with probability p_1. For such a channel,*

$$m = \exp(O((k/q_1)^{1/3}\log^{2/3} n))$$

traces suffice for sparse trace reconstruction with high probability where $q_1 = 1 - p_1$, provided $q_1 = \Omega(\sqrt{k^{-1}\log n})$.

We will prove this result shortly, but we first derive our main result for this section as a simple corollary.

Corollary 2.2 (Deletion Channel Trace Reconstruction)**.** *For the (p_0, p_1)-deletion channel,*

$$m = q_0^{-1}\exp(O((k/q_1)^{1/3}\log^{2/3} n))$$

traces suffice for sparse trace reconstruction with high probability where $q_0 = 1 - p_0$ and $q_1 = 1 - p_1 = \Omega(\sqrt{k^{-1}\log n})$.

Proof. This follows from Theorem 2.14. By focusing on just a single 0, it is clear that the probability that a trace from the (p_0, p_1)-deletion channel contains at least one 0 is at least q_0. If among the retained 0s we keep one at random and remove the rest, we generate a sample from the austere deletion channel. Thus, with m samples from the (p_0, p_1) deletion channel, we obtain at least mq_0 samples from the austere channel and the result follows. Note that Theorem 2.13 is a special case where $p_0 = p_1 = p$. □

Remark 2.1. *Note that the case where q_1 is constant (a typical setting for the problem) and $k = o(\log n)$ is not covered by the corollary. However, in this case a simpler approach applies to argue that* $\mathrm{poly}(n)$ *traces suffice: with probability $q_1^k \geq 1/\mathrm{poly}(n)$ no 1s are deleted in the generation of the trace and given* $\mathrm{poly}(n)$ *such traces, we can infer the original position of each 1 based on the average position of each 1 in each trace.*

Remark 2.2. *Note that the weak dependence on q_0 ensures that as long as $q_0 = 1/\exp(O((k/q_1)^{1/3}\log^{2/3} n))$, we still have the* $\exp(O((k/q_1)^{1/3}\log^{2/3} n))$ *bound. Thus, our result shows that sparse trace reconstruction is possible even when zeros are retained with super-polynomially small probability.*

2.4.1 Reduction to Learning Binomial Mixtures

We prove Theorem 2.14 via a reduction from austere deletion channel trace reconstruction to learning binomial mixtures. Given a string $\boldsymbol{x}$ of length n, let r_i be the number of ones before the i^{th} zero in $\boldsymbol{x}$. For example, if $\boldsymbol{x} = 1001100$ then $r_1 = 1, r_2 = 1, r_3 = 3, r_4 = 3$. Note that the multi-set $\{r_1, r_2, \ldots, \}$ uniquely determines $\boldsymbol{x}$, that each $r_i \leq k$, and that the multi-set has size n_0. The reduction from trace reconstruction to learning binomial mixtures is appealingly simple:

1. Given traces $\boldsymbol{t}_1, \ldots, \boldsymbol{t}_m$ from the austere channel, let s_i be the number of leading ones in $\boldsymbol{t}_i$.

2. Observe that each s_i is generated by a uniform[§] mixture of $\mathrm{Bin}(r_1, q_1), \ldots, \mathrm{Bin}(r_{n_0}, q_1)$ where $q_1 = 1 - p_1$. Hence, learning $r_1, r_2, \ldots, r_{n_0}$ from $s_1, s_2, \ldots, s_m$ allows us to reconstruct $\boldsymbol{x}$.

[§]Note that since the r_i are not necessarily distinct some of the binomial distributions are the same.

We will say that a number x has t-precision if $10^y \times x \in \mathbb{Z}$ where $y \in \mathbb{Z}$ and $y = O(\log t)$. To obtain Theorem 2.14, we establish the following new guarantee for learning binomial mixtures.

Theorem 2.15 (Learning Binomial Mixtures)**.** *Let $\mathcal{M}$ be a mixture of $d = \mathrm{poly}(n)$ binomials:*

$$\textit{Draw sample from } \mathrm{Bin}(a_t, q) \textit{ with probability } \alpha_t$$

where $0 \leq a_1, \ldots, a_d \leq a$ are distinct integers, the values α_t have $\mathrm{poly}(n)$ precision, and $q = \Omega(\sqrt{a^{-1} \log n})$. Then $\exp(O((a/q)^{1/3} \log^{2/3} n))$ samples suffice to learn the parameters exactly with high probability.

Proof. Let $\mathcal{M}'$ be a mixture where the samples are drawn from $\sum_{t=1}^{d} \beta_t \mathrm{Bin}(b_t, q)$, where $0 \leq b_1, \ldots, b_d \leq a$ are distinct and the probabilities $\beta_t \in \{0, \gamma, 2\gamma, \ldots, 1\}$ where $1/\gamma = \mathrm{poly}(n)$. Consider the variational distance $\sum_t |A_t - B_t|$ between $\mathcal{M}$ and $\mathcal{M}'$ where

$$A_t = \mathbb{P}\text{sample from } \mathcal{M} \text{ is } t = \sum_{j=1}^{d} \alpha_j \binom{a_j}{t} q^t (1-q)^{a_j - t}$$
$$B_t = \mathbb{P}\text{sample from } \mathcal{M}' \text{ is } t = \sum_{j=1}^{d} \beta_j \binom{b_j}{t} q^t (1-q)^{b_j - t} \ .$$

We will show that the variational distance between $\mathcal{M}$ and $\mathcal{M}'$ is at least

$$\epsilon = \exp(-O((a/q)^{1/3} (\log 1/\gamma)^{2/3})) \ .$$

Since there are at most $((a+1) \cdot (1/\gamma + 1))^d$ possible choices for the parameters of $\mathcal{M}'$, standard union bound arguments show that

$$O(\log(((a+1) \cdot (1/\gamma + 1))^d)/\epsilon^2) = \exp(O((a/q)^{1/3} (\log 1/\gamma)^{2/3}))$$

samples are sufficient to distinguish $\mathcal{M}$ from all other mixtures.

To prove the total variation bound, observe that by applying the binomial formula, for any complex number w, we have

$$\begin{aligned}\sum_{t\geq 0}(A_t - B_t)w^t &= \sum_{t\geq 0} w^t\Big(\sum_{j\geq 0}\alpha_j\binom{a_j}{t}q^i(1-q)^{a_j-t} - \beta_j\binom{b_j}{t}q^i(1-q)^{b_j-t}\Big)\\ &= \sum_{j\geq 0}(\alpha_j z^{a_j} - \beta_j z^{b_j})\end{aligned}$$

where $z = qw + (1-q)$. Let $G(z) = \sum_{j\geq 0}(\alpha_j z^{a_j} - \beta_j z^{b_j})$ and apply the triangle inequality to obtain:

$$\sum_{t\geq 0}|A_t - B_t||w^t| \geq |G(z)| \ .$$

Note that $G(z)$ is a non-zero degree d polynomial with coefficients in the set

$$\{-1, \ldots, -2\gamma, -\gamma, 0, \gamma, 2\gamma, \ldots, 1\}.$$

We would like to find a z such that $G(z)$ has large modulus but $|w^t|$ is small, since this will yield a total variation lower bound. We proceed along similar lines to Nazarov and Peres [116] and De et al. [45]. It follows from Corollary 3.2 in Borwein and Erdélyi [24] that there exists $z \in \{e^{i\theta} : -\pi/L \leq \theta \leq \pi/L\}$ such that

$$|G(z)| \geq \gamma\exp(-c_1 L\log(1/\gamma))$$

for some constant $c_1 > 0$. For such a value of z, Nazarov and Peres [116] show that

$$|w| \leq \exp(c_2/(qL)^2)$$

for some constant $c_2 > 0$. Therefore,

$$\sum_{t\geq 0}|A_t - B_t|\exp(tc_2/(qL)^2) \geq \sum_{t\geq 0}|A_t - B_t||w^t| \geq |G(z)| \geq \gamma\exp(-c_1 L\log(1/\gamma))$$

For $t > \tau = 6qa$, by an application of the Chernoff bound, $A_t, B_t \leq 2^{-t}$, so we obtain

$$\underbrace{\sum_{t>\tau} 2^{-t} \exp(tc_2/(qL)^2)}_{=T_\tau} + \sum_{t=0}^{\tau} |A_t - B_t| \exp(\tau c_2/(qL)^2) \geq \gamma \exp(-c_1 L \log(1/\gamma)) \ .$$

$$\sum_{t=0}^{\tau} |A_t - B_t| \geq \frac{\gamma \exp(-c_1 L \log(1/\gamma))}{\exp(\tau c_2/(qL)^2)} - \frac{T_\tau}{\exp(\tau c_2/(qL)^2)} \tag{2.14}$$

$$\geq \frac{\gamma \exp(-c_1 L \log(1/\gamma))}{\exp(\tau c_2/(qL)^2)} - O(2^{-\tau}) \tag{2.15}$$

where the second equality follows from the assumption that $c_2/(qL^2) \leq (\ln 2)/2$ (which we will ensure when we set L) since,

$$\frac{T_\tau}{\exp(\tau c_2/(qL)^2)} = \frac{O(1) \cdot 2^{-\tau} \exp(\tau c_2/(qL)^2)}{\exp(\tau c_2/(qL)^2)} = O(2^{-\tau}) \ .$$

Set

$$L = c \sqrt[3]{\tau/(q^2 \log(1/\gamma))} = c \sqrt[3]{6a/(q \log(1/\gamma))}$$

for some sufficiently large constant c. This ensures that the first term of Eqn. 2.14 is

$$\exp(-O((a/q)^{1/3} \log^{2/3}(1/\gamma))).$$

Note that

$$\frac{c_2}{qL^2} < \frac{c_2}{qc^2(a/(q \log(1/\gamma)))^{2/3}} \leq \frac{c_2}{c^2} \cdot \left(\frac{\log(1/\gamma)}{aq^{1/2}}\right)^{2/3} \leq \frac{c_2}{c^2} \cdot \left(\frac{\log(1/\gamma)}{aq^2}\right)^{2/3}$$

and so by the assumption that $q = \Omega(\sqrt{\log(1/\gamma)/a})$ we may set the constant c large enough such that $c_2/(qL^2) \leq (\ln 2)/2$ as required. The second term of Eqn. 2.14 is a

lower order term given the assumption on q and thus we obtain the required lower bound on the total variation distance. □

Theorem 2.14 now follows from Theorem 2.15, since in the reduction, we have $d = O(n)$ binomials, one per 0 in $\boldsymbol{x}$, α_i is a multiple of $1/n_0$ and importantly, we have $a = k$. The key is that we have a polynomial with degree $a = k$ rather than a degree n polynomial as in the previous analysis.

Remark If all α_t are equal, Theorem 2.15 can be improved to $\text{poly}(n)\cdot\exp(O((a/p)^{1/3}))$ by using a more refined bound from Borwein and Erdélyi [24] in our proof. This follows by observing that if $\alpha_t = \beta_t = 1/d$, then $\sum_{j\geq 0}(\alpha_j z^{a_j} - \beta_j z^{s_j})$ is a multiple of a Littlewood polynomial and we may use the stronger bound $|G(z)| \geq \exp(-c_1 L)/d$, see Borwein and Erdélyi [24].

2.4.2 Lower Bound on Learning Binomial Mixtures

We now show that the exponential dependence on $a^{1/3}$ in Theorem 2.15 is necessary.

Theorem 2.16 (Binomial Mixtures Lower Bound)**.** *There exists subsets*

$$\{a_1, \ldots, a_k\} \neq \{b_1, \ldots, b_k\} \subset \{0, \ldots, a\}$$

such that if $\mathcal{M} = \sum_{i=1}^{k} \text{Bin}(a_i, 1/2)/k$ *and* $\mathcal{M}' = \sum_{i=1}^{k} \text{Bin}(b_i, 1/2)/k$*, then* $\|\mathcal{M} - \mathcal{M}'\|_{\mathsf{TV}} = \exp(-\Omega(a^{1/3}))$*. Thus,* $\exp(\Omega(a^{1/3}))$ *samples are required to distinguish* $\mathcal{M}$ *from* $\mathcal{M}'$ *with constant probability.*

Proof. Previous work [116, 45] shows the existence of two strings $\boldsymbol{x}, \boldsymbol{y} \in \{0,1\}^n$ such that $\sum_i |\boldsymbol{t}_i^{\boldsymbol{x}} - \boldsymbol{t}_i^{\boldsymbol{y}}| = \exp(-\Omega(n^{1/3}))$ where $\boldsymbol{t}_i^{\boldsymbol{z}}$ is the expected value of the ith element (element at ith position counted from beginning) of a string formed by applying the

$(1/2, 1/2)$-deletion channel to the string $\boldsymbol{z}$. We may assume $\sum_{i\in[n]} \boldsymbol{x}_i = \sum_{i\in[n]} \boldsymbol{y}_i \equiv k$ since otherwise

$$\begin{aligned}\sum_i |\boldsymbol{t}_i^{\boldsymbol{x}} - \boldsymbol{t}_i^{\boldsymbol{y}}| &\geq \left|\sum_i \boldsymbol{t}_i^{\boldsymbol{x}} - \sum_i \boldsymbol{t}_i^{\boldsymbol{y}}\right| \\ &= \left|\sum_{i\in[n]} \boldsymbol{x}_i/2 - \sum_{i\in[n]} \boldsymbol{y}_i/2\right| \geq 1/2\end{aligned}$$

which would contradict the assumption $\sum_i |\boldsymbol{t}_i^{\boldsymbol{x}} - \boldsymbol{t}_i^{\boldsymbol{y}}| = \exp(-\Omega(n^{1/3}))$.

Consider $\mathcal{M} = \sum_{i=1}^k \mathrm{Bin}(a_i, 1/2)/k$ and $\mathcal{M}' = \sum_{i=1}^k \mathrm{Bin}(b_i, 1/2)/k$, where a_i (b_i) is the number of coordinates preceding the ith 1 in $\boldsymbol{x}$ ($\boldsymbol{y}$). Note that

$$\boldsymbol{t}_i^{\boldsymbol{x}} = \sum_{r=1}^k \binom{a_r}{i} / 2^{a_r+1} \quad \text{and} \quad \boldsymbol{t}_i^{\boldsymbol{y}} = \sum_{r=1}^k \binom{b_r}{i} / 2^{b_r+1} \ ,$$

and so

$$\begin{aligned}\|\mathcal{M} - \mathcal{M}'\|_{TV} &= \sum_i |\mathbb{P}\mathcal{M} = i - \mathbb{P}\mathcal{M}' = i| \\ &= \sum_i \frac{1}{k} \left|\sum_{r=1}^k \binom{a_r}{i} / 2^{a_r} - \sum_{r=1}^k \binom{b_r}{i} / 2^{b_r}\right| \\ &= \frac{2}{k} \sum_i |\boldsymbol{t}_i^{\boldsymbol{x}} - \boldsymbol{t}_i^{\boldsymbol{y}}| = \exp(-\Omega(n^{1/3})) \ ,\end{aligned}$$

which proves the result. □

CHAPTER 3

MIXTURES OF SPARSE LINEAR REGRESSIONS

3.1 Introduction

In this chapter and the next, we propose a mixture of simple canonical hypotheses functions in order to model the data. We start by modeling numerical data where we assume that the dependent variable is a mixture of L linear functions of the features. We study this rich and expressive model in the setting where the unknown weight vectors corresponding to each linear function is sparse and further, we can query the label for a particular feature from an oracle as described in Chapter 1. This setting is a generalization of the very well-known Compressed Sensing setting but significantly more difficult since the information of which linear function the oracle response corresponds to is unknown. This problem was first introduced by [153] who showed sufficient conditions on the query complexity under several constraints only for the special case of $L = 2$. We study two objectives namely 1) *Parameter estimation* where we estimate the unknown weight vectors upto a certain desired error and 2) *Support Recovery* where our goal is to recover the non-zero indices of every unknown weight vector. We demonstrate algorithms with provable sample complexity guarantees for both the parameter estimation and support recovery problems under extremely mild conditions on the latent parameters for any general value of L. For the special case of $L = 2$, we provide efficient algorithms for parameter recovery without any assumptions. The results of this chapter can be found in [97], [109] and [69].

3.1.1 Parameter Estimation

Consider L unknown distinct vectors $\boldsymbol{\beta}^1, \boldsymbol{\beta}^2, \ldots, \boldsymbol{\beta}^L \in \mathbb{R}^n$ such that each is k-sparse, i.e., the number of non-zero entries in each $\boldsymbol{\beta}^i$ is at most k where k is some known parameter. We define an oracle $\mathcal{O}$ which, when queried with a vector $\boldsymbol{x} \in \mathbb{R}^n$, returns the noisy output $y \in \mathbb{R}$:

$$y = \langle \boldsymbol{x}, \boldsymbol{\beta} \rangle + \eta \tag{3.1}$$

where $\eta \sim \mathcal{N}(0, \sigma^2)$ is a zero-mean Gaussian random variable with variance σ^2 that represents the measurement noise and $\boldsymbol{\beta}$ is chosen uniformly* from the set $\mathcal{B} = \{\boldsymbol{\beta}^1, \boldsymbol{\beta}^2, \ldots, \boldsymbol{\beta}^L\}$. The goal is to recover all vectors in $\mathcal{B}$ by making a set of queries $\mathbf{x}_1, \mathbf{x}_2, \ldots, \mathbf{x}_m$ to the oracle. We refer to the values returned by the oracle given these queries as *samples*. Note that the case of $L = 1$ corresponds to the problem of compressed sensing. Our primary focus is on the sample complexity of the problem, i.e., minimizing the number of queries that suffices to recover the sparse vectors up to some tolerable error. Recall that the most relevant previous work is by Yin et al. [153]. For the noiseless case, i.e., $\eta = 0$, they show that $O(kL \log(kL))$ queries are sufficient to recover all vectors in $\mathcal{B}$ exactly with high probability. However, their result requires a restrictive assumption on the set of vectors and do not hold for an arbitrary set of sparse vectors. Specifically, they require that for any $\boldsymbol{\beta}, \boldsymbol{\beta}' \in \mathcal{B}$,

$$\boldsymbol{\beta}_j \neq \boldsymbol{\beta}'_j \quad \text{for each} \quad j \in \operatorname{supp}(\boldsymbol{\beta}) \cap \operatorname{supp}(\boldsymbol{\beta}') \ . \tag{3.2}$$

Their approach depends crucially on this assumption and this limits the applicability of their approach. Note that our results will not depend on such an assumption. For the noisy case, the approach taken by Yin et al. only handles the $L = 2$ case and they

*Many of our results can be generalized to non-uniform distributions but we will assume a uniform distribution throughout for the sake of clarity.

state the case of $L > 2$ as an important open problem. Resolving this open problem will be another one of our contributions. Furthermore, we also improve the sample complexity guarantees for the $L = 2$ significantly and also design algorithms for this specific case without any assumptions.

3.1.2 Support Recovery

Our objective in this setting is to recover the support of all unknown vectors $\boldsymbol{\beta}^1, \boldsymbol{\beta}^2, \ldots, \boldsymbol{\beta}^L \in \mathbb{R}^n$ while minimizing the number of queries for a fixed SNR. For our results to hold we further assume that the minimum magnitude of any non-zero entry of any unknown vector in $\mathcal{B}$ is known to be at least δ, i.e., $\min_{i\in[L]} \min_{j\in[n]:\boldsymbol{\beta}^i_j\neq 0} |\boldsymbol{\beta}^i_j| \geq \delta$. While approximate recovery guarantees of vectors can also be translated into support recovery, the parameter estimation results proved in this chapter for any general L are valid only under the restrictive assumption that the sparse vectors all belong to some scaled integer lattice. The result for $L = 2$ does not have any restriction, but it holds only for the case of two components. Here we provide results for support recovery of $L \geq 2$ unknown vectors that does not have any of the aforementioned restrictions and also have a polynomial dependence on the noise variance, sparsity and a near polynomial dependence on the number of unknown vectors.

The major building block of our algorithms for support recovery is low-rank tensor decomposition also known as Canonical Polyadic (CP) decomposition. Tensor decomposition has been widely used in parameter estimation in mixture models and latent variable models. We refer to the reader to [122] and the references therein for a detailed survey. We also crucially make use of combinatorial structures such as a general class of Union Free Families (UFF), see, [133], to recover the support. UFFs have been previously used in [1] and [68] for support recovery in linear classifiers.

Organization: The rest of the chapter is organized as follows: in Section 3.2, we present our results on parameter estimation for any number of unknown vectors under

certain assumptions; in particular, in Section 3.2.1 and Section 3.2.2, we present our results for the noiseless and noisy cases respectively. In Section 3.3, we provide efficient algorithms along with sample complexity guarantees for parameter estimation of two unknown vectors without any assumptions. In Section 3.4, we provide a general framework for support recovery for number of unknown vectors without any assumptions. All missing proofs in this chapter can be found in the Appendix in Chapter B.

Notation Let $\mathbf{1}_n$ denote a length n vector of all 1's. We will write $[n]$ to denote the set $\{1, \ldots, n\}$ and let $\mathcal{P}([n])$ be the power set of $[n]$. For a vector $\boldsymbol{\beta} \in \mathbb{R}^n$, let $\boldsymbol{\beta}_i$ denote its i-th coordinate. We will use $\mathsf{supp}(\boldsymbol{\beta}) \subset [n]$ to denote the support of the vector $\boldsymbol{\beta}$, i.e, the set of indices with non-zero entries in $\boldsymbol{\beta}$. We will abuse notations a bit, and also sometimes use $\mathsf{supp}(\boldsymbol{\beta})$ to denote the binary indicator vector of length n that takes 1 at index i if and only if $\boldsymbol{\beta}_i \neq 0$. For a vector $\boldsymbol{\beta} \in \mathbb{R}^n$ and subset $S \subset [n]$ of indices, let $\boldsymbol{\beta}|_S \in \mathbb{R}^{|S|}$ denote the vector $\boldsymbol{\beta}$ restricted to the indices in S. Finally, let $f : \mathcal{P}([n]) \times \{0,1\}^n \to \{0,1\}^n$ be a function that takes a binary vector $\boldsymbol{\beta} \in \{0,1\}^n$ and a subset $\mathcal{S} \subseteq [n]$ as input and returns another binary vector $\boldsymbol{\beta}'$ such that the indices of $\boldsymbol{\beta}$ corresponding to the the set $\mathcal{S}$ is flipped i.e. $\boldsymbol{\beta}'_i = \boldsymbol{\beta}_i \oplus 1$ if $i \in \mathcal{S}$ and $\boldsymbol{\beta}'_i = \boldsymbol{\beta}_i$ otherwise.

3.2 Parameter Estimation under Grid Assumption

3.2.1 Exact sparse vectors and noiseless samples

We first consider the easier case when there is no noise and all unknown vectors are k-sparse. In this case, we show that $O(kL \log(kL))$ queries suffice and that $\Omega(kL)$ queries are necessary. The approach we take is as follows: In compressed sensing, exact recovery of k-sparse vectors is possible by taking samples with an $m \times n$ matrix with any $2k$ columns linearly independent. Such matrices exists with $m = 2k$ (such as Vandermonde matrices) and are called MDS matrices. We use rows of such a matrix

repeatedly to generate samples. Since there are L different vectors in the mixture, with $O(L \log L)$ measurements with a row we will be able to see the samples corresponding to each of the L vectors with that row. However, even if this is true for measurements with each rows, we will still not be able to align measurements across the rows. For example, even though we will obtain $\langle \mathbf{x}, \boldsymbol{\beta}^\ell \rangle$ for all $\ell \in [L]$ and for all $\mathbf{x}$ that are rows of an MDS matrix, we will be unable to identify the samples corresponding to $\boldsymbol{\beta}^1$. To tackle this problem, we propose using a special type of MDS matrix that allows us to align measurements corresponding to the same $\boldsymbol{\beta}$s. After that, we just use the sparse recovery property of the MDS matrix to individually recover each of the vectors. The main result for this section is the following.

Theorem 3.1. *For a collection of L vectors $\boldsymbol{\beta}^1, \boldsymbol{\beta}^2, \ldots, \boldsymbol{\beta}^L \in \mathbb{R}^n$ such that $\|\boldsymbol{\beta}^i\|_0 \leq k \; \forall i \in [L]$, one can recover all of them exactly with probability at least $1 - 3/k$ with a total of $2kL \log Lk^2$ oracle queries. See Algorithm 3.1.*

Algorithm 3.1 `Noiseless Recovery` The algorithm for extracting recovering vectors via queries to oracle in noiseless setting.

Require: Number of unknown sparse vectors L, dimension n, sparsity k.

1: Let $t \sim_{\mathsf{Uniform}} \{0, 1, 2, \ldots, k^2L^2 - 1\}$ and define $\alpha_1, \alpha_2, \ldots, \alpha_{2k}$ where $\alpha_j = \frac{2kt+j}{2k^3L^2}$.
2: **for** $i = 1, 2, \ldots, 2k$ **do**
3: Make $L \log(Lk^2)$ oracle queries with vector $[1 \; \alpha_i \; \alpha_i^2 \; \ldots \; \alpha_i^{n-1}]$. Refer to these as a *batch*.
4: **end for**
5: **for** $i = 1, 2, \ldots, 2k$ **do**
6: For each batch of query responses corresponding to the same query vector, retain unique values and sort them in ascending order. Refer to this as the *processed batch*.
7: **end for**
8: Set matrix $\mathcal{Q}$ of dimension $2k \times L$ such that its j^{th} row is the processed batch corresponding to the query vector $[1 \; \alpha_j \; \alpha_j^2 \; \ldots \; \alpha_j^{n-1}]$
9: **for** $i = 1, 2, \ldots, L$ **do**
10: Decode the i^{th} column of the matrix $\mathcal{Q}$ to recover $\boldsymbol{\beta}^i$.
11: **end for**
12: Return $\boldsymbol{\beta}^1, \boldsymbol{\beta}^2, \ldots, \boldsymbol{\beta}^L$.

A Vandermonde matrix is a matrix such that the entries in each row of the matrix are in geometric progression i.e., for an $m \times n$ dimensional Vandermonde matrix the entry in the (i, j)th entry is α_i^j where $\alpha_1, \alpha_2, \ldots, \alpha_m \in \mathbb{R}$ are distinct values. We will use the following useful property of the Vandermonde matrices; see, e.g., [70, Section XIII.8] for the proof. For the sake of completeness, we provide an alternative proof below:

Lemma 3.1. *The rank of any $m \times m$ square submatrix of a Vandermonde matrix is m assuming $\alpha_1, \alpha_2, \ldots, \alpha_m$ are distinct and positive.*

This implies that, with the samples from a $2k \times n$ Vandermonde matrix, a k-sparse vector can be exactly recovered. This is because for any two unknown vectors β and $\hat{\beta}$, the same set of responses for all the $2k$ rows of the Vandermonde matrix implies that a $2k \times 2k$ square submatrix of the Vandermonde matrix is not full rank which is a contradiction to Lemma 3.1.

Proof. Consider any $m \times m$ square submatrix Q of a $m \times n$ Vandermonde matrix

$$Q = \begin{bmatrix} \alpha_1^{\zeta_1} & \alpha_1^{\zeta_2} & \ldots & \alpha_1^{\zeta_m} \\ \alpha_2^{\zeta_1} & \alpha_2^{\zeta_2} & \ldots & \alpha_2^{\zeta_m} \\ \vdots & \vdots & \vdots & \vdots \\ \alpha_m^{\zeta_1} & \alpha_m^{\zeta_2} & \ldots & \alpha_m^{\zeta_m} \end{bmatrix}$$

where $\zeta_1, \zeta_2, \ldots, \zeta_m$ are distinct positive integers less than $n - 1$. The determinant of the square matrix Q is:

$$\det(Q) = \sum_{\tau \in S_m} \operatorname{sgn}(\tau) \prod_{i=1}^{m} Q_{i,\tau(i)}$$

where S_m is the permutation group of order m, $\operatorname{sgn}(\tau)$ is the parity of a permutation and $Q_{i,\tau(i)}$ denotes the entry in the i^{th} row and $\tau(i)^{\text{th}}$ column of Q. If we consider

$\alpha_1, \alpha_2, \ldots, \alpha_{m-1}$ to be fixed, then $\det(Q)$ is a polynomial in α_m of degree at most n but has at most m coefficients. Since there are at most m coefficients, there are at most $m-1$ sign changes and therefore by using Descartes' theorem [37], the number of positive roots of the polynomial $\det(Q)$ is at most $m-1$. On the other hand, if we replace α_m by any value in $\{\alpha_1, \alpha_2, \ldots, \alpha_{m-1}\}$, then the determinant becomes zero and therefore $\alpha_1, \alpha_2, \ldots, \alpha_{m-1}$ are roots of the polynomial $\det(Q)$. Therefore, if $\alpha_1, \alpha_2, \ldots, \alpha_{m-1}$ are all positive and distinct, then it must imply that

$$\det(Q) = P \prod_{i=1}^{m-1} (\alpha_m - \alpha_i)$$

where P does not have any positive root. Hence if α_m is positive and different from $\alpha_1, \alpha_2, \ldots, \alpha_{m-1}$ it must imply that $\det(\mathrm{Q}) \neq 0$ and therefore the $m \times m$ square submatrix Q is full rank. □

We are now ready to prove Theorem 3.1.

Proof. For the case of $L = 1$, note that the setting is the same as the well-known compressed sensing problem. Furthermore, suppose a $2k \times n$ matrix has the property that any $2k \times 2k$ submatrix is full rank, then using the rows of this matrix as queries is sufficient to recover any k-sparse vector. By Lemma 3.1, any $2k \times n$ Vandemonde matrix has the necessary property.

Let $\boldsymbol{\beta}^1, \boldsymbol{\beta}^2, \ldots, \boldsymbol{\beta}^L$ be the set of unknown k-sparse vectors. Notice that a particular row of the Vandermonde matrix looks like $[1\ z\ z^2\ z^3\ \ldots\ z^{n-1}]$ for some value of $z \in \mathbb{R}$. Therefore, for some vector $\boldsymbol{\beta}^i$ and a particular row of the Vandermonde matrix, the inner product of the two can be interpreted as a degree n polynomial evaluated at z such that the coefficients of the polynomial form the vector $\boldsymbol{\beta}^i$. More formally, the inner product can be written as $f^i(z) = \sum_{j=0}^{n-1} \boldsymbol{\beta}^i_j z^j$ where f^i is the polynomial corresponding to the vector $\boldsymbol{\beta}^i$. For any value $z \in \mathbb{R}^n$, we can define an ordering over the L polynomials $f^1, f^2, \ldots, f^L$ such that $f^i > f^j$ iff $f^i(z) > f^j(z)$.

For two distinct indices $i, j \in [L]$, we will call the polynomial $f^i - f^j$ a *difference polynomial*. Each difference polynomial has at most $2k$ non-zero coefficients and therefore has at most $2k$ positive roots by Descartes' Rule of Signs [37]. Since there are at most $L(L-1)/2$ distinct difference polynomials, the total number of distinct values that are roots of at least one difference polynomial is less than kL^2. Note that if an interval does not include any of these roots, then the ordering of $f^1, \ldots, f^L$ remains consistent for any point in that interval. In particular, consider the intervals $(0, \gamma], (\gamma, 2\gamma], \ldots, (1-\gamma, 1]$ where $\gamma = 1/(k^2L^2)$. At most kL^2 of these intervals include a root of a difference polynomial and hence if we pick a random interval then with probability at least $1 - 1/k$, the ordering of $f^1, \ldots, f^L$ are consistent throughout the interval. If the interval chosen is $(t\gamma, (t+1)\gamma]$ then set $\alpha_j = t\gamma + j\gamma/(2k)$ for $j = 1, \ldots, 2k$.

Now for each value of α_i, define the vector $\mathbf{x}_i \equiv [1\ \alpha_i\ \alpha_i^2\ \alpha_i^3\ \ldots\ \alpha_i^{n-1}]$. For each $i \in [2k]$, the vector $\mathbf{x}_i$ will be used as query to the oracle repeatedly for $L \log Lk^2$ times. We will call the set of query responses from the oracle for a fixed query vector $\mathbf{x}_i$ a *batch*. For a fixed batch and $\boldsymbol{\beta}^j$,

$$\begin{aligned}&\Pr(\boldsymbol{\beta}^j \text{ is not sampled by the oracle in the batch}) \leq \left(1 - \frac{1}{L}\right)^{L \log Lk^2} \\ &\leq e^{-\log Lk^2} = \frac{1}{Lk^2}.\end{aligned}$$

Taking a union bound over all the vectors (L of them) and all the batches ($2k$ of them), we get that in every batch every vector $\boldsymbol{\beta}^j$ for $j \in [L]$ is sampled with probability at least $1 - 2/k$. Now, for each batch, we will retain the unique values (there should be exactly L of them with high probability) and sort the values in each batch. Since the ordering of the polynomial remains same, after sorting, all the values in a particular position in each batch correspond to the same vector $\boldsymbol{\beta}^j$ for some unknown index $j \in [L]$. We can aggregate the query responses of all the batches in each position

and since there are $2k$ linear measurements corresponding to the same vector, we can recover all the unknown vectors $\boldsymbol{\beta}^j$ using Lemma 3.1. The failure probability of this algorithm is at most $3/k$. □

The following theorem establishes that our method is almost optimal in terms of sample complexity.

Theorem 3.2. *At least $2Lk$ oracle queries are necessary to recover an arbitrary set of L vectors that are k-sparse.*

Proof. It is known that for any particular vector β, at least $2k$ queries to the oracle are necessary in order to recover the vector exactly. Suppose the random variable X denotes the number of queries until the oracle has sampled the vector β at least $2k$ times. Notice that $X = \sum_{i=1}^{2k} X_i$ can be written as a sum of independent and identical random variables X_i distributed according to the geometric distribution with parameter $1/L$ where X_i denotes the number of attempts required to obtain the i^{th} sample after the $(i-1)^{\text{th}}$ sample has been made by the oracle. Since X is a sum of independent random variables, we must have

$$\mathbb{E}X = 2Lk \quad \text{and} \quad \text{Var}(X) = 2k(L^2 - L)$$

Therefore by using Chebychev's inequality [26], we must have

$$\Pr\left(X \leq 2Lk - k^{\frac{1}{4}}\sqrt{2k(L^2 - L)}\right) \leq \frac{1}{\sqrt{k}}$$

and therefore $X > 2Lk(1 - o(1))$ with high probability which proves the statement of the theorem. □

3.2.2 Noisy Samples and Sparse Approximation

Going forward, we will no longer assume vectors in $\mathcal{B}$ are sparse. From the noisy samples, our objective is to recover an estimate $\hat{\boldsymbol{\beta}}$ for each $\boldsymbol{\beta} \in \mathcal{B}$ such that

$$\|\boldsymbol{\beta} - \hat{\boldsymbol{\beta}}\| \leq c\|\boldsymbol{\beta} - \boldsymbol{\beta}^*\|, \tag{3.3}$$

where c is an absolute constant and $\boldsymbol{\beta}^*$ is the best k-sparse approximation of $\boldsymbol{\beta}$, i.e., all except the largest (by absolute value) k coordinates set to 0. The norms in the above equation can be arbitrary defining the strength of the guarantee, e.g., when we refer to an ℓ_1/ℓ_1 guarantee both norms are $\|\cdot\|_1$. Our results should be contrasted with [153], where results not only hold for only $L = 2$ and under assumption (3.2), but the vectors are also strictly k-sparse. However, like [153], we assume ϵ-precision of the unknown vectors, i.e., the value in each coordinate of each $\boldsymbol{\beta} \in \mathcal{B}$ is an integer multiple of ϵ.[†]

Notice that in this model the noise is additive and not multiplicative. Hence, it is possible to increase the ℓ_2 norm of the queries arbitrarily so that the noise becomes inconsequential. However, in a real setting, this cannot be allowed since increasing the strength (norm) of the queries has a cost and it is in our interest to minimize the cost. Suppose the algorithm designs the i^{th} query vector by first choosing a distribution Q_i and subsequently sampling a query vector $\boldsymbol{x}_i \sim Q_i$. Let us now define the signal to noise ratio as follows:

$$\mathsf{SNR} = \max_i \min_{\ell \in [L]} \frac{\mathbb{E}_{\boldsymbol{x}_i \sim Q_i} |\langle \boldsymbol{x}_i, \boldsymbol{\beta}^\ell \rangle|^2}{\mathbb{E}\eta^2} \ . \tag{3.4}$$

Our objective in the noisy setting is to recover the unknown vectors $\boldsymbol{\beta}^1, \boldsymbol{\beta}^2, \ldots, \boldsymbol{\beta}^L \in \mathbb{R}^n$ while minimizing the number of queries and the SNR at the same time. In

[†]Note that we do not assume ϵ-precision in the noiseless case.

this setting, assuming that all the unknown vectors have unit norm, we show that $O(k \log^3 n \exp((\sigma/\epsilon)^{2/3}))$ queries with $\mathsf{SNR} = O(1/\sigma^2)$ suffice to reconstruct the $L = O(1)$ vectors in $\mathcal{B}$ with the approximation guarantees given in Eq. (3.3) with high probability if the noise η is a zero mean gaussian with a variance of σ^2. This is equivalent to stating that $O(k \log^3 n \exp(1/(\epsilon\sqrt{\mathsf{SNR}})^{2/3}))$ queries suffice to recover the L unknown vectors with high probability.

Note that in the previous work $\epsilon\sqrt{\mathsf{SNR}}$ is assumed to be at least constant and, if this is the case, our result is optimal up to polynomial factors since $\Omega(k)$ queries are required even if $L = 1$. More generally, the dependence upon $\epsilon\sqrt{\mathsf{SNR}}$ in our result improves upon the dependence in the result by Yin et al. Note that we assumed $L = O(1)$ in our result because the dependence of sample complexity on L is complicated as it is implicit in the signal-to-noise ratio.

As in noiseless case, our approach is to use a compressed sensing matrix and use its rows multiple time as queries to the oracle. At the first step, we would like to separate out the different $\boldsymbol{\beta}$s from their samples with the same rows. Unlike the noiseless case, even this turns out to be a difficult task. Under the assumption of Gaussian noise, however, we are able to show that this is equivalent to learning a mixture of Gaussians with different means. In this case, the means of the Gaussians belong to an "ϵ-grid", because of the assumption on the precision of $\boldsymbol{\beta}$s. This is not a standard setting in the literature of learning Gaussian mixtures, e.g., [9, 72, 114]. Note that, this is possible if the vector that we are sampling with has integer entries. As we will see a binary-valued compressed sensing matrix will do the job for us. We will rely on a novel complex-analytic technique to exactly learn the means of a mixture of Gaussians, with means belonging to an ϵ-grid. This technique is paralleled by the recent developments in trace reconstructions where similar methods were used for learning a mixture of binomials [98, 116].

Once for each query, the samples are separated, we are still tasked with aligning them so that we know the samples produced by the same $\boldsymbol{\beta}$ across different queries. The method for the noiseless case fails to work here. Instead, we use a new method motivated by error-correcting codes. In particular, we perform several redundant queries, that help us to do this alignment. For example, in addition to the pair of queries $\mathbf{x}_i, \mathbf{x}_j$, we also perform the queries defined by $\mathbf{x}_i + \mathbf{x}_j$ and $\mathbf{x}_i - \mathbf{x}_j$.

After the alignment, we use the compressed sensing recovery to estimate the unknown vectors. For this, we must start with a matrix that with minimal number of rows, will allow us to recover any vector with a guarantee such as (3.3). On top of this, we also need the matrix to have integer entries so that we can use our method of learning a mixture of Gaussians with means on an ϵ-grid. Fortunately, a random binary ± 1 matrix satisfies all the requirements [15]. Putting now these three steps of learning mixtures, aligning and compressed sensing, lets us arrive at our results.

While we concentrate on sample complexity in this paper, our algorithm for the noiseless case is computationally efficient, and the only computationally inefficient step in the general noisy case is that of learning Gaussian mixtures. However, in practice one can perform a simple clustering (such as Lloyd's algorithm) to learn the means of the mixture. Our main result of this section is the following.

Theorem 3.3. *It is possible to recover approximations with the ℓ_1/ℓ_1 guarantee in Eq.* (3.3) *with probability at least $1 - 2/n$ of all the unknown vectors $\boldsymbol{\beta}^\ell \in \{0, \pm\epsilon, \pm 2\epsilon, \pm 3\epsilon, \dots\}^n, \ell = 1, \dots, L$ with $O(k(\log^3 n)\exp((\sigma/\epsilon)^{2/3})$ oracle queries where $\mathsf{SNR} = O(1/\sigma^2)$.*

Before we proceed with the ideas of proof, it would be useful to recall the *restricted isometry property* (RIP) of matrices in the context of recovery guarantees of (3.3). A matrix $\Phi \in \mathbb{R}^{m \times n}$ satisfies the (k, δ)-RIP if for any vector $\mathbf{z} \in \mathbb{R}^n$ with $\|\mathbf{z}\|_0 \leq k$,

$$(1 - \delta)\|\mathbf{z}\|_2^2 \leq \|\Phi \mathbf{z}\|_2^2 \leq (1 + \delta)\|\mathbf{z}\|_2^2. \tag{3.5}$$

It is known that if a matrix is $(2k, \delta)$-RIP with $\delta < \sqrt{2} - 1$, then the guarantee of (3.3) (in particular, ℓ_1/ℓ_1-guarantee and also an ℓ_2/ℓ_1-guarantee) is possible [29] with the *the basis pursuit* algorithm, an efficient algorithm based on linear programming. It is also known that a random ± 1 matrix (with normalized columns) satisfies the property with $c_s k \log n$ rows, where c_s is an absolute constant [15].

There are several key ideas of the proof. Since the case of $L = 2$ is simpler to handle, we start with that and then provide the extra steps necessary for the general case subsequently.

Algorithm 3.2 `Noisy Recovery for` $L = 2$ The algorithm for recovering best k-sparse approximation of vectors via queries to oracle in noisy setting.

Require: $\mathsf{SNR} = 1/\sigma^2$, Precision of unknown vectors ϵ, and the constant c_s where $c_s k \log n$ rows are sufficient for RIP in binary matrices.

1: **for** $i = 1, 2, \ldots, c_s k \log(n/k)$ **do**
2: Call SampleAndRecover($\mathbf{v}_i$) where $\mathbf{v}_i \sim_{\mathsf{Uniform}} \{+1, -1\}^n$.
3: **end for**
4: **for** $i \in [\log n]$ and $j \in [c_s k \log(n/k)]$ with $j \neq i$ **do**
5: Call SampleAndRecover($(\mathbf{v}^i + \mathbf{v}^j)/2$) and SampleAndRecover($(\mathbf{v}^i - \mathbf{v}^j)/2$)
6: **end for**
7: Choose vector $\mathbf{v}$ from $\{\mathbf{v}^1, \mathbf{v}^2, \ldots, \mathbf{v}^{\log n}\}$ such that $\langle \mathbf{v}, \boldsymbol{\beta}^1 \rangle \neq \langle \mathbf{v}, \boldsymbol{\beta}^2 \rangle$.
8: **for** $i = 1, 2, \ldots, k \log(n/k)$ and $\mathbf{v}^i \neq \mathbf{v}$ **do**
9: Label one of $\langle \mathbf{v}^i, \boldsymbol{\beta}^1 \rangle$,$\langle \mathbf{v}^i, \boldsymbol{\beta}^2 \rangle$ to be $\langle \mathbf{v}, \boldsymbol{\beta}^1 \rangle$ if their sum is in the pair $\langle \frac{\mathbf{v}^i + \mathbf{v}}{2}, \boldsymbol{\beta}^1 \rangle$,$\langle \frac{\mathbf{v}^i + \mathbf{v}}{2}, \boldsymbol{\beta}^2 \rangle$ and their difference is in the pair $\langle \frac{\mathbf{v} - \mathbf{v}^i}{2}, \boldsymbol{\beta}^1 \rangle$,$\langle \frac{\mathbf{v} - \mathbf{v}^i}{2}, \boldsymbol{\beta}^2 \rangle$. Label the other $\langle v, \boldsymbol{\beta}^2 \rangle$.
10: **end for**
11: Aggregate all (query, denoised query response pairs) labelled $\langle \mathbf{v}, \boldsymbol{\beta}^1 \rangle$ and $\langle \mathbf{v}, \boldsymbol{\beta}^2 \rangle$ separately and multiply all denoised query responses by a factor of $1/(\sqrt{c_s k \log(n/k)})$.
12: Return best k-sparse approximation of $\boldsymbol{\beta}^1$ and $\boldsymbol{\beta}^2$ by using Basis Pursuit algorithm on each aggregated cluster of (query, denoised query response) pairs.

13: **function** SampleAndRecover ($\boldsymbol{v}$)
14: Issue $T = c_2 \exp\left((\sigma/\epsilon)^{2/3}\right)$ queries to oracle with $\boldsymbol{v}$.
15: Return $\langle \boldsymbol{v}, \boldsymbol{\beta}^1 \rangle$, $\langle \boldsymbol{v}, \boldsymbol{\beta}^2 \rangle$ via min-distance estimator (Gaussian mixture learning, lemma 3.2).
16: **end function**

Gaussian Noise: Two vectors Algorithm 3.2 addresses the setting with only two unknown vectors. We will assume $\|\boldsymbol{\beta}^1\|_2 = \|\boldsymbol{\beta}^2\|_2 = 1$, so that we can subsequently show that the SNR is simply $1/\sigma^2$. This assumption is not necessary but we make this for the ease of presentation. The assumption of ϵ-precision for $\boldsymbol{\beta}^1, \boldsymbol{\beta}^2$ was made in Yin et al. [153], and we stick to the same assumption. On the other hand, Yin et al. requires further assumptions that we do not need to make. Furthermore, the result of Yin et al. is restricted to exactly sparse vectors, whereas our result holds for general sparse approximation.

For the two-vector case the result we aim to show is following.

Theorem 3.4. *Algorithm 3.2 uses* $O(k \log^3 n \exp((\sigma/\epsilon)^{2/3}))$ *queries to recover both the vectors* $\boldsymbol{\beta}^1$ *and* $\boldsymbol{\beta}^2$ *with an* ℓ_1/ℓ_1 *guarantee in Eq.* (3.3) *with probability at least* $1 - 2/n$.

This result is directly comparable with [153]. On the statistical side, we improve their result in several ways: (1) we improve the dependence on σ/ϵ in the sample complexity from $\exp(\sigma/\epsilon)$ to $\exp((\sigma/\epsilon)^{2/3})$,[‡] (2) our result applies for dense vectors, recovering the best k-sparse approximations, and (3) we do not need the overlap assumption (eq. (3.2)) used in their work.

Once we show $\mathsf{SNR} = 1/\sigma^2$, Theorem 3.4 trivially implies Theorem 3.3 in the case $L = 2$. Indeed, from Algorithm 3.2, notice that we have used vectors $\mathbf{v}$ sampled uniformly at random from $\{+1, -1\}^n$ and use them as query vectors. We must have $\mathbb{E}_{\mathbf{v}}|\langle \mathbf{v}, \boldsymbol{\beta}^\ell \rangle|^2 / \mathbb{E}\eta^2 = \|\boldsymbol{\beta}^\ell\|_2^2/\sigma^2 = 1/\sigma^2$ for $\ell = 1, 2$. Further, we have used the sum and difference query vectors which have the form $(\mathbf{v^1} + \mathbf{v^2})/2$ and $(\mathbf{v^1} - \mathbf{v^2})/2$ respectively where v_1, v_2 are sampled uniformly and independently from $\{+1, -1\}^n$. Therefore, we must have for $\ell = 1, 2$, $\mathbb{E}_{\mathbf{v^1}, \mathbf{v^2}}|\langle (\mathbf{v^1} \pm \mathbf{v^2})/2, \boldsymbol{\beta}^\ell \rangle|^2 / \mathbb{E}\eta^2 = 1/2\sigma^2$. According to our definition of SNR, we have that $\mathsf{SNR} = 1/\sigma^2$.

[‡]Note that [153] treat σ/ϵ as constant in their theorem statement, but the dependence can be extracted from their proof.

A description of Algorithm 3.2 that lead to proof of Theorem 3.4 can be found in Chapter B, Section B.1. We provide a short sketch here and state an important lemma that we will use in the more general case.

The main insight is that for a fixed sensing vector $\boldsymbol{v}$, if we repeatedly query with $\boldsymbol{v}$, we obtain samples from a mixture of Gaussians $\frac{1}{2}\mathcal{N}(\langle \boldsymbol{v}, \boldsymbol{\beta}^1 \rangle, \sigma^2) + \frac{1}{2}\mathcal{N}(\langle \boldsymbol{v}, \boldsymbol{\beta}^2 \rangle, \sigma^2)$. If we can *exactly* recover the means of these Gaussians, we essentially reduce to the noiseless case from the previous section. The first key step upper bounds the sample complexity for exactly learning the parameters of a mixture of Gaussians.

Lemma 3.2 (Learning Gaussian mixtures)**.** *Let $\mathcal{M} = \frac{1}{L}\sum_{i=1}^{L} \mathcal{N}(\mu_i, \sigma^2)$ be a uniform mixture of L univariate Gaussians, with known shared variance σ^2 and with means $\mu_i \in \epsilon\mathbb{Z}$. Then, for some constant $c > 0$ and some $t = \omega(L)$, there exists an algorithm that requires $ctL^2 \exp((\sigma/\epsilon)^{2/3})$ samples from $\mathcal{M}$ and exactly identifies the parameters $\{\mu_i\}_{i=1}^{L}$ with probability at least $1 - 2e^{-2t}$.*

If we sense with $\boldsymbol{v} \in \{-1, +1\}^n$ then $\langle \boldsymbol{v}, \boldsymbol{\beta}^1 \rangle, \langle \boldsymbol{v}, \boldsymbol{\beta}^2 \rangle \in \epsilon\mathbb{Z}$, so appealing to the above lemma, we can proceed assuming we know these two values exactly. Unfortunately, the sensing vectors here are more restricted — we must maintain bounded SNR and our technique of mixture learning requires that the means have finite precision — so we cannot simply appeal to our noiseless results for the alignment step. Instead we design a new alignment strategy, inspired by error correcting codes. Given two query vectors $\boldsymbol{v}^1, \boldsymbol{v}^2$ and the exact means $\langle \boldsymbol{v}^i, \boldsymbol{\beta}^j \rangle$, $i, j \in \{1, 2\}$, we must identify which values correspond to $\boldsymbol{\beta}^1$ and $\boldsymbol{\beta}^2$. In addition to sensing with any pair $\boldsymbol{v}^1$ and $\boldsymbol{v}^2$ we sense with $\frac{\boldsymbol{v}^1 \pm \boldsymbol{v}^2}{2}$, and we use these two additional measurements to identify which recovered means correspond to $\boldsymbol{\beta}^1$ and which correspond to $\boldsymbol{\beta}^2$. Intuitively, we can check if our alignment is correct via these reference measurements.

Therefore, we can obtain aligned, denoised inner products with each of the two parameter vectors. At this point we can apply a standard compressed sensing result as mentioned at the start of this section to obtain the sparse approximations of vectors.

General value of L In this setting, we will have $L > 2$ unknown vectors

$$\boldsymbol{\beta}^1, \boldsymbol{\beta}^2, \ldots, \boldsymbol{\beta}^L \in \mathbb{R}^n$$

of unit norm each from which the oracle can sample from with equal probability. We assume that L does not grow with n or k and as before, all the elements in the unknown vectors lie on a ϵ-grid. Here, we will build on the ideas for the special case of $L = 2$.

The main result of this section is the following.

Theorem 3.5. *Algorithm 3.3 uses* $O\Big(k(\log n)^3 \exp\Big((\frac{\sigma}{\epsilon})^{2/3}\Big)\Big)$ *queries with* $\mathsf{SNR} = O(1/\sigma^2)$ *to recover all the vectors* $\boldsymbol{\beta}^1, \ldots, \boldsymbol{\beta}^L$ *with* ℓ_1/ℓ_1 *guarantees in Eq.* (3.3) *with probability at least* $1 - 2/n$.

Theorem 3.3 follows as a corollary of this result.

The analysis of Algorithm 3.3 and the proofs of Theorems 3.3 and 3.5 are provided in detail in Chapter B, Section B.2. Below we sketch some of the main points of the proof.

There are two main hurdles in extending the steps explained for $L = 2$. For a query vector $\mathbf{v}$, we define the *denoised query means* to be the set of elements $\{\langle \mathbf{v}, \boldsymbol{\beta}^i \rangle\}_{i=1}^L$. Recall that a query vector $\mathbf{v}$ is defined to be *good* if all the elements in the set of denoised query means $\{\langle \mathbf{v}, \boldsymbol{\beta}^1 \rangle, \langle \mathbf{v}, \boldsymbol{\beta}^2 \rangle, \ldots, \langle \mathbf{v}, \boldsymbol{\beta}^L \rangle\}$ are distinct. For $L = 2$, the probability of a query vector $\mathbf{v}$ being *good* for $L = 2$ is at least $1/2$ but for a value of L larger than 2, it is not possible to obtain such guarantees without further assumptions. For a more concrete example, consider $L \geq 4$ and the unknown vectors $\boldsymbol{\beta}^1, \boldsymbol{\beta}^2, \ldots, \boldsymbol{\beta}^L$ to be such that $\boldsymbol{\beta}^i$ has $\mathtt{1}$ in the i^{th} position and zero everywhere else. If $\mathbf{v}$ is sampled from $\{+1, -1\}^n$ as before, then $\langle \mathbf{v}, \boldsymbol{\beta}^i \rangle$ can take values only in $\{-1, 0, +1\}$ and therefore it is not possible that all the values $\langle \mathbf{v}, \boldsymbol{\beta}^i \rangle$ are distinct. Secondly, even if we have a *good* query vector, it is no longer trivial to extend the

Algorithm 3.3 `Noisy Recovery for any constant` L The algorithm for recovering best k-sparse approximation of vectors via queries to oracle in noisy setting.

Require: $c', \alpha^\star, z^\star$ as defined in equations (A), (B) and (C) respectively, Variance of noise $\mathbb{E}\eta^2 = \sigma^2$ and precision of unknown vectors as ϵ.

for $i = 1, 2, \ldots, \sqrt{\alpha^\star}\log n + c'\alpha^\star k\log(n/k)$ **do**
  Let $\mathbf{v}^i \in_R \{+1, -1\}^n$, $\mathbf{r}_i \in_R \{-2z^\star, -2z^\star+1, \ldots, 2z^\star\}^n$, $q_i \in_R \{1, 2, \ldots, 4z^\star+1\}$
  Make $c_2 \exp((\sigma/\epsilon)^{2/3})$ queries to the oracle using each of the vectors $(q_i - 1)\mathbf{r_i}, \mathbf{v_i} + q_i\mathbf{r_i}$ and $\mathbf{v_i} + \mathbf{r_i}$.
  Recover $\langle\{(q_i-1)\mathbf{r}_i, \boldsymbol{\beta}^t\rangle\}_{t=1}^L$,$\{\langle\mathbf{v_i}+\mathbf{r_i}, \boldsymbol{\beta}^t\rangle\}_{t=1}^L$,$\{\langle\mathbf{v}^i + q_i\mathbf{r}_i, \boldsymbol{\beta}^t\rangle\}_{t=1}^L$ by using min-distance estimator (Gaussian mixture learning, lemma 3.2).
end for
for $i \in [\sqrt{\alpha^\star}\log n]$ and $j \in [\alpha^\star k\log(nk)]$ **do**
  Make $c_2 \exp((\sigma/\epsilon)^{2/3})$ queries to the oracle using the vector $\mathbf{r}_{i+j} + \mathbf{r}_i$.
  Recover $\{\langle\mathbf{r}_{i+j} + \mathbf{r}_i, \boldsymbol{\beta}^t\rangle\}_{t=1}^L$, by using the min-distance estimator (Gaussian mixture learning, Lemma 3.2).
end for
Choose vector $(\mathbf{v}^\star, \mathbf{r}^\star, q^\star)$ from $\{(\mathbf{v}^t, \mathbf{r}_t, q_t)\}_{t=1}^{\sqrt{\alpha^\star}\log n}$ such that $(\mathbf{v}^\star+\mathbf{r}^\star, (q-1)\mathbf{r}^\star, \mathbf{v}^\star + q^\star\mathbf{r}^\star)$ is good. Call a triplet $(\mathbf{v}+\mathbf{r}, (q-1)\mathbf{r}, \mathbf{v}+q\mathbf{r})$ to be good if no element in $\{\langle\mathbf{v}+q\mathbf{r}, \boldsymbol{\beta}^i\rangle\}_{i=1}^L$ can be written in two possible ways as sum of two elements, one each from $\{\langle\mathbf{v}, \boldsymbol{\beta}^i\rangle\}_{i=1}^L$ and $\{\langle\mathbf{v}+(q-1)\mathbf{r}, \boldsymbol{\beta}^i\rangle\}_{i=1}^L$.
Initialize $\mathcal{S}_j = \phi$ for $j = 1, \ldots, L$
for $i = \sqrt{\alpha^\star}\log n + 1, 2, \ldots, \sqrt{\alpha^\star}\log n + c'\alpha^\star k\log\frac{n}{k}$ **do**
  if $(\mathbf{v}^i + \mathbf{r}_i, (q_i-1)\mathbf{r}_i, \mathbf{v}^i + q\mathbf{r}_i)$ is matching good with respect to $(\mathbf{v}^\star + \mathbf{r}^\star, (q-1)\mathbf{r}^\star, \mathbf{v}^\star + q^\star\mathbf{r}^\star)$ (Call a triplet $(\mathbf{v}'+\mathbf{r}', (q'-1)\mathbf{r}', \mathbf{v}'+q'\mathbf{r}')$ to be matching good w.r.t a good triplet $(\mathbf{v}^\star+\mathbf{r}^\star, (q^\star-1)\mathbf{r}^\star, \mathbf{v}^\star+q^\star\mathbf{r}^\star)$ if $(\mathbf{v}'+\mathbf{r}', (q'-1)\mathbf{r}', \mathbf{v}'+q'\mathbf{r}')$ and $(\mathbf{r}', \mathbf{r}^\star, \mathbf{r}'+\mathbf{r}^\star)$ are good.) **then**
    Label the elements in $\{\langle\mathbf{v}^i, \boldsymbol{\beta}^t\rangle\}_{t=1}^L$ as described in Lemma B.8
    for $j = 1, 2, \ldots, L$ **do**
      $\mathcal{S}_j = \mathcal{S}_j \cup \{\langle\mathbf{v}^i, \boldsymbol{\beta}^t\rangle\}$ if label of $\langle\mathbf{v}^i, \boldsymbol{\beta}^t\rangle$ is $\langle\mathbf{r}^\star, \boldsymbol{\beta}^j\rangle$
    end for
  end if
end for
for $j = 1, 2, \ldots, L$ **do**
  Aggregate the elements of $\mathcal{S}_j$ and scale them by a factor of $1/c'k\log(n/k)$.
  Recover the vector $\boldsymbol{\beta}^j$ by using basis pursuit algorithms (compressed sensing decoding).
end for
Return $\boldsymbol{\beta}^1, \boldsymbol{\beta}^2, \ldots, \boldsymbol{\beta}^L$.

clustering or alignment step. Hence a number of new ideas are necessary to solve the problem for any general value of L.

We need to define a few constants which are used in the algorithm. Let $\delta < \sqrt{2}-1$ be a constant (we need a δ that allow k-sparse approximation given a $(2k,\delta)$-RIP matrix). Let c' be a large positive constant such that

$$\frac{\delta^2}{16} - \frac{\delta^3}{48} - \frac{1}{c'} > 0. \tag{A}$$

Secondly, let $\alpha^\star$ be another positive constant that satisfies the following for a given value of c',

$$\alpha^\star = \max\Big\{\alpha : \frac{\alpha^\alpha}{(\alpha-1)^{\alpha-1}} < \exp\Big(\frac{\delta^2}{16} - \frac{\delta^3}{48} - \frac{1}{c'}\Big)\Big\}. \tag{B}$$

Finally, for a given value of $\alpha^\star$ and L, let $z^\star$ be the smallest integer that satisfies the following:

$$z^\star = \min\Big\{z \in \mathbb{Z} : 1 - L^3\Big(\frac{3}{4z+1} - \frac{1}{4z^2+1}\Big) \geq \frac{1}{\sqrt{\alpha^\star}}\Big\}. \tag{C}$$

The Denoising Step. In each step of the algorithm, we sample a vector $\mathbf{v}$ uniformly at random from $\{+1,-1\}^n$, another vector $\mathbf{r}$ uniformly at random from $\mathcal{G} \equiv \{-2z^\star, -2z^\star+1, \ldots, 2z^\star-1, 2z^\star\}^n$ and a number q uniformly at random from $\{1,2,\ldots,4z^\star+1\}$. Now, we will use a batch of queries corresponding to the vectors $\mathbf{v}+\mathbf{r}, (q-1)\mathbf{r}$ and $\mathbf{v}+q\mathbf{r}$. We define a triplet of query vectors $(\mathbf{v}^1,\mathbf{v}^2,\mathbf{v}^3)$ to be *good* if for all triplets of indices $i,j,k \in [L]$ such that i,j,k are not identical,

$$\langle \mathbf{v}^1, \boldsymbol{\beta}^i\rangle + \langle \mathbf{v}^2, \boldsymbol{\beta}^j\rangle \neq \langle \mathbf{v}^3, \boldsymbol{\beta}^k\rangle.$$

We show that the query vector triplet $(\mathbf{v}+\mathbf{r}, (q-1)\mathbf{r}, \mathbf{v}+q\mathbf{r})$ is good with at least some probability. This implies if we choose $O(\log n)$ triplets of such query vectors,

then at least one of the triplets are good with probability $1 - 1/n$. It turns out that, for a good triplet of vectors $(\mathbf{v} + \mathbf{r}, (q-1)\mathbf{r}, \mathbf{v} + q\mathbf{r})$, we can obtain $\langle \mathbf{v}, \boldsymbol{\beta}^i \rangle$ for all $i \in [L]$.

Furthermore, it follows from Lemma 3.2 that for a query vector $\mathbf{v}$ with integral entries, a batch size of $T > c_3 \log n \exp((\sigma/\epsilon)^{2/3})$, for some constant $c_3 > 0$, is sufficient to recover the denoised query responses $\langle \mathbf{v}, \boldsymbol{\beta}^1 \rangle, \langle \mathbf{v}, \boldsymbol{\beta}^2 \rangle, \ldots, \langle \mathbf{v}, \boldsymbol{\beta}^L \rangle$ for all the queries with probability at least $1 - 1/\text{poly}(n)$.

The Alignment Step. Let a particular good query vector triplet be $(\mathbf{v}^\star + \mathbf{r}^\star, (q^\star - 1)\mathbf{r}^\star, \mathbf{v}^\star + q^\star \mathbf{r}^\star)$. From now, we will consider the L elements $\{\langle \mathbf{r}^\star, \boldsymbol{\beta}^i \rangle\}_{i=1}^L$ to be labels and for a vector $\mathbf{u}$, we will associate a label with every element in $\{\langle \mathbf{u}, \beta^i \rangle\}_{i=1}^L$. The labelling is correct if, for all $i \in [L]$, the element labelled as $\langle \mathbf{r}^\star, \boldsymbol{\beta}^i \rangle$ also corresponds to the same unknown vector $\boldsymbol{\beta}^i$. Notice that we can label the elements $\{\langle \mathbf{v}^\star, \beta^i \rangle\}_{i=1}^L$ correctly because the triplet $(\mathbf{v}^\star + \mathbf{r}^\star, (q^\star - 1)r^\star, \mathbf{v}^\star + q^\star \mathbf{r}^\star)$ is good. Consider another good query vector triplet $(\mathbf{v}' + \mathbf{r}', (q'-1)\mathbf{r}', \mathbf{v}' + q'\mathbf{r}')$. This *matches* with the earlier query triplet if additionally, the vector triplet $(\mathbf{r}', \mathbf{r}^\star, \mathbf{r}' + \mathbf{r}^\star)$ is also good.

Such matching pair of good triplets exists, and can be found by random choice with some probability. We show that, the matching good triplets allow us to do the alignment in the case of general $L > 2$.

At this point we would again like to appeal to the standard compressed sensing results. However we need to show that the matching good vectors themselves form a matrix that has the required RIP property. As our final step, we establish this fact.

Remark 3.1 (Refinement and adaptive queries)**.** *It is possible to have a sample complexity of* $O\Big(k(\log n)^2 \log k \exp\Big((\epsilon\sqrt{\mathsf{SNR}})^{-2/3}\Big)\Big)$ *in Theorem 3.3, but with a probability of* $1 - \text{poly}(k^{-1})$*. Also it is possible to shave-off another* $\log n$ *factor from sample complexity if we can make the queries adaptive.*

3.3 Parameter Estimation for two unknown vectors

3.3.1 Our Techniques and Results

In this section, we provide generic sample complexity results for the special case of two unknown vectors i.e. for $\mathcal{B} \equiv \{\boldsymbol{\beta}^1, \boldsymbol{\beta}^2\}$ without making any assumptions on the unknown vectors. To make the problem well-posed, let us make a slightly different definition of Signal-to-Noise Ratio (SNR) for a query $\boldsymbol{x}$:

$$\mathsf{SNR}(\boldsymbol{x}) \triangleq \frac{\mathbb{E}|\langle \boldsymbol{x}, \boldsymbol{\beta}^1 - \boldsymbol{\beta}^2 \rangle|^2}{\mathbb{E}\eta^2}$$

where the expectation is over the randomness of the query. Furthermore define the overall SNR to be $\mathsf{SNR} := \max_{\boldsymbol{x}} \mathsf{SNR}(\boldsymbol{x})$, where the maximization is over all the queries used in the recovery process. For a vector $\boldsymbol{\beta} \in \mathbb{R}^n$, the best k-sparse approximation $\boldsymbol{\beta}_{(k)}$ is defined to be the vector obtained from $\boldsymbol{\beta}$ where all except the k-largest (by absolute value) coordinates are set to 0. For each $\boldsymbol{\beta} \in \{\boldsymbol{\beta}^1, \boldsymbol{\beta}^2\}$, our objective in this setting is to return a *sparse approximation* of $\hat{\boldsymbol{\beta}}$ using minimum number of queries such that

$$||\hat{\boldsymbol{\beta}} - \boldsymbol{\beta}|| \leq c||\boldsymbol{\beta} - \boldsymbol{\beta}_{(k)}|| + \gamma$$

Our main result is following:

Theorem 3.6. *[Main Result] Let* $\mathsf{NF} := \frac{\gamma}{\sigma}$ *(the* noise factor*) where* $\gamma > 0$ *is a parameter representing the desired recovery precision and* $\sigma > 0$ *is the standard deviation of* η *in Eq.* (3.1)*.*

Case 1. For any $\gamma < \left|\left|\boldsymbol{\beta}^1 - \boldsymbol{\beta}^2\right|\right|_2 /2$*, there exists an algorithm that makes*

$$O\left(k \log n \log k \left\lceil \frac{\log k}{\log(\sqrt{\mathsf{SNR}}/\mathsf{NF})} \right\rceil \left\lceil \frac{1}{\mathsf{NF}^4\sqrt{\mathsf{SNR}}} + \frac{1}{\mathsf{NF}^2} \right\rceil \right)$$

queries to recover $\hat{\boldsymbol{\beta}}^1, \hat{\boldsymbol{\beta}}^2$*, estimates of* $\boldsymbol{\beta}^1, \boldsymbol{\beta}^2$*, with high probability such that , for* $i = 1, 2$*,*

$$||\hat{\boldsymbol{\beta}}^{\pi(i)} - \boldsymbol{\beta}^i||_2 \leq \frac{c||\boldsymbol{\beta}^i - \boldsymbol{\beta}^i_{(k)}||_1}{\sqrt{k}} + O(\gamma)$$

where $\pi : \{1, 2\} \to \{1, 2\}$ *is some permutation of* $\{1, 2\}$ *and* c *is a universal constant.*

Case 2. For any $\gamma = \Omega(\left|\left|\boldsymbol{\beta}^1 - \boldsymbol{\beta}^2\right|\right|_2)$*, there exists an algorithm that makes* $O\Big(k \log n \Big\lceil \frac{\log k}{\mathsf{SNR}} \Big\rceil\Big)$ *queries to recover* $\hat{\boldsymbol{\beta}}$*, estimates of both* $\boldsymbol{\beta}^1, \boldsymbol{\beta}^2$*, with high probability such that , for both* $i = 1, 2$*,*

$$||\hat{\boldsymbol{\beta}} - \boldsymbol{\beta}^i||_2 \leq \frac{c||\boldsymbol{\beta}^i - \boldsymbol{\beta}^i_{(k)}||_1}{\sqrt{k}} + O(\gamma)$$

where c *is a universal constant.*

For a $\gamma = \Theta(\left|\left|\boldsymbol{\beta}^1 - \boldsymbol{\beta}^2\right|\right|_2)$ the first case of the Theorem holds but using the second case may give better result in that regime of precision. The second case of the theorem shows that if we allow a rather large precision error, then the number of queries is similar to the required number for recovering a single vector. This is expected, because in this case we can find just one line approximating both regressions.

The recovery guarantee that we are providing (an ℓ_2-ℓ_1 guarantee) is in line with the standard guarantees of the compressed sensing literature. In this paper, we are interested in the regime $\log n \leq k \ll n$ as in compressed sensing. Note that, our number of required samples scales linearly with k and has only poly-logarithmic scaling with n, and polynomial scaling with the noise σ. In the previous works as well as the general result in the previous section, the complexities scaled exponentially with noise.

Furthermore, the query complexity of our algorithm decreases with the Euclidean distance between the vectors (or the 'gap') - which makes sense intuitively. Consider the case when when we want a precise recovery (γ very small). It turns out that when

the gap is large, the query complexity varies as $O((\log \text{gap})^{-1})$ and when the gap is small the query complexity scale as $O((\text{gap} \log \text{gap})^{-1})$.

Remark 3.2 (The zero noise case)**.** *When $\sigma = 0$, i.e., the samples are not noisy, the problem is still nontrivial, and is not covered by the statement of Theorem 3.6. However this case is strictly simpler to handle as it will involve only the alignment step (as will be discussed later), and not the mixture learning step. Recovery with $\gamma = 0$ is possible with only $O(k \log n \log k)$ queries.*

If the responses to the queries were to contain tags of the models they are coming from, then we could use rows of any standard compressed sensing matrix as queries and just segregate the responses using the tags. Then by running a compressed sensing recovery on the groups with same tags, we would be done. In what follows, we try to infer this 'tag' information by making redundant queries.

If we repeat just the same query multiple time, the noisy responses are going to come from a mixture of Gaussians, with the the actual responses being the component means. To learn the actual responses we rely on methods for parameter learning in Gaussian mixtures. It turns out that for different parameter regimes, different methods are best-suited for our purpose - and it is not known in advance what regime we would be in. The method of moments is a well-known procedure for parameter learning in Gaussian mixtures and rigorous theoretical guarantees on sample complexity exist [72]. However we are in a specialized regime of scalar uniform mixtures with known variance; and we leverage these information to get better sample complexity guarantee for exact parameter learning. In particular we show that, in this case the mean and variance of the mixture are sufficient statistics to recover the unknown means, as opposed to the first six moments of the general case [72]. While recovery using other methods are straight forward adaption of known literature, we show that only a small set of samples are needed to determine what method to use.

It turns out that method of moments still needs significantly more samples than the other methods. However we can avoid using method of moments and use a less intensive method (such as EM, Algorithms), provided we are in a regime when the gap between the component means is high. The only fact that the Euclidean distance between $\boldsymbol{\beta}^1$ and $\boldsymbol{\beta}^2$ are far does not guarantee that. However, if we choose the queries to be Gaussians, then the gap is indeed high with certain probability. If the queries were to be generated by any other distribution, then such fact will require strong anti-concentration inequalities that in general do not hold. Therefore, we cannot really work with any standard compressed sensing matrix, but have to choose Gaussian matrices (which are incidentally also good standard compressed sensing matrices).

The main technical challenge comes in the next step, alignment. For any two queries $\boldsymbol{x}, \boldsymbol{x}'$, even if we know $y_1 = \langle \boldsymbol{\beta}^1, \boldsymbol{x} \rangle, y_2 = \langle \boldsymbol{\beta}^2, \boldsymbol{x} \rangle$ and $y_1' = \langle \boldsymbol{\beta}^1, \boldsymbol{x}' \rangle, y_2' = \langle \boldsymbol{\beta}^2, \boldsymbol{x}' \rangle$, we do not know how to club y_1 and y_1' together as their order could be different. And this is an issue with all pairs of queries which leaves us with exponential number of possibilities to choose form. We form a simple error-correcting code to tackle this problem.

For two queries, $\boldsymbol{x}, \boldsymbol{x}'$, we deal with this issue by designing two additional queries $\boldsymbol{x} + \boldsymbol{x}'$and $\boldsymbol{x} - \boldsymbol{x}'$. Now even if we mis-align, we can cross-verify with the samples from 'sum' query and the 'difference' query, and at least one of these will show inconsistency. We subsequently extend this idea to align all the samples. Once the samples are all aligned, we can just use some any recovery algorithm for compressed sensing to deduce the sparse vectors.

3.3.2 Overview of Our Algorithm

Our scheme to recover the unknown vectors is described below. We will carefully chose the numbers m, m' so that the overall query complexity meets the promise of Theorem 3.6.

- We pick m query vectors $\boldsymbol{x}^1, \boldsymbol{x}^2, \ldots, \boldsymbol{x}^m$ independently, each according to $\mathcal{N}(\mathbf{0}, \boldsymbol{I}_n)$ where $\mathbf{0}$ is the n-dimensional all zero vector and $\boldsymbol{I}_n$ is the $n \times n$ identity matrix.
- (Mixture) We repeatedly query the oracle with $\boldsymbol{x}^i$ for T_i times for all $i \in [m]$ in order to offset the noise. The samples obtained from the repeated querying of $\boldsymbol{x}^i$ is referred to as a *batch* corresponding to $\boldsymbol{x}^i$. T_i is referred to as the *batchsize* of $\boldsymbol{x}^i$. Our objective is to return $\hat{\mu}_{i,1}$ and $\hat{\mu}_{i,2}$, estimates of $\langle \boldsymbol{x}^i, \boldsymbol{\beta}^1 \rangle$ and $\langle \boldsymbol{x}^i, \boldsymbol{\beta}^2 \rangle$ respectively from the batch of samples. However, it will not be possible to label which estimated mean corresponds to $\boldsymbol{\beta}^1$ and which one corresponds to $\boldsymbol{\beta}^2$.
- (Alignment) For some $m' < m$ and for each $i \in [m], j \in [m']$ such that $i \neq j$, we also query the oracle with the vectors $\boldsymbol{x}^i + \boldsymbol{x}^j$ (*sum query*) and $\boldsymbol{x}^i - \boldsymbol{x}^j$ (*difference* query) repeatedly for $T_{i,j}^{\mathsf{sum}}$ and $T_{i,j}^{\mathsf{diff}}$ times respectively. Our objective is to cluster the set of estimated means $\{\hat{\mu}_{i,1}, \hat{\mu}_{i,2}\}_{i=1}^m$ into two equally sized clusters such that all the elements in a particular cluster are good estimates of querying the same unknown vector.
- Since the queries $\{\boldsymbol{x}^i\}_{i=1}^m$ has the property of being a good compressed sensing matrix (they satisfy δ-RIP condition, a sufficient condition for ℓ_2-ℓ_1 recovery in compressed sensing, with high probability), we can formulate a convex optimization problem using the estimates present in each cluster to recover the unknown vectors $\boldsymbol{\beta}^1$ and $\boldsymbol{\beta}^2$.

3.3.3 Recovering Unknown Means from a Batch

For a query $\boldsymbol{x} \in \{\boldsymbol{x}^1, \boldsymbol{x}^2, \ldots, \boldsymbol{x}^m\}$, notice that the samples from the batch corresponding to $\boldsymbol{x}$ is distributed according to a Gaussian mixture $\mathcal{M}$,

$$\mathcal{M} \triangleq \frac{1}{2}\mathcal{N}(\langle \boldsymbol{x}, \boldsymbol{\beta}^1 \rangle, \sigma^2) + \frac{1}{2}\mathcal{N}(\langle \boldsymbol{x}, \boldsymbol{\beta}^2 \rangle, \sigma^2),$$

an equally weighted mixture of two Gaussian distributions having means $\langle \boldsymbol{x}, \boldsymbol{\beta}^1 \rangle, \langle \boldsymbol{x}, \boldsymbol{\beta}^2 \rangle$ with known variance σ^2. For brevity, let us denote $\langle \boldsymbol{x}, \boldsymbol{\beta}^1 \rangle$ by μ_1 and $\langle \boldsymbol{x}, \boldsymbol{\beta}^2 \rangle$ by μ_2

from here on in this sub-section. In essence, our objective is to find the sufficient batchsize of $\boldsymbol{x}$ so that it is possible to estimate $\langle \boldsymbol{x}, \boldsymbol{\beta}^1 \rangle$ and $\langle \boldsymbol{x}, \boldsymbol{\beta}^2 \rangle$ upto an additive error of γ. Below, we go over some methods providing theoretical guarantees on the sufficient sample complexity for approximating the means that will be suitable for different parameter regimes.

Algorithm 3.4 EM$(\boldsymbol{x}, \sigma, T)$ Estimate the means $\langle \boldsymbol{x}, \boldsymbol{\beta}^1 \rangle$, $\langle \boldsymbol{x}, \boldsymbol{\beta}^2 \rangle$ for a query $\boldsymbol{x}$ using EM algorithm

---

Require: An oracle $\mathcal{O}$ which when queried with a vector $\boldsymbol{x} \in \mathbb{R}^n$ returns $\langle \boldsymbol{x}, \boldsymbol{\beta} \rangle + \mathcal{N}(0, \sigma^2)$ where $\boldsymbol{\beta}$ is sampled uniformly from $\{\boldsymbol{\beta}^1, \boldsymbol{\beta}^2\}$.

1: **for** $i = 1, 2, \ldots, T$ **do**
2: Query the oracle $\mathcal{O}$ with $\boldsymbol{x}$ and obtain a response y^i.
3: **end for**
4: Set the function $w : \mathbb{R}^3 \to \mathbb{R}$ as $w(y, \mu_1, \mu_2) = e^{-(y-\mu_1)^2/2\sigma^2} \Big(e^{-(y-\mu_1)^2/2\sigma^2} + e^{-(y-\mu_2)^2/2\sigma^2} \Big)^{-1}$.
5: **Initialize** $\hat{\mu}_1^0, \hat{\mu}_2^0$ randomly and $t = 0$.
6: **while** Until Convergence **do**
7: $\hat{\mu}_1^{t+1} = \sum_{i=1}^T y_i w(y_i, \hat{\mu}_1^t, \hat{\mu}_2^t) / \sum_{i=1}^T w(y_i, \hat{\mu}_1^t, \hat{\mu}_2^t)$.
8: $\hat{\mu}_2^{t+1} = \sum_{i=1}^T y_i w(y_i, \hat{\mu}_2^t, \hat{\mu}_1^t) / \sum_{i=1}^T w(y_i, \hat{\mu}_2^t, \hat{\mu}_1^t)$.
9: $t \leftarrow t + 1$.
10: **end while**
11: Return $\hat{\mu}_1^t, \hat{\mu}_2^t$

Recovery using EM algorithm The Expectation Maximization (EM) algorithm is widely known, and used for the purpose of parameter learning of Gaussian mixtures, cf. [13] and [150]. The EM algorithm tailored towards recovering the parameters of the mixture $\mathcal{M}$ is described in Algorithm 3.4. The following result can be derived from [41] (with our terminology) that gives a sample complexity guarantee of using EM algorithm.

Theorem 3.7 (Finite sample EM analysis [41])**.** *From an equally weighted two component Gaussian mixture with unknown component means μ_1, μ_2 and known and shared*

variance σ^2*, a total* $O\Big(\Big\lceil \sigma^6/(\epsilon^2(\mu_1-\mu_2)^4)\log 1/\eta\Big\rceil\Big)$ *samples suffice to return* $\hat{\mu}_1, \hat{\mu}_2$*, such that for some permutation* $\pi : \{1,2\} \to \{1,2\}$*, for* $i = 1, 2$*,*

$$\left|\hat{\mu}_i - \mu_{\pi(i)}\right| \leq \epsilon$$

using the EM algorithm with probability at least $1 - \eta$*.*

This theorem implies that EM algorithm requires smaller number of samples as the separation between the means $|\mu_1 - \mu_2|$ grows larger. However, it is possible to have a better dependence on $|\mu_1 - \mu_2|$, especially when it is small compared to σ^2.

Method of Moments Consider any Gaussian mixture with two components,

$$\mathcal{G} \triangleq p_1\mathcal{N}(\mu_1, \sigma_1^2) + p_2\mathcal{N}(\mu_2, \sigma_2^2),$$

where $0 < p_1, p_2 < 1$ and $p_1 + p_2 = 1$. Define the variance of a random variable distributed according to $\mathcal{G}$ to be

$$\sigma_{\mathcal{G}}^2 \triangleq p_1p_2((\mu_1 - \mu_2)^2 + p_1\sigma_1^2 + p_2\sigma_2^2.$$

It was shown in [72] that $\Theta(\sigma_{\mathcal{G}}^{12}/\epsilon^{12})$ samples are both necessary and sufficient to recover the unknown parameters $\mu_1, \mu_2, \sigma_1^2, \sigma_2^2$ upto an additive error of ϵ. However, in our setting the components of the mixture $\mathcal{M}$ have the same known variance σ^2 and further the mixture is equally weighted. Our first contribution is to show significantly better results for this special case.

Theorem 3.8. *With* $O\Big(\Big\lceil \sigma_{\mathcal{M}}^2/\epsilon_1^2, \sigma_{\mathcal{M}}^4/\epsilon_2^2\Big\rceil \log 1/\eta\Big)$ *samples, Algorithm 3.6 returns* $\hat{\mu}_1, \hat{\mu}_2$*, such that for some permutation* $\pi : \{1,2\} \to \{1,2\}$*, we have, for* $i = 1, 2$*,* $\left|\hat{\mu}_i - \mu_{\pi(i)}\right| \leq 2\epsilon_1 + 2\sqrt{\epsilon_2}$ *with probability at least* $1 - \eta$*.*

This theorem states that $O(\sigma_{\mathcal{M}}^4)$ samples are sufficient to recover the unknown means of $\mathcal{M}$ (as compared to the $O(\sigma_{\mathcal{G}}^{12})$ result for the general case). This is because the mean and variance are sufficient statistics for this special case (as compared to first six excess moments in the general case). We first show two technical lemmas providing guarantees on recovering the mean and the variance of a random variable X distributed according to $\mathcal{M}$. The procedure to return $\hat{M}_1$ and $\hat{M}_2$ (estimates of $\mathbb{E}X$ and $\text{var}X$ respectively) is described in Algorithm 3.5.

Lemma 3.3. *$O\Big(\Big\lceil \sigma_{\mathcal{M}}^2/\epsilon_1^2 \Big\rceil \log \eta^{-1}\Big)$ samples divided into $B = 36 \log \eta^{-1}$ equally sized batches are sufficient to compute $\hat{M}_1$ (see Algorithm 3.5) such that $\Big|\hat{M}_1 - \mathbb{E}X\Big| \leq \epsilon_1$ with probability at least $1 - 2\eta$.*

Lemma 3.4. *$O\Big(\Big\lceil \sigma_{\mathcal{M}}^4/\epsilon_2^2 \Big\rceil \log \eta^{-1}\Big)$ samples divided into $B = 36 \log \eta^{-1}$ equally sized batches is sufficient to compute $\hat{M}_2$ (see Algorithm 3.5) such that $\Big|\hat{M}_2 - \text{var}X\Big| \leq \epsilon_2$ with probability at least $1 - 2\eta$.*

Algorithm 3.5 ESTIMATE$(\boldsymbol{x}, T, B)$ Estimating $\mathbb{E}X$ and $\text{var}X$ for $X \sim \mathcal{M}$

Require: I.i.d samples $y^1, y^2, \ldots, y^T \sim \mathcal{M}$ where $\mathcal{M} = \frac{1}{2}\mathcal{N}(\langle \boldsymbol{x}, \boldsymbol{\beta}^1 \rangle, \sigma^2) + \frac{1}{2}\mathcal{N}(\langle \boldsymbol{x}, \boldsymbol{\beta}^2 \rangle, \sigma^2)$.

1: Set $t = T/B$
2: **for** $i = 1, 2, \ldots, B$ **do**
3: Set Batch i to be the samples y^j for $j \in \{it + 1, it + 2, \ldots, (i+1)t\}$.
4: Set $S_1^i = \sum_{j \in \text{ Batch } i} \frac{y^i}{t}$, $S_2^i = \sum_{j \in \text{ Batch } i} \frac{(y^i - S_1^i)^2}{t-1}$.
5: **end for**
6: $\hat{M}_1 = \mathsf{median}(\{S_1^i\}_{i=1}^B)$, $\hat{M}_2 = \mathsf{median}(\{S_2^i\}_{i=1}^B)$.
7: Return $\hat{M}_1, \hat{M}_2$.

The detailed proofs of Lemma 3.3 and 3.4 can be found in Chapter B, Section B.4. We are now ready to prove Theorem 3.8.

Proof of Theorem 3.8. We will set up the following system of equations in the variables $\hat{\mu}_1$ and $\hat{\mu}_2$:

$$\hat{\mu}_1 + \hat{\mu}_2 = 2\hat{M}_1 \text{ and } (\hat{\mu}_1 - \hat{\mu}_2)^2 = 4\hat{M}_2 - 4\sigma^2$$

Recall that from Lemma 3.3 and Lemma 3.4, we have computed $\hat{M}_1$ and $\hat{M}_2$ with the following guarantees: $\left|\hat{M}_1 - \mathbb{E}X\right| \leq \epsilon_1$ and $\left|\hat{M}_2 - \mathrm{var}X\right| \leq \epsilon_2$. Therefore, we must have $|\hat{\mu}_1 + \hat{\mu}_2 - \mu_1 - \mu_2| \leq 2\epsilon_1$, $|(\hat{\mu}_1 - \hat{\mu}_2)^2 - (\mu_1 - \mu_2)^2| \leq 4\epsilon_2$. We can factorize the left hand side of the second equation in the following way:

$$|\hat{\mu}_1 - \hat{\mu}_2 - \mu_1 + \mu_2|\,|\hat{\mu}_1 - \hat{\mu}_2 + \mu_1 - \mu_2| \leq 4\epsilon_2.$$

Notice that one of the factors must be less than $2\sqrt{\epsilon_2}$. Without loss of generality, let us assume that $|\hat{\mu}_1 - \hat{\mu}_2 - \mu_1 + \mu_2| \leq 2\sqrt{\epsilon_2}$. This, along with the fact $|\hat{\mu}_1 + \hat{\mu}_2 - \mu_1 - \mu_2| \leq 2\epsilon_1$ implies that (by adding and subtracting) $|\hat{\mu}_i - \mu_i| \leq 2\epsilon_1 + 2\sqrt{\epsilon_2} \quad \forall i = 1, 2.$ □

Algorithm 3.6 METHOD OF MOMENTS$(\boldsymbol{x}, \sigma, T, B)$ Estimate the means $\langle \boldsymbol{x}, \boldsymbol{\beta}^1 \rangle$, $\langle \boldsymbol{x}, \boldsymbol{\beta}^2 \rangle$ for a query $\boldsymbol{x}$

Require: An oracle $\mathcal{O}$ which when queried with a vector $\boldsymbol{x} \in \mathbb{R}^n$ returns $\langle \boldsymbol{x}, \boldsymbol{\beta} \rangle + \mathcal{N}(0, \sigma^2)$ where $\boldsymbol{\beta}$ is sampled uniformly from $\{\boldsymbol{\beta}^1, \boldsymbol{\beta}^2\}$.

1: **for** $i = 1, 2, \ldots, T$ **do**
2: Query the oracle $\mathcal{O}$ with $\boldsymbol{x}$ and obtain a response y^i.
3: **end for**
4: Compute $\hat{M}_1, \hat{M}_2$ (estimates of $\mathbb{E}_{X \sim \mathcal{M}} X, \mathrm{var}_{X \sim \mathcal{M}} X$ respectively) using Algorithm ESTIMATE$(\boldsymbol{x}, T, B)$.
5: Solve for $\hat{\mu}_1, \hat{\mu}_2$ in the system of equations $\hat{\mu}_1 + \hat{\mu}_2 = 2\hat{M}_1,\ (\hat{\mu}_1 - \hat{\mu}_2)^2 = 4\hat{M}_2 - 4\sigma^2$.
6: Return $\hat{\mu}_1, \hat{\mu}_2$.

Fitting a single Gaussian In the situation when both the variance σ^2 of each component in $\mathcal{M}$ and the separation between the means $|\mu_1 - \mu_2|$ are very small, fitting a single Gaussian $\mathcal{N}(\hat{\mu}, \sigma^2)$ to the samples obtained from $\mathcal{M}$ works better than the aforementioned techniques. The procedure to compute $\hat{M}_1$, an estimate of $\mathbb{E}_{X \sim \mathcal{M}} X = (\mu_1 + \mu_2)/2$ is adapted from [41] and is described in Algorithm 3.7. Notice that Algorithm 3.7 is different from the naive procedure (averaging all samples) described in Algorithm 3.5 for estimating the mean of the mixture. The sample

complexity for the naive procedure (see Lemma 3.3) scales with the gap $|\mu_1 - \mu_2|$ even when the variance σ^2 is small which is undesirable. In stead we have the following lemma.

Lemma 3.5 (Lemma 5 in [41]). *With Algorithm 3.7, $O\Big(\Big\lceil \sigma^2 \log \eta^{-1}/\epsilon^2 \Big\rceil\Big)$ samples are sufficient to compute $\hat{M}_1$ such that $\Big|\hat{M}_1 - (\mu_1 + \mu_2)/2\Big| \leq \epsilon$ with probability at least $1 - \eta$.*

In this case, we will return $\hat{M}_1$ to be estimates of both the means μ_1, μ_2.

Algorithm 3.7 FIT A SINGLE GAUSSIAN($\boldsymbol{x}, T$) Estimate the means $\langle \boldsymbol{x}, \boldsymbol{\beta}^1 \rangle$, $\langle \boldsymbol{x}, \boldsymbol{\beta}^2 \rangle$ for a query $\boldsymbol{x}$

Require: An oracle $\mathcal{O}$ which when queried with a vector $\boldsymbol{x} \in \mathbb{R}^n$ returns $\langle \boldsymbol{x}, \boldsymbol{\beta} \rangle + \mathcal{N}(0, \sigma^2)$ where $\boldsymbol{\beta}$ is sampled uniformly from $\{\boldsymbol{\beta}^1, \boldsymbol{\beta}^2\}$.

1: **for** $i = 1, 2, \ldots, T$ **do**
2: Query the oracle $\mathcal{O}$ with $\boldsymbol{x}$ and obtain a response y^i.
3: **end for**
4: Set $\hat{Q}_1$ and $\hat{Q}_3$ to be the first and third quartiles of the samples $y^1, y^2, \ldots, y^t$ respectively.
5: Return $(\hat{Q}_1 + \hat{Q}_3)/2$.

Choosing appropriate methods Among the above three methods to learn mixtures, the appropriate algorithm to apply for each parameter regime is listed below.

Case 1 ($|\mu_1 - \mu_2| = \Omega(\sigma)$): We use the EM algorithm for this regime to recover μ_1, μ_2. Notice that in this regime, by using Theorem 3.7 with $\epsilon = \gamma$, we obtain that $O\Big(\Big\lceil (\sigma^2/\gamma^2) \log 1/\eta \Big\rceil\Big)$ samples are sufficient to recover μ_1, μ_2 up to an additive error of γ with probability at least $1 - \eta$.

Case 2 ($\sigma \geq \gamma, |\mu_1 - \mu_2| = O(\sigma)$): We use the method of moments to recover μ_1, μ_2. In this regime, we must have $\sigma^2_{\mathcal{M}} = O(\sigma^2)$. Therefore, by using Theorem 3.8 with $\epsilon_1 = \gamma/4, \epsilon_2 = \gamma^2/16$, it is evident that $O\Big(\Big\lceil (\sigma/\gamma)^4 \Big\rceil \log 1/\eta\Big)$ samples are sufficient to recover μ_1, μ_2 upto an additive error of γ with probability at least $1 - \eta$.

Case 3 ($\sigma \leq \gamma, |\mu_1 - \mu_2| \leq \gamma$): In this setting, we fit a single Gaussian. Using Theorem 3.5 with $\epsilon = \gamma/2$, we will be able to estimate $(\mu_1 + \mu_2)/2$ up to an additive error of $\gamma/2$ using $O\Big(\Big\lceil (\sigma^2/\gamma^2) \log 1/\eta \Big\rceil\Big)$ samples. This, in turn implies

$$|\mu_i - \hat{M}_1| \leq \frac{|\mu_1 - \mu_2|}{2} + \left|\frac{\mu_1 + \mu_2}{2} - \hat{M}_1\right| \leq \gamma.$$

for $i \in \{1, 2\}$ and therefore both the means μ_1, μ_2 are recovered up to an additive error of γ. Note that these three cases covers all possibilities.

Test for appropriate method Now, we describe a test to infer which parameter regime we are in and therefore which algorithm to use. The final algorithm to recover the means μ_1, μ_2 from $\mathcal{M}$ including the test is described in Algorithm 3.8. We have the following result, the proof of which is delegated to appendix B.5.

Algorithm 3.8 TEST AND ESTIMATE$(\boldsymbol{x}, \sigma, \gamma, \eta)$ Test for the correct parameter regime and apply the parameter estimation algorithm accordingly for a query $\boldsymbol{x}$

Require: An oracle $\mathcal{O}$ which when queried with a vector $\boldsymbol{x} \in \mathbb{R}^n$ returns $\langle \boldsymbol{x}, \boldsymbol{\beta} \rangle + \mathcal{N}(0, \sigma^2)$ where $\boldsymbol{\beta}$ is sampled uniformly from $\{\boldsymbol{\beta}^1, \boldsymbol{\beta}^2\}$.

Set $T = O\Big(\Big\lceil \log \eta^{-1} \Big\rceil\Big)$.
for $i = 1, 2, \ldots, T$ **do**
    Query the oracle $\mathcal{O}$ with $\boldsymbol{x}$ and obtain a response y^i.
end for
Compute $\tilde{\mu}_1, \tilde{\mu}_2$ by running Algorithm METHOD OF MOMENTS $(\boldsymbol{x}, \sigma, T, 72 \log n)$.
if $\sigma > \gamma$ and $|\tilde{\mu}_1 - \tilde{\mu}_2| \leq 15\sigma/32$ **then**
    Compute $\hat{\mu}_1, \hat{\mu}_2$ by running Algorithm METHOD OF MOMENTS $(\boldsymbol{x}, \sigma, O\Big(\Big\lceil (\sigma/\gamma)^4 \Big\rceil \log 1/\eta, 72 \log n\Big)$.
else if $\sigma \leq \gamma$ and $|\tilde{\mu}_1 - \tilde{\mu}_2| \leq 15\gamma/32$ **then**
    Compute $\hat{\mu}_1, \hat{\mu}_2$ by running Algorithm FIT A SINGLE GAUSSIAN $(\boldsymbol{x}, O\Big(\Big\lceil (\sigma^2/\gamma^2) \log 1/\eta \Big\rceil\Big)$.
else
    Compute $\hat{\mu}_1, \hat{\mu}_2$ by running Algorithm EM$(\boldsymbol{x}, \sigma, O\Big(\Big\lceil (\sigma^2/\gamma^2) \log 1/\eta \Big\rceil\Big)$.
end if
Return $\hat{\mu}_1, \hat{\mu}_2$.

Lemma 3.6. *The number of samples required for Algorithm 3.8 to infer the parameter regime correctly with probability at least* $1-\eta$ *is atmost* $O(\log \eta^{-1})$.

The proof of this lemma can be found in Chapter B, Section B.5.

3.3.4 Alignment

For a query $\boldsymbol{x}^i, i \in [m]$, let us introduce the following notations for brevity:

$$\mu_{i,1} := \langle \boldsymbol{x}^i, \boldsymbol{\beta}^1 \rangle \quad \mu_{i,2} := \langle \boldsymbol{x}^i, \boldsymbol{\beta}^2 \rangle.$$

Now, using Algorithm 3.8, we can compute $(\hat{\mu}_{i,1}, \hat{\mu}_{i,2})$ (estimates of $\mu_{i,1}, \mu_{i,2}$) using a batchsize of T_i such that $\left|\hat{\mu}_{i,j} - \mu_{i,\pi_i(j)}\right| \leq \gamma \quad \forall\, i \in [m], j \in \{1,2\}$, where $\pi_i : \{1,2\} \to \{1,2\}$ is a permutation on $\{1,2\}$.

Algorithm 3.9 ALIGN PAIR$(\boldsymbol{x}^i, \boldsymbol{x}^j, \{\hat{\mu}_{s,t}\}_{s=i,j\ t=1,2}, \sigma, \gamma, \eta)$ Align the mean estimates for $\boldsymbol{x}^i$ and $\boldsymbol{x}^j$.

1: Recover $\hat{\mu}_{\mathsf{sum},i}, \hat{\mu}_{\mathsf{sum},j}$ using Algorithm TEST AND ESTIMATE $(\boldsymbol{x}^i + \boldsymbol{x}^j, \sigma, \gamma, \eta)$.
2: Recover $\hat{\mu}_{\mathsf{diff},i}, \hat{\mu}_{\mathsf{diff},j}$ using Algorithm TEST AND ESTIMATE $(\boldsymbol{x}^i - \boldsymbol{x}^j, \sigma, \gamma, \eta)$.
3: **if** $|\hat{\mu}_{\mathsf{sum},i} - \hat{\mu}_{i,p} - \hat{\mu}_{j,q}| \leq 3\gamma$ such that $p, q \in \{1,2\}$ is unique **then**
4: **if** $p == q$ **then** Return TRUE **else** Return FALSE **end if**
5: **else**
6: Find p, q such that $|\hat{\mu}_{\mathsf{diff},i} - \hat{\mu}_{i,p} + \hat{\mu}_{j,q}| \leq 3\gamma$ for $p, q \in \{1,2\}$.
7: **if** $p == q$ **then** Return TRUE **else** Return FALSE **end if**
8: **end if**

The most important step in our process is to separate the estimates of the means according to the generative unknown sparse vectors ($\boldsymbol{\beta}^1$ and $\boldsymbol{\beta}^2$) (i.e., alignment). Formally, we construct two m-dimensional vectors $\boldsymbol{u}$ and $\boldsymbol{v}$ such that, for all $i \in [m]$ the following hold:

- The i^{th} elements of $\boldsymbol{u}$ and $\boldsymbol{v}$, i.e., u_i and v_i, are $\hat{\mu}_{i,1}$ and $\hat{\mu}_{i,2}$ (but may not be respectively).
- Moreover, we must have the u_i and v_i to be good estimates of $\langle \boldsymbol{x}^i, \boldsymbol{\beta}^{\pi_i(1)} \rangle$ and $\langle \boldsymbol{x}^i, \boldsymbol{\beta}^{\pi_i(2)} \rangle$ respectively i.e. $\left|u_i - \langle \boldsymbol{x}^i, \boldsymbol{\beta}^{\pi_i(1)} \rangle\right| \leq 10\gamma$; $\left|v_i - \langle \boldsymbol{x}^i, \boldsymbol{\beta}^{\pi_i(2)} \rangle\right| \leq 10\gamma$ for all $i \in [m]$ where $\pi_i : \{1,2\} \to \{1,2\}$ is some permutation of $\{1,2\}$.

In essence, for the alignment step, we want to find out all permutations $\pi_i, i \in [m]$. First, note that the aforementioned objective is trivial when $|\mu_{i,1} - \mu_{i,2}| \leq 9\gamma$. To see this, suppose π_i is the identity permutation without loss of generality. In that case, we have for $\hat{\mu}_{i,1}$, $|\hat{\mu}_{i,1} - \mu_{i,1}| \leq \gamma$ and $|\hat{\mu}_{i,1} - \mu_{i,2}| \leq |\hat{\mu}_{i,1} - \mu_{i,1}| + |\mu_{i,1} - \mu_{i,2}| \leq 10\gamma$. Similar guarantees also hold for $\hat{\mu}_{i,2}$ and therefore the choice of the i^{th} element of $\boldsymbol{u}, \boldsymbol{\beta}$ is trivial. This conclusion implies that the interesting case is only for those queries $\boldsymbol{x}^i$ when $|\mu_{i,1} - \mu_{i,2}| \geq 9\gamma$. In other words, this objective is equivalent to separate out the permutations $\{\pi_i\}_{i=1}^m$ for $i : |\mu_{i,1} - \mu_{i,2}| \geq 9\gamma$ into two groups such that all the permutations in each group are the same.

Alignment for two queries Consider two queries $\boldsymbol{x}^1, \boldsymbol{x}^2$ such that $|\mu_{i,1} - \mu_{i,2}| \geq 9\gamma$ for $i = 1, 2$. In this section, we will show how we can infer if π_1 is same as π_2. Our strategy is to make two additional batches of queries corresponding to $\boldsymbol{x}^1 + \boldsymbol{x}^2$ and $\boldsymbol{x}^1 - \boldsymbol{x}^2$ (of size $T_{1,2}^{\mathsf{sum}}$ and $T_{1,2}^{\mathsf{diff}}$ respectively) which we shall call the *sum* and *difference* queries. Again, let us introduce the following notations: $\mu_{\mathsf{sum},1} = \langle \boldsymbol{x}^1 + \boldsymbol{x}^2, \boldsymbol{\beta}^1 \rangle \quad \mu_{\mathsf{sum},2} = \langle \boldsymbol{x}^1 + \boldsymbol{x}^2, \boldsymbol{\beta}^2 \rangle$, $\mu_{\mathsf{diff},1} = \langle \boldsymbol{x}^1 - \boldsymbol{x}^2, \boldsymbol{\beta}^1 \rangle \quad \mu_{\mathsf{diff},2} = \langle \boldsymbol{x}^1 - \boldsymbol{x}^2, \boldsymbol{\beta}^2 \rangle$. As before, using Algorithm 3.8, we can compute $(\hat{\mu}_{\mathsf{sum},1}, \hat{\mu}_{\mathsf{sum},2})$ (estimates of $\mu_{\mathsf{sum},1}, \mu_{\mathsf{sum},2}$) and $(\hat{\mu}_{\mathsf{diff},1}, \hat{\mu}_{\mathsf{diff},2})$ (estimates of $\mu_{\mathsf{diff},1}, \mu_{\mathsf{diff},2}$) using a batchsize of $T_{1,2}^{\mathsf{sum}}$ and $T_{1,2}^{\mathsf{diff}}$ for the *sum* and *difference* query respectively such that $\left|\hat{\mu}_{\mathsf{sum},j} - \mu_{\mathsf{sum},\pi_{\mathsf{sum}}(j)}\right| \leq \gamma \quad$ for $j \in \{1, 2\}$ and $\left|\hat{\mu}_{\mathsf{diff},j} - \mu_{\mathsf{diff},\pi_{\mathsf{diff}}(j)}\right| \leq \gamma \quad$ for $j \in \{1, 2\}$ where $\pi_{\mathsf{sum}}, \pi_{\mathsf{diff}} : \{1, 2\} \to \{1, 2\}$ are again unknown permutations of $\{1, 2\}$. We show the following lemma.

Lemma 3.7. *We can infer, using Algorithm 3.9, if π_1 and π_2 are same using the estimates $\hat{\mu}_{\mathsf{sum},i}, \hat{\mu}_{\mathsf{diff},i}$ provided $|\mu_{i,1} - \mu_{i,2}| \geq 9\gamma$, $i = 1, 2$.*

We provide an outline of the proof over here. In Algorithm 3.9, we first choose one value from $\{\hat{\mu}_{\mathsf{sum},1}, \hat{\mu}_{\mathsf{sum},2}\}$ (say z) and we check if we can choose one element (say a) from the set $\{\hat{\mu}_{1,1}, \hat{\mu}_{1,2}\}$ and one element $\{\hat{\mu}_{2,1}, \hat{\mu}_{2,2}\}$ (say b) in exactly one way such that $|z - a - b| \leq 3\gamma$. If that is true, then we infer that the tuple $\{a, b\}$ are estimates

of the same unknown vector and accordingly return if π_1 is same as π_2. If not possible, then we choose one value from $\{\hat{\mu}_{\mathsf{diff},1}, \hat{\mu}_{\mathsf{diff},2}\}$ (say z') and again we check if we can choose one element (say c) from the set $\{\hat{\mu}_{1,1}, \hat{\mu}_{1,2}\}$ and one element from $\{\hat{\mu}_{2,1}, \hat{\mu}_{2,1}\}$ (say d) in exactly one way such that $|z' - c - d| \leq 3\gamma$. If that is true, then we infer that $\{c, d\}$ are estimates of the same unknown vector and accordingly return if π_1 is same as π_2. It can be shown that we will succeed in this step using at least one of the sum or difference queries.

Alignment for all queries We will align the mean estimates for all the queries $\boldsymbol{x}^1, \boldsymbol{x}^2, \ldots, \boldsymbol{x}^m$ by aligning one pair at a time. This routine is summarized in Algorithm 3.10, which works when $\gamma \leq \frac{13\sqrt{2}}{\sqrt{\pi}}\|\boldsymbol{\beta}^1 - \boldsymbol{\beta}^2\|_2 \approx 0.096\|\boldsymbol{\beta}^1 - \boldsymbol{\beta}^2\|_2$. To understand the routine, we start with the following technical lemma:

Lemma 3.8. *Let,* $\gamma \leq \frac{13\sqrt{2}}{\sqrt{\pi}}\|\boldsymbol{\beta}^1 - \boldsymbol{\beta}^2\|_2$. *For* $m' = \left\lceil \log \eta^{-1} / \log \frac{\sqrt{\pi}||\boldsymbol{\beta}^1 - \boldsymbol{\beta}^2||_2}{13\sqrt{2}\gamma} \right\rceil$, *there exists a query* $\boldsymbol{x}^{i^\star}$ *among* $\{\boldsymbol{x}^i\}_{i=1}^{m'}$ *such that* $|\mu_{i^\star,1} - \mu_{i^\star,2}| \geq 13\gamma$ *with probability at least* $1 - \eta$.

Proof of Lemma 3.8. Notice that for a particular query $\boldsymbol{x}^i, i \in [m]$, the difference of the means $\mu_{i,1} - \mu_{i,2}$ is distributed according to

$$\mu_{i,1} - \mu_{i,2} \sim \mathcal{N}(0, ||\boldsymbol{\beta}^1 - \boldsymbol{\beta}^2||_2^2).$$

Therefore, we have

$$\Pr(|\mu_{i,1} - \mu_{i,2}| \leq 13\gamma) = \int_{-13\gamma}^{13\gamma} \frac{e^{-x^2/2||\boldsymbol{\beta}^1 - \boldsymbol{\beta}^2||_2^2}}{\sqrt{2\pi||\boldsymbol{\beta}^1 - \boldsymbol{\beta}^2||_2^2}} dx \leq \frac{13\sqrt{2}\gamma}{\sqrt{\pi}||\boldsymbol{\beta}^1 - \boldsymbol{\beta}^2||_2}. \tag{3.6}$$

Algorithm 3.10 ALIGN ALL($\{\boldsymbol{x}^i\}_{i\in[m]}, \{\hat{\mu}_{s,t}\}_{s\in[m]\ t=1,2}, \sigma, \gamma, \eta$) Align mean estimates for all queries $\{\boldsymbol{x}^i\}_{i=1}^m$.

Initialize: $\boldsymbol{u}, \boldsymbol{\beta}$ to be m-dimensional all zero vector.
Set $m' = \left\lceil \log \eta^{-1} / \log \frac{\sqrt{\pi}||\boldsymbol{\beta}^1 - \boldsymbol{\beta}^2||_2}{13\sqrt{2}\gamma} \right\rceil$
for $i = 1, 2, \ldots, m$ **do**
  for $j = 1, 2, \ldots, m', j \neq i$ **do**
    Run Algorithm ALIGN PAIR ($\boldsymbol{x}^i, \boldsymbol{x}^j$, $\{\hat{\mu}_{s,t}\}_{\substack{s=i,j \\ t=1,2}}, \sigma, \gamma, \eta$) and store the output.
  end for
end for
Identify $\boldsymbol{x}^p$ from $p \in [m']$ such that $|\hat{\mu}_{p,1} - \hat{\mu}_{p,2}| \geq 11\gamma$.
Set $u_p := \boldsymbol{u}[p] = \hat{\mu}_{p,1}$ and $v_p := \boldsymbol{\beta}[p] = \hat{\mu}_{p,2}$
for $i = 1, 2, \ldots, m, i \neq p$ **do**
  if Output of Algorithm 3.9 for $\boldsymbol{x}^i$ and $\boldsymbol{x}^p$ is TRUE **then**
    Set $\boldsymbol{u}[i] = \hat{\mu}_{i,1}$ and $\boldsymbol{\beta}[i] = \hat{\mu}_{i,2}$.
  else
    Set $\boldsymbol{u}[i] = \hat{\mu}_{i,2}$ and $\boldsymbol{\beta}[i] = \hat{\mu}_{i,1}$.
  end if
end for
Return $\boldsymbol{u}, \boldsymbol{\beta}$.

where the upper bound is obtained by using $e^{-x^2/2||\boldsymbol{\beta}^1 - \boldsymbol{\beta}^2||_2^2} \leq 1$. Therefore the probability that for all the m' queries $\{\boldsymbol{x}^i\}_{i=1}^m$, the difference between the means is less than 13γ must be

$$\begin{aligned}
&\Pr(\bigcup_{i=1}^{m'} \mu_{i,1} - \mu_{i,2} \leq 13\gamma) \\
&= \prod_{i=1}^{m'} \Pr(\mu_{i,1} - \mu_{i,2} \leq 13\gamma) \\
&\leq \Big(\frac{13\sqrt{2}\gamma}{\sqrt{\pi}||\boldsymbol{\beta}^1 - \boldsymbol{\beta}^2||_2}\Big)^{m'} \\
&= e^{-m' \log \frac{\sqrt{\pi}||\boldsymbol{\beta}^1 - \boldsymbol{\beta}^2||_2}{13\sqrt{2}\gamma}}
\end{aligned}$$

Therefore for $m' = \left\lceil \log \eta^{-1} / \log \frac{\sqrt{\pi}||\boldsymbol{\beta}^1 - \boldsymbol{\beta}^2||_2}{13\sqrt{2}\gamma} \right\rceil$, we have that $\Pr(\bigcup_{i=1}^{m'} \mu_{i,1} - \mu_{i,2} \leq 13\gamma) \leq \eta$. □

Now, for $i \in [m'], j \in [m]$ such that $i \neq j$, we will align $\boldsymbol{x}^i$ and $\boldsymbol{x}^j$ using Algorithm 3.9 and according to Lemma 3.7, this alignment procedure will succeed for all such pairs

where $|\mu_{i,1} - \mu_{i,2}|, |\mu_{j,1} - \mu_{j,2}| \geq 9\gamma$ with probability at least $1 - mm'\eta$ (using a union bound). Note that according to Lemma 3.8, there must exist a query $\boldsymbol{x}^{i^\star} \in \{\boldsymbol{x}^i\}_{i=1}^{m'}$ for which $|\mu_{i^\star,1} - \mu_{i^\star,2}| \geq 13\gamma$. This implies that for some $i^\star \in [m']$, we must have $|\hat{\mu}_{i^\star,1} - \hat{\mu}_{i^\star,2}| \geq |\mu_{i^\star,1} - \mu_{i^\star,2}| - |\hat{\mu}_{i^\star,1} - \mu_{i^\star,1}| - |\hat{\mu}_{i^\star,2} - \mu_{i^\star,2}| \geq 11\gamma$. Therefore, we can identify at least one query $\boldsymbol{x}^{\tilde{i}}$ for $\tilde{i} \in [m']$ such that $|\hat{\mu}_{\tilde{i},1} - \hat{\mu}_{\tilde{i},2}| \geq 11\gamma$. However, this implies that $|\mu_{\tilde{i},1} - \mu_{\tilde{i},2}| \geq |\hat{\mu}_{\tilde{i},1} - \hat{\mu}_{\tilde{i},2}| - |\mu_{\tilde{i},1} - \hat{\mu}_{\tilde{i},1}| - |\mu_{\tilde{i},2} - \hat{\mu}_{\tilde{i},2}| \geq 9\gamma$. Therefore we will be able to infer for every query $\boldsymbol{x}^i, i \in [m]$ for which $|\mu_{i,1} - \mu_{i,2}| \geq 9\gamma$ if π_i is same as $\pi_{\tilde{i}}$. Now, we are ready to put everything together and provide the proof for the main result (Thm. 3.6).

3.3.5 Proof of Theorem 3.6

Case 1($\gamma < \left|\left|\boldsymbol{\beta}^1 - \boldsymbol{\beta}^2\right|\right|_2/2$): The overall recovery procedure is described as Algorithm 3.11. Since this algorithm crucially uses Algorithm 3.10, it works only when $\gamma \leq 0.096 \left|\left|\boldsymbol{\beta}^1 - \boldsymbol{\beta}^2\right|\right|_2$; so assume that to hold for now. We will start by showing that for any two Gaussian queries, the samples are far enough (a simple instance of Gaussian anti-concentration).

Algorithm 3.11 RECOVER UNKNOWN VECTORS(σ, γ) Recover the unknown vectors $\boldsymbol{\beta}^1$ and $\boldsymbol{\beta}^2$

1: Set $m = c_s k \log n$.
2: Sample $\boldsymbol{x}^1, \boldsymbol{x}^2, \ldots, \boldsymbol{x}^m \sim \mathcal{N}(\boldsymbol{0}, \boldsymbol{I}_n)$ independently.
3: **for** $i = 1, 2, \ldots, m$ **do**
4: Compute $\hat{\mu}_{i,1}, \hat{\mu}_{i,2}$ by running Algorithm TEST AND ESTIMATE$(\boldsymbol{x}^i, \sigma, \gamma, n^{-2})$.
5: **end for**
6: Construct $\boldsymbol{u}, \boldsymbol{\beta}$ by running Algorithm ALIGN ALL$(\{\boldsymbol{x}^i\}_{i\in[m]}, \{\hat{\mu}_{s,t}\}_{\substack{s\in[m]\\t=1,2}}, \sigma, \gamma, \eta)$.
7: Set $\boldsymbol{A}$ to be the $m \times n$ matrix such that its i^{th} row is $\boldsymbol{x}^i$, with each entry normalized by $\sqrt{m}$.
8: Set $\hat{\boldsymbol{\beta}}^1$ to be the solution of the optimization problem $\min_{\boldsymbol{z}\in\mathbb{R}^n} ||\boldsymbol{z}||_1$ s.t. $||\boldsymbol{A}\boldsymbol{z} - \frac{1}{\sqrt{m}}\boldsymbol{u}||_2 \leq 10\gamma$
9: Set $\hat{\boldsymbol{\beta}}^2$ to be the solution of the optimization problem $\min_{\boldsymbol{z}\in\mathbb{R}^n} ||\boldsymbol{z}||_1$ s.t. $||\boldsymbol{A}\boldsymbol{z} - \frac{1}{\sqrt{m}}\boldsymbol{\beta}||_2 \leq 10\gamma$
10: Return $\hat{\boldsymbol{\beta}}^1, \hat{\boldsymbol{\beta}}^2$.

Lemma 3.9. *For all queries $\boldsymbol{x}$ designed in Algorithm 3.11, for any constant $c_1 > 0$, and some c_2 which is a function of c_1,*

$$\Pr(\big|\langle \boldsymbol{x}, \boldsymbol{\beta}^1\rangle - \langle \boldsymbol{x}, \boldsymbol{\beta}^2\rangle\big| \leq c_1\sigma)) \leq \frac{c_2\sigma}{||\boldsymbol{\beta}^1 - \boldsymbol{\beta}^2||_2}.$$

Proof of Lemma 3.9. The proof of Lemma 3.9 is very similar to the proof of Lemma 3.8. Again for a particular query $\boldsymbol{x}^i, i \in [m]$, the difference of the means $\mu_{i,1} - \mu_{i,2}$ is distributed according to

$$\mu_{i,1} - \mu_{i,2} \sim \mathcal{N}(0, ||\boldsymbol{\beta}^1 - \boldsymbol{\beta}^2||_2^2).$$

Therefore, we have for any constant $c_1 > 0$,

$$\Pr(|\mu_{i,1} - \mu_{i,2}| \leq c_1\sigma) = \int_{-c_1\sigma}^{c_1\sigma} \frac{e^{-x^2/2||\boldsymbol{\beta}^1 - \boldsymbol{\beta}^2||_2^2}}{\sqrt{2\pi||\boldsymbol{\beta}^1 - \boldsymbol{\beta}^2||_2^2}} dx \leq \frac{\sqrt{2}c_1\sigma}{\sqrt{\pi}||\boldsymbol{\beta}^1 - \boldsymbol{\beta}^2||_2}.$$

where the upper bound is obtained by using $e^{-x^2/2||\boldsymbol{\beta}^1 - \boldsymbol{\beta}^2||_2^2} \leq 1$. Hence the lemma is proved by substituting $c_2 = \sqrt{2}c_1/\sqrt{\pi}$. □

Now the theorem is proved via a series of claims.

Claim 3.1. *The expected batchsize for any query designed in Algorithm 3.11 is* $O\Big(\Big\lceil \frac{\sigma^5}{\gamma^4||\boldsymbol{\beta}^1 - \boldsymbol{\beta}^2||_2} + \frac{\sigma^2}{\gamma^2}\Big\rceil \log \eta^{-1}\Big)$.

Proof. In Algorithm 3.11, we make m batches of queries corresponding to $\{\boldsymbol{x}^i\}_{i=1}^m$ and mm' batches of queries corresponding to $\{\boldsymbol{x}^i + \boldsymbol{x}^j\}_{i=1,j=1,i\neq j}^{i=m,j=m'}$ and $\{\boldsymbol{x}^i - \boldsymbol{x}^j\}_{i=1,j=1,i\neq j}^{i=m,j=m'}$. Recall that the batchsize corresponding to $\boldsymbol{x}^i, \boldsymbol{x}^i + \boldsymbol{x}^j, \boldsymbol{x}^i - \boldsymbol{x}^j$ is denoted by T_i,$T_{i,j}^{\mathsf{sum}}$ and $T_{i,j}^{\mathsf{diff}}$ respectively. Recall from Section 3.3.3, we will use method of moments or or fit a single Gaussian (Case 2 and 3 in Section 3.3.3) to estimate the means when the difference between the means is $O(\sigma)$. By Lemma 3.9, this happens with probability

$O(\sigma/||\boldsymbol{\beta}^1-\boldsymbol{\beta}^2||_2)$. Otherwise we will use the EM algorithm (Case 1 in Section 3.3.3). or fit a single gaussian, both of which require a batchsize of at most $O\Big(\Big\lceil \sigma^2/\gamma^2 \Big\rceil \log \eta^{-1}\Big)$. We can conclude that the expected size of any of the aforementioned batchsize is bounded from above as the following: $\mathbb{E}T \leq O\Big(\Big\lceil \frac{\sigma^5}{\gamma^4||\boldsymbol{\beta}^1-\boldsymbol{\beta}^2||_2} + \frac{\sigma^2}{\gamma^2} \Big\rceil \log \eta^{-1}\Big)$ where $T \in \{T_i\} \cup \{T_{i,j}^{\mathsf{sum}}\} \cup \{T_{i,j}^{\mathsf{diff}}\}$ so that we can recover all the mean estimates upto an an additive error of γ with probability at least $1 - O(mm'\eta)$. □

Claim 3.2. *Algorithm 3.10 returns two vectors $\boldsymbol{u}$ and $\boldsymbol{v}$ of length m each such that*

$$\Big|\boldsymbol{u}[i] - \langle \boldsymbol{x}^i, \boldsymbol{\beta}^{\pi(1)} \rangle\Big| \leq 10\gamma; \Big|\boldsymbol{\beta}[i] - \langle \boldsymbol{x}^i, \boldsymbol{\beta}^{\pi(2)} \rangle\Big| \leq 10\gamma$$

for some permutation $\pi : \{1,2\} \to \{1,2\}$ for all $i \in [m]$ with probability at least $1 - \eta$.

The proof of this claim directly follows from the discussion in Section 3.3.4.

The matrix $\boldsymbol{A}$ is size $m \times n$ whose i^{th} row is the query vector $\boldsymbol{x}^i$ normalized by $\sqrt{m}$. .

Claim 3.3. *We must have*

$$||\boldsymbol{A}\boldsymbol{\beta}^{L(1)} - \frac{\boldsymbol{u}}{\sqrt{m}}||_2 \leq 10\gamma \;\&\; ||\boldsymbol{A}\boldsymbol{\beta}^{L(2)} - \frac{\boldsymbol{\beta}}{\sqrt{m}}||_2 \leq 10\gamma.$$

Proof. The proof of this claim follows from the fact that after normalization by $\sqrt{m}$, the error in each entry is also normalized by $\sqrt{m}$ and is therefore at most $10\gamma/\sqrt{m}$. Hence the ℓ_2 difference is at most 10γ. □

It is known that for $m \geq c_s k \log n$ where $c_s > 0$ is some appropriate constant, the matrix $\boldsymbol{A}$ satisfy *restricted isometric property* of order $2k$, which means for any exactly $2k$-sparse vector $\boldsymbol{x}$ and a constant δ, we have $|\|\boldsymbol{A}\boldsymbol{x}\|_2^2 - \|\boldsymbol{x}\|_2^2| \leq \delta\|\boldsymbol{x}\|_2^2$ cf.[15].

We now solve the following convex optimization problems, standard recovery method called basis pursuit:

$$\hat{\boldsymbol{\beta}}^{\pi(1)} = \min_{\boldsymbol{z} \in \mathbb{R}^n} ||\boldsymbol{z}||_1 \text{ s.t. } ||\boldsymbol{A}\boldsymbol{z} - \frac{\boldsymbol{u}}{\sqrt{m}}||_2 \leq 10\gamma$$

$$\hat{\boldsymbol{\beta}}^{\pi(2)} = \min_{\boldsymbol{z}\in\mathbb{R}^n} ||\boldsymbol{z}||_1 \text{ s.t. } ||\boldsymbol{A}\boldsymbol{z} - \frac{\boldsymbol{\beta}}{\sqrt{m}}||_2 \leq 10\gamma$$

to recover $\hat{\boldsymbol{\beta}}^{\pi(1)}, \hat{\boldsymbol{\beta}}^{\pi(2)}$, estimates of $\boldsymbol{\beta}^1, \boldsymbol{\beta}^2$ having the guarantees given in Theorem 3.6 (see, Thm. 1.6 in [22]). The expected query complexity is $O\Big(mm' \log \eta^{-1} \Big\lceil \frac{\sigma^5}{\gamma^4 ||\boldsymbol{\beta}^1 - \boldsymbol{\beta}^2||_2} + \frac{\sigma^2}{\gamma^2} \Big\rceil\Big)$ Substituting $m = O(k \log n)$, $m' = O\Big(\Big\lceil \frac{\log \eta^{-1}}{\log \frac{||\boldsymbol{\beta}^1 - \boldsymbol{\beta}^2||_2}{\gamma}} \Big\rceil\Big)$ and $\eta = (mm' \log n)^{-1}$, we obtain the total query complexity

$$O\Bigg(k \log n \log k \Big\lceil \frac{\log k}{\log(\|\boldsymbol{\beta}^1 - \boldsymbol{\beta}^2\|_2/\gamma)} \Big\rceil \times \Big\lceil \frac{\sigma^5}{\gamma^4 ||\boldsymbol{\beta}^1 - \boldsymbol{\beta}^2||_2} + \frac{\sigma^2}{\gamma^2} \Big\rceil \Bigg)$$

and the error probability to be $o(1)$. We can just substitute the definition of NF and notice that $\mathsf{SNR} = \|\boldsymbol{\beta}^1 - \boldsymbol{\beta}^2\|_2^2/\sigma^2$ to obtain the query complexity promised in Theorem 3.6. Note that, we have assumed $k = \Omega(\log n)$ above.

It remains to be proved that the same (orderwise) number of samples is sufficient to recover both unknown vectors with high probability. For each query $\boldsymbol{x}$ designed in Algorithm 3.11, consider the indicator random variable $Y_i = \mathbb{1}[|\mu_{i,1} - \mu_{i,2}| = \Omega(\sigma)]$. The total number of queries for which this event is true (given by $\sum_i Y_i$) is sampled according to the binomial distribution $\mathsf{Bin}(mm', O(\sigma/||\boldsymbol{\beta}^1 - \boldsymbol{\beta}^2||_2))$ and therefore concentrates tightly around its mean. A simple use of Chernoff bound leads to the desired result.

While we have proved the theorem for any $\gamma \leq 0.096 \left|\left|\boldsymbol{\beta}^1 - \boldsymbol{\beta}^2\right|\right|_2$, it indeed holds for any $\gamma = c' \left|\left|\boldsymbol{\beta}^1 - \boldsymbol{\beta}^2\right|\right|_2$, where c' is a constant strictly less than 1. If the desired $\gamma > 0.096 \left|\left|\boldsymbol{\beta}^1 - \boldsymbol{\beta}^2\right|\right|_2$, then one can just define $\gamma' = 0.096 \left|\left|\boldsymbol{\beta}^1 - \boldsymbol{\beta}^2\right|\right|_2$ and obtain a precision γ' which is a constant factor within γ. Since the quantity NF defined with γ' is also within a constant factor of the original NF, the sample complexity can also change by at most a constant factor.

Algorithm 3.12 RECOVER UNKNOWN VECTORS $2(\sigma, \gamma)$ Recover the unknown vectors $\boldsymbol{\beta}^1$ and $\boldsymbol{\beta}^2$

1: Set $m = c_s k \log n$ and $T = O\Big(\Big\lceil \sigma^2 \log k/(\gamma - 0.8 \left|\left|\boldsymbol{\beta}^1 - \boldsymbol{\beta}^2\right|\right|_2)^2 \Big\rceil\Big)$.
2: Sample $\boldsymbol{x}^1, \boldsymbol{x}^2, \ldots, \boldsymbol{x}^m \sim \mathcal{N}(\boldsymbol{0}, \boldsymbol{I}_n)$ independently.
3: **for** $i = 1, 2, \ldots, m$ **do**
4: Compute $\hat{\mu}_i$ by running Algorithm FIT A SINGLE GAUSSIAN $(\boldsymbol{x}^i, T)$.
5: **end for**
6: Set $\boldsymbol{u}$ to be the m-dimensional vector whose i^{th} element is $\hat{\mu}_i$.
7: Set $\boldsymbol{A}$ to be the $m \times n$ matrix such that its i^{th} row is $\boldsymbol{x}^i$, with each entry normalized by $\sqrt{m}$.
8: Set $\hat{\boldsymbol{\beta}}$ to be the solution of the optimization problem $\min_{\boldsymbol{z} \in \mathbb{R}^n} ||\boldsymbol{z}||_1$ s.t. $||\boldsymbol{A}\boldsymbol{z} - \frac{1}{\sqrt{m}}\boldsymbol{u}||_2 \leq \gamma$
9: Return $\hat{\boldsymbol{\beta}}$.

Case 2 ($\gamma = \Omega(\left|\left|\boldsymbol{\beta}^1 - \boldsymbol{\beta}^2\right|\right|_2)$): We will first assume $\gamma > 0.8 \left|\left|\boldsymbol{\beta}^1 - \boldsymbol{\beta}^2\right|\right|_2$ to prove the claim, and later extend this to any $\gamma = \Omega(\left|\left|\boldsymbol{\beta}^1 - \boldsymbol{\beta}^2\right|\right|_2)$. The recovery procedure in the setting when $\gamma > 0.8 \left|\left|\boldsymbol{\beta}^1 - \boldsymbol{\beta}^2\right|\right|_2$ is described in Algorithm 3.12. We will start by proving the following claim

Claim 3.4. *Algorithm 3.12 returns a vector $\boldsymbol{u}$ of length m using $O\Big(m\Big\lceil \sigma^2 \log \eta^{-1}/\epsilon^2 \Big\rceil\Big)$ queries such that*

$$\left|\boldsymbol{u}[i] - \langle \boldsymbol{x}^i, \boldsymbol{\beta}^1 \rangle\right| \leq \epsilon + \frac{\left|\langle \boldsymbol{x}^i, \boldsymbol{\beta}^1 - \boldsymbol{\beta}^2 \rangle\right|}{2}$$
$$\left|\boldsymbol{u}[i] - \langle \boldsymbol{x}^i, \boldsymbol{\beta}^2 \rangle\right| \leq \epsilon + \frac{\left|\langle \boldsymbol{x}^i, \boldsymbol{\beta}^1 - \boldsymbol{\beta}^2 \rangle\right|}{2}$$

for all $i \in [m]$ with probability at least $1 - m\eta$.

Proof. In Algorithm 3.12, for each query $\boldsymbol{x}^i \sim \mathcal{N}(\boldsymbol{0}, \boldsymbol{I}_n)$, we can use a batchsize of $O\Big(\Big\lceil \sigma^2 \log \eta^{-1}/\epsilon^2 \Big\rceil\Big)$ to recover $\hat{\mu}_i$ such that

$$\left|\hat{\mu}_i - \frac{\langle \boldsymbol{x}^i, \boldsymbol{\beta}^1 \rangle + \langle \boldsymbol{x}^i, \boldsymbol{\beta}^2 \rangle}{2}\right| \leq \epsilon$$

with probability at least $1-\eta$ according to the guarantees of Lemma 3.5. We therefore have

$$\left|\hat{\mu}_i - \langle \boldsymbol{x}^i, \boldsymbol{\beta}^1 \rangle\right| \leq \left|\hat{\mu}^i - \frac{\langle \boldsymbol{x}^i, \boldsymbol{\beta}^1 \rangle + \langle \boldsymbol{x}^i, \boldsymbol{\beta}^2 \rangle}{2}\right| + \frac{\left|\langle \boldsymbol{x}^i, \boldsymbol{\beta}^1 - \boldsymbol{\beta}^2 \rangle\right|}{2} \leq \epsilon + \frac{\left|\langle \boldsymbol{x}^i, \boldsymbol{\beta}^1 - \boldsymbol{\beta}^2 \rangle\right|}{2}.$$

where the last inequality follows by using the guarantees on $\hat{\mu}_i$. We can show a similar chain of inequalities for $\left|\hat{\mu}_i - \langle \boldsymbol{x}^i, \boldsymbol{\beta}^2 \rangle\right|$ and finally take a union bound over all $i \in [m]$ to conclude the proof of the claim. □

Next, let us define the random variable $\omega_i \triangleq \left|\langle \boldsymbol{x}^i, \boldsymbol{\beta}^1 - \boldsymbol{\beta}^2 \rangle\right|$ where the randomness is over $\boldsymbol{x}^i$. Subsequently let us define the m-dimensional vector $\boldsymbol{b}$ whose i^{th} element is $\epsilon + \omega_i/2$. Again, for $m \geq c_s k \log n$, let $\boldsymbol{A}$ denote the matrix whose i^{th} row is $\boldsymbol{x}^i$ normalized by $\sqrt{m}$.

Claim 3.5. *We must have*

$$\left|\left|\boldsymbol{A}\boldsymbol{\beta}^1 - \frac{\boldsymbol{u}}{\sqrt{m}}\right|\right|_2 \leq \frac{||\boldsymbol{b}||_2}{\sqrt{m}} \;\&\; \left|\left|\boldsymbol{A}\boldsymbol{\beta}^2 - \frac{\boldsymbol{u}}{\sqrt{m}}\right|\right|_2 \leq \frac{||\boldsymbol{b}||_2}{\sqrt{m}}.$$

Proof. The proof of the claim is immediate from definition of $\boldsymbol{A}$ and $\boldsymbol{b}$. □

Next, we show high probability bounds on ℓ_2-norm of the vector $\boldsymbol{b}$ in the following claim.

Claim 3.6. *We must have* $\frac{||\boldsymbol{b}||_2^2}{m} \leq 2\epsilon^2 + 0.64\left|\left|\boldsymbol{\beta}^1 - \boldsymbol{\beta}^2\right|\right|_2^2$ *with probability at least* $1 - O(e^{-m})$.

Proof. Notice that

$$\frac{||\boldsymbol{b}||_2^2}{m} \leq \frac{1}{m}\sum_{i=1}^{m}\left(\epsilon + \frac{\omega_i}{2}\right)^2 \leq \frac{2}{m}\sum_{i=1}^{m}\left(\epsilon^2 + \frac{\omega_i^2}{4}\right).$$

Using the fact that $\boldsymbol{x}^i \sim \mathcal{N}(\mathbf{0}, \boldsymbol{I}_n)$ and by definition, we must have that ω_i is a random variable distributed according to $\mathcal{N}(0, \left|\left|\boldsymbol{\beta}^1 - \boldsymbol{\beta}^2\right|\right|_2)$. Therefore, we have (see Lemma 1.2 [22])

$$\Pr\left(\left|\sum_{i=1}^{m} \omega_i^2 - m\left|\left|\boldsymbol{\beta}^1 - \boldsymbol{\beta}^2\right|\right|_2^2\right| \geq m\rho\left|\left|\boldsymbol{\beta}^1 - \boldsymbol{\beta}^2\right|\right|_2^2\right)$$
$$\leq 2\exp\Big(-\frac{m}{2}\Big(\frac{\rho^2}{2} - \frac{\rho^3}{3}\Big)\Big)$$

for $0 < \rho < 1$. Therefore, by substituting $\rho = 0.28$, we get that

$$\frac{2}{m}\sum_{i=1}^{m}\Big(\epsilon^2 + \frac{\omega_i^2}{4}\Big) \leq 2\epsilon^2 + 0.64\left|\left|\boldsymbol{\beta}^1 - \boldsymbol{\beta}^2\right|\right|_2^2$$

with probability at least $1 - O(e^{-m})$. □

From Claim 3.6, we get

$$\frac{||\boldsymbol{b}||_2}{\sqrt{m}} \leq \sqrt{2}\epsilon + 0.8\left|\left|\boldsymbol{\beta}^1 - \boldsymbol{\beta}^2\right|\right|_2.$$

where we use the inequality $\sqrt{a+b} \leq \sqrt{a} + \sqrt{b}$ for $a, b > 0$. Subsequently we solve the following convex optimization problem

$$\min_{\boldsymbol{z}\in\mathbb{R}^n} ||\boldsymbol{z}||_1 \text{ s.t. } ||\boldsymbol{A}\boldsymbol{z} - \frac{\boldsymbol{u}}{\sqrt{m}}||_2 \leq \gamma$$

where $\gamma = \sqrt{2}\epsilon + 0.8\left|\left|\boldsymbol{\beta}^1 - \boldsymbol{\beta}^2\right|\right|_2$ in order to recover $\hat{\beta}$ and return it as estimate of both $\boldsymbol{\beta}^1, \boldsymbol{\beta}^2$. For $m = O(k\log n), \eta = (m\log n)^{-1}$ and $\sqrt{2}\epsilon = \gamma - 0.8\left|\left|\boldsymbol{\beta}^1 - \boldsymbol{\beta}^2\right|\right|_2$, the number of queries required is $O\Big(k\log n\Big\lceil\sigma^2\log k/(\gamma - 0.8\left|\left|\boldsymbol{\beta}^1 - \boldsymbol{\beta}^2\right|\right|_2)^2\Big\rceil\Big)$. Further, by using the theoretical guarantees provided in Theorem 1.6 in [22], we obtain the guarantees of the main theorem with error probability atmost $o(1)$. Again, by

substituting the definition of the Noise Factor $\mathsf{NF} = \gamma/\sigma$ and the Signal to Noise ratio $\mathsf{SNR} = O(\left|\left|\boldsymbol{\beta}^1 - \boldsymbol{\beta}^2\right|\right|_2^2/\sigma^2)$, we obtain the query complexity to be

$$O\Big(k \log n \Big\lceil \frac{\log k}{(\mathsf{NF} - 0.8\sqrt{\mathsf{SNR}})^2} \Big\rceil\Big).$$

Now let us assume any $\gamma = \Omega(\left|\left|\boldsymbol{\beta}^1 - \boldsymbol{\beta}^2\right|\right|_2)$. If the desired $\gamma < \left|\left|\boldsymbol{\beta}^1 - \boldsymbol{\beta}^2\right|\right|_2$, then one can just define $\gamma' = \left|\left|\boldsymbol{\beta}^1 - \boldsymbol{\beta}^2\right|\right|_2$ and obtain a precision γ' which is a constant factor within γ. Further, the query complexity also becomes independent of the noise factor since $\mathsf{NF} = \sqrt{\mathsf{SNR}}$ for this choice of γ' and thus we obtain the promised query complexity in Theorem 3.6.

3.3.6 Discussion on Noiseless Setting ($\sigma = 0$)

Step 1: In the noiseless setting, we obtain $m = O(k \log n)$ query vectors $\boldsymbol{x}^1, \ldots, \boldsymbol{x}^m$ sampled i.i.d according to $\mathcal{N}(\mathbf{0}, \boldsymbol{I}_n)$ and repeat each of them for $2 \log m$ times. For a particular query $\boldsymbol{x}_i$, the probability that we do not obtain any samples from $\boldsymbol{\beta}^1$ or $\boldsymbol{\beta}^2$ is at most $(1/2)^{2 \log m}$. We can take a union bound to conclude that for all queries, we obtain samples from both $\boldsymbol{\beta}^1$ and $\boldsymbol{\beta}^2$ with probability at least $1 - O(m^{-1})$. Further note that for each query $\boldsymbol{x}^i$, $\langle \boldsymbol{x}^i, \boldsymbol{\beta}^1 - \boldsymbol{\beta}^2 \rangle$ is distributed according to $\mathcal{N}(0, \left|\left|\boldsymbol{\beta}^1 - \boldsymbol{\beta}^2\right|\right|_2^2)$ and therefore, it must happen with probability 1 that $\langle \boldsymbol{x}^i, \boldsymbol{\beta}^1 \rangle \neq \langle \boldsymbol{x}^i, \boldsymbol{\beta}^1 \rangle$. Thus for each query $\boldsymbol{x}_i$, we can recover the tuple $(\langle \boldsymbol{x}^i, \boldsymbol{\beta}^1 \rangle, \langle \boldsymbol{x}^i, \boldsymbol{\beta}^2 \rangle)$ but we cannot recover the ordering i.e. we do not know which element of the tuple corresponds to $\boldsymbol{\beta}^1$ and which one to $\boldsymbol{\beta}^2$.

Step 2: Note that we are still left with the alignment step where we need to cluster the $2m$ recovered parameters $\{(\langle \boldsymbol{x}^i, \boldsymbol{\beta}^1 \rangle, \langle \boldsymbol{x}^i, \boldsymbol{\beta}^2 \rangle)\}_{i=1}^m$ into two clusters of size m each so that there exists exactly one element from each tuple in each of the two clusters and all the elements in the same cluster correspond to the same unknown vector. In order to complete this step, we query $\boldsymbol{x}_1 + \boldsymbol{x}_i$ and $\boldsymbol{x}_1 - \boldsymbol{x}_i$ for all $i \neq 1$ each for

$2 \log m$ times to the oracle and recover the tuples $(\langle \boldsymbol{x}^1 + \boldsymbol{x}^i, \boldsymbol{\beta}^1 \rangle, \langle \boldsymbol{x}^1 + \boldsymbol{x}^i, \boldsymbol{\beta}^2 \rangle)$ and $(\langle \boldsymbol{x}^1 - \boldsymbol{x}^i, \boldsymbol{\beta}^1 \rangle, \langle \boldsymbol{x}^1 - \boldsymbol{x}^i, \boldsymbol{\beta}^2 \rangle)$ for all $i \neq 1$. For a particular $i \in [m] \setminus \{1\}$, we will choose two elements (say a and b) from the pairs $(\langle \mathbf{x}_1, \boldsymbol{\beta}^1 \rangle, \langle \mathbf{x}_1, \boldsymbol{\beta}^2 \rangle)$ and $(\langle \mathbf{x}_i, \boldsymbol{\beta}^1 \rangle, \langle \mathbf{x}_i, \boldsymbol{\beta}^2 \rangle)$ (one element from each pair) such that their sum belongs to the pair $\langle \mathbf{x}_1 + \mathbf{x}_i, \boldsymbol{\beta}^1 \rangle, \langle \mathbf{x}_1 + \mathbf{x}_i, \boldsymbol{\beta}^2 \rangle$ and their difference belongs to the pair $\langle \mathbf{x}_1 - \mathbf{x}_i, \boldsymbol{\beta}^1 \rangle, \langle \mathbf{x}_1 - \mathbf{x}_i, \boldsymbol{\beta}^2 \rangle$. In our algorithm, we will put a, b into the same cluster and the other two elements into the other cluster. From construction, we must put $(\langle \mathbf{x}_1, \boldsymbol{\beta}^1 \rangle, \langle \mathbf{x}_i, \boldsymbol{\beta}^1 \rangle)$ in one cluster and $(\langle \mathbf{x}_1, \boldsymbol{\beta}^2 \rangle, \langle \mathbf{x}_i, \boldsymbol{\beta}^2 \rangle)$ in other. Note that a failure in this step is not possible because the $2m$ recovered parameters are different from each other with probability 1.

Step 3: Once we have clustered the samples, we have reduced our problem to the usual compressed sensing setting (with only 1 unknown vector) and therefore we can run the well known convex optimization routine in order to recover the unknown vectors $\boldsymbol{\beta}^1$ and $\boldsymbol{\beta}^2$. The total query complexity is $O(k \log n \log k)$.

3.3.7 'Proof of Concept' Simulations

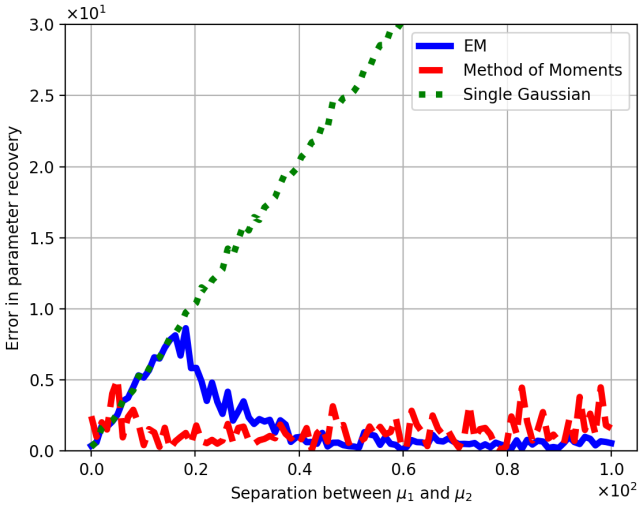
The methods of parameter recovery in Gaussian mixtures are compared in Fig 3.1a. As claimed in Sec. 3.3.3, the EM starts performing better than the method of moments when the gap between the parameters is large.

We have also run Algorithm 3.11 for different set of pairs of sparse vectors and example recovery results for visualization are shown in Figures 3.1b and 3.1c. Note that, while quite accurate reconstruction is possible the vectors are not reconstructed in order, as to be expected.

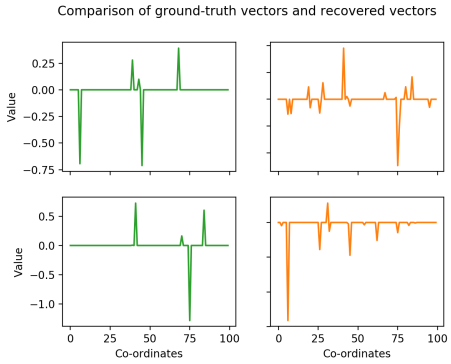
3.4 Support Recovery

3.4.1 Our Techniques and Results

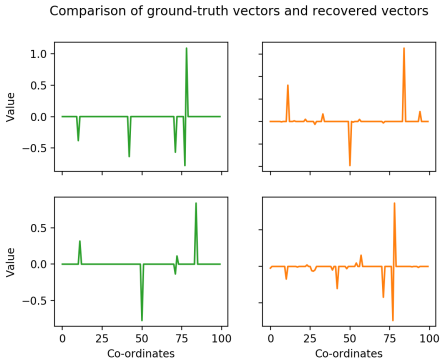
Tensor Decomposition: Consider a tensor $\mathcal{A}$ of order $w \in \mathbb{N}, w > 2$ on $\mathbb{R}^n$ which is denoted by $\mathcal{A} \in \mathbb{R}^n \otimes \mathbb{R}^n \otimes \cdots \otimes \mathbb{R}^n$ (w times). Let us denote by $\mathcal{A}_{i_1, i_2, \ldots, i_w}$ where



(a) Comparison of the three techniques for recovery of parameters of a Gaussian mixture with 1000 samples (see Algorithms 3.4,3.6 and 3.7). The error in parameter recovery is plotted with separation between μ_1 and μ_2 (by keeping μ_1 fixed at 0 and varying μ_2).



(b) The 100-dimensional ground truth vectors β^1 and β^2 with sparsity $k = 5$ plotted in green (left) and the recovered vectors (using Algorithm 3.11) $\hat{\beta}^1$ and $\hat{\beta}^2$ plotted in orange (right) using a batch-size ~ 100 for each of 150 random gaussian queries. The order of the recovered vectors and the ground truth vectors is reversed.



(c) The 100-dimensional ground truth vectors β^1 and β^2 with sparsity $k = 5$ plotted in green (left) and the recovered vectors (using Algorithm 3.11) $\hat{\beta}^1$ and $\hat{\beta}^2$ plotted in orange (right) using a batch-size ~ 600 for each of 150 random gaussian queries. The order of the recovered vectors and the ground truth vectors is reversed.

Figure 3.1: Simulation results of our techniques.

$i_1, i_2, \ldots, i_w \in [n]$, the element in $\mathcal{A}$ whose location along the j^{th} dimension is i_j i.e. there are $i_j - 1$ elements along the j^{th} dimension before $\mathcal{A}_{i_1,i_2,\ldots,i_w}$. Notice that this indexing protocol uniquely determines the element within the tensor. For a detailed review of tensors, we defer the reader to [93]. In this work, we are interested in low rank decomposition of tensors. A tensor $\mathcal{A}$ can be described as a rank-1 symmetric tensor if it can be expressed as

$$\mathcal{A} = \underbrace{\boldsymbol{z} \otimes \boldsymbol{z} \otimes \cdots \otimes \boldsymbol{z}}_{w \quad \text{times}}$$

for some $\boldsymbol{z} \in \mathbb{R}^n$ i.e. $\mathcal{A}_{i_1,i_2,\ldots,i_w} = \prod_{j=1}^{w} \boldsymbol{z}_{i_j}$. A tensor $\mathcal{A}$ that can be expressed as a sum of R rank-1 symmetric tensors is defined as a rank R symmetric tensor. For such a rank R tensor $\mathcal{A}$ provided as input, we are concerned with the problem of unique decomposition of $\mathcal{A}$ into a sum of R rank-1 symmetric tensors; such a decomposition is also known as a Canonical Polyadic (CP) decomposition. Below, we show a result due to [128] describing the sufficient conditions (Kruskal's result) for the unique CP decomposition of a rank R tensor $\mathcal{A}$:

Lemma 3.10 (Unique CP decomposition [128])**.** *Suppose $\mathcal{A}$ is the sum of R rank-one tensors i.e.*

$$\mathcal{A} = \sum_{r=1}^{R} \underbrace{\boldsymbol{z}^r \otimes \boldsymbol{z}^r \otimes \cdots \otimes \boldsymbol{z}^r}_{w \quad \textit{times}}.$$

and further, the Kruskal Rank of the $n \times R$ matrix whose columns are formed by $\boldsymbol{z}^1, \boldsymbol{z}^2, \ldots, \boldsymbol{z}^R$ is J. Then, if $wJ \geq 2R + (w-1)$, then the CP decomposition is unique and we can recover the vectors $\boldsymbol{z}^1, \boldsymbol{z}^2, \ldots, \boldsymbol{z}^R$ up to permutations.

There exist many different techniques for CP decomposition of a tensor but the most well-studied ones are Jennrich's Algorithm (see Section 3.3, [112]) and the Alternating Least Squares (ALS) algorithm [93]. Among these, Jennrich's algorithm (see Section D.2 for more details) is efficient and recovers the latent rank-1 tensors uniquely

but it works only for tensors of order 3 when the underlying vectors $\boldsymbol{z}^1, \boldsymbol{z}^2, \ldots, \boldsymbol{z}^R$ are linearly independent (See Theorem 3.3.2, [112]); this is a stronger condition than what we obtain from Lemma 3.10 for $w = 3$. On the other hand, the ALS algorithm is an iterative algorithm which is easy to implement for tensors of any order but unfortunately, it takes many iterations to converge and furthermore, it is not guaranteed to converge to the correct solution. Jennrich's algorithm also has the additional advantage that it will throw an error if its sufficient condition for unique CP decomposition is not satisfied. This property will turn out to be useful for the problem that we study in this work. Finally, notice that if $\mathcal{A}$ is the weighted sum of R rank-1 tensors i.e.,

$$\mathcal{A} = \sum_{r=1}^{R} \lambda_r \underbrace{\boldsymbol{z}^r \otimes \boldsymbol{z}^r \otimes \cdots \otimes \boldsymbol{z}^r}_{w \quad \text{times}}.$$

then we can rewrite $\mathcal{A} = \sum_{r=1}^{R} \boldsymbol{y}^r \otimes \boldsymbol{y}^r \otimes \cdots \otimes \boldsymbol{y}^r$ where $\boldsymbol{y}^r = \lambda_r^{1/w} \boldsymbol{z}^r$. If $\{\boldsymbol{y}^r\}_{r=1}^R$ satisfies the conditions of Lemma 3.10 and if it is known that $\{\boldsymbol{z}^r\}_{r=1}^R$ are binary vectors, then we can still recover $\boldsymbol{z}^r$ by first computing $\boldsymbol{y}^r$ and then taking its support for all $r \in [R]$. Subsequently, notice that we can also recover $\{\lambda_r\}_{r=1}^R$. As we discussed, for tensors of order $w > 3$, there is no known efficient algorithm that can recover the correct solution even if its existence and uniqueness is known. Due to this limitation, it was necessary in prior works using low rank decomposition of tensors that the unknown parameter vectors are linearly independent [32, 6] since tensors of order > 3 could not be used. However, if it is known apriori that the vectors $\{\boldsymbol{z}^r\}_{r=1}^R$ are binary and the coefficients $\{\lambda_r\}_{r=1}^R$ are positive integers bounded from above by some $C > 0$, then we can exhaustively search over all possibilities ($O(C2^n)$ of them) to find the unique decomposition even in higher order tensors. The set of possible solutions can be reduced significantly if the unknown vectors are known to be sparse as is true in our setting.

Family of sets: We now review literature on some important families of sets called *union free families* [60] and *cover free families* [90] that found applications in cryptography, group testing and 1-bit compressed sensing. These special families of sets are used crucially in this work.

Definition 3.1 (Robust Union Free Family (d,t,α)- RUFF [1])**.** *Let d,t be integers and $0 \leq \alpha \leq 1$. A family of sets $\mathcal{F} = \{\mathcal{H}_1, \mathcal{H}_2, \ldots, \mathcal{H}_n\}$ with each $\mathcal{H}_i \subseteq [m]$ and $|\mathcal{H}| = d$ is a (d,t,α)-RUFF if for any set of t indices $T \subset [n], |T| = t$, and any index $j \notin T$, $\left|\mathcal{H}_j \setminus \left(\bigcup_{i\in T} \mathcal{H}_i\right)\right| > (1-\alpha)d$.*

We refer to n as the size of the family of sets, and m to be the alphabet over which the sets are defined. RUFFs were studied earlier in the context of support recovery of 1bCS [1], and a simple randomized construction of (d,t,α)-RUFF with $m = O(t^2 \log n)$ was proposed by De Wolf [48].

Lemma 3.11. *[1, 48] Given n, t and $\alpha > 0$, there exists an (d,t,α)-RUFF, $\mathcal{F}$ with $m = O\big((t^2 \log n)/\alpha^2\big)$ and $d = O((t \log n)/\alpha)$.*

RUFF is a generalization of the family of sets known as the Union Free Familes (UFF) - which are essentially $(d,t,1)$-RUFF. We require yet another generalization of UFF known as Cover Free Families (CFF) that are also sometimes referred to as superimposed codes [59].

Definition 3.2 (Cover Free Family (r,t)-CFF)**.** *A family of sets $\mathcal{F} = \{\mathcal{H}_1, \mathcal{H}_2, \ldots, \mathcal{H}_n\}$ where each $\mathcal{H}_i \subseteq [m]$ is an (r,t)-CFF if for any pair of disjoint sets of indices $T_1, T_2 \subset [n]$ such that $|T_1| = r, |T_2| = t, T_1 \cap T_2 = \emptyset$, $\left|\bigcap_{i\in T_1} \mathcal{H}_i \setminus \bigcup_{i\in T_2} \mathcal{H}_i\right| > 0$.*

Several constructions and bounds on existence of CFFs are known in literature. We state the following lemma regarding the existence of CFF which can be found in [123, 66].

Lemma 3.12. *For any given integers r, t, there exists an (r,t)-CFF, $\mathcal{F}$ of size n with $m = O(t^{r+1} \log n)$.*

Proof. We give a non-constructive proof for the existence of $(r,t) - \mathsf{CFF}$ of size n and alphabet $m = O(t^{r+1} \log n)$. Recall that a family of sets $\mathcal{F} = \{\mathcal{H}_1, \mathcal{H}_2, \ldots, \mathcal{H}_n\}$ where each $\mathcal{H}_i \subseteq [m]$ is an $(r,t) - \mathsf{CFF}$ if the following holds: for all distinct $j_0, j_1, \ldots, j_{t+r-1} \in [n]$, it is the case that

$$\bigcap_{p \in \{0,1,\ldots,r-1\}} \mathcal{H}_{j_p} \not\subseteq \bigcup_{q \in \{r,r+1,\ldots,t+r-1\}} \mathcal{H}_{j_q}.$$

Since PUFF is a special case of $(r,t) - \mathsf{CFF}$ for $r = 2$, this result holds for PUFF as well.

Consider a matrix $\boldsymbol{G}$ of size $m \times n$ where each entry is generated independently from a Bernoulli(p) distribution with p as a parameter. Consider a distinct set of $t+r$ indices $j_0, j_1, \ldots, j_{t+r-1} \in [n]$. For a particular row of the matrix $\boldsymbol{G}$, the event that there exists a $\mathtt{1}$ in the indices $j_0, j_1, \ldots, j_{r-1}$ and $\mathtt{0}$ in the indices $j_r, j_{r+1}, \ldots, j_{t+r-1}$ holds with probability $p^r(1-p)^t$. Therefore, for a fixed row, this event does not hold with probability $1 - p^r(1-p)^t$ and the probability that for all the rows the event does not hold is $(1 - p^r(1-p)^t)^m$. Notice that the number of such possible sets of $t+r$ columns is $\binom{n}{t+r}\binom{t+r}{r}$. By taking a union bound, the probability (P_e) that the event does not hold for all the rows for at least one set of $t+r$ indices is

$$P_e \leq \binom{n}{t+r}\binom{t+r}{r}\left(1 - p^r(1-p)^t\right)^m$$

Since we want to minimize the upper bound, we want to maximize $p^r(1-p)^t$. Substituting $p = \frac{1}{t+1}$, we get that

$$p^r(1-p)^t = \left(\frac{t}{t+1}\right)^t \cdot \frac{1}{(t+1)^r} > \frac{1}{e(t+1)^r}.$$

Further, using the fact that $\binom{n}{t} \leq \left(\frac{en}{t}\right)^t$, we obtain

$$P_e \leq \frac{(en)^{t+r}}{(t+r)^t}\Big(1 - \frac{1}{e(t+1)^r}\Big)^m \leq \frac{(en)^{t+r}}{(t+r)^t}\exp\Big(-\frac{m}{e(t+1)^r}\Big) < \alpha$$

for some very small number η. Taking log on both sides and after some rearrangement, we obtain

$$m > e(t+1)^r\Big((t+r)\log\frac{en}{t+r} + r\log(t+r) + \log\frac{1}{\eta}\Big).$$

Hence, using $m = O(t^{r+1}\log n)$, the event holds for at least one row for every set of $t+r$ indices with high probability. Therefore, with high probability, the family of sets $\mathcal{F} = \{\mathcal{H}_1, \mathcal{H}_2, \ldots, \mathcal{H}_n\}$ corresponding to the rows of $\boldsymbol{G}$ is a $(r,t) - \mathsf{CFF}$. □

Recall that the set of unknown vectors is denoted by $\mathcal{B} \equiv \{\boldsymbol{\beta}^1, \boldsymbol{\beta}^2, \ldots, \boldsymbol{\beta}^L\}$. Let $\mathbf{A} \in \{0,1\}^{n\times L}$ denote the support matrix corresponding to $\mathcal{B}$ where each column vector $\boldsymbol{A}_i \in \{0,1\}^n$ represents the support of the i^{th} unknown vector $\boldsymbol{\beta}^i$. For any ordered tuple $C \subset [n]$ of indices, and any binary string $\mathbf{a} \in \{0,1\}^{|C|}$, define $\mathsf{occ}(C, \mathbf{a})$ to be the set of unknown vectors whose corresponding supports have the substring $\mathbf{a}$ at positions indexed by C, i.e.,

$$\mathsf{occ}(C, \boldsymbol{a}) := \{\boldsymbol{\beta}^i \in \mathcal{B} \mid \operatorname{supp}\boldsymbol{\beta}^i|_C = \mathbf{a}\}.$$

In order to describe our techniques and our results, we need to introduce three different properties of matrices and extend them to a set of vectors by using their corresponding support matrix. The proofs of our main results follow from the guarantees of algorithms (Algorithm 3.13, Algorithm 3.14 and Algorithm 3.15) each of which leverage the aforementioned key properties of the unknown support matrix $\boldsymbol{A}$. While explaining the intuition behind the introduced matrix properties, we will assume that

all the unknown vectors in $\mathcal{B}$ have distinct supports to keep things simple. However, this assumption is not necessary and all our algorithms work even when the supports are not distinct.

Definition 3.3 (p-identifiable)**.** *The i^{th} column $\boldsymbol{A}_i$ of a binary matrix $\boldsymbol{A} \in \{0,1\}^{n \times L}$ with all distinct columns is called p-identifiable if there exists a set $S \subset [n]$ of at most p-indices and a binary string $\boldsymbol{a} \in \{0,1\}^p$ such that $\boldsymbol{A}_i|_S = \boldsymbol{a}$, and $\boldsymbol{A}_j|_S \neq \boldsymbol{a}$ for all $j \neq i$.*

A binary matrix $\boldsymbol{A} \in \{0,1\}^{n \times L}$ with all distinct columns is called p-identifiable if there exists a permutation $\sigma : [L] \to [L]$ such that for all $i \in [L]$, the sub-matrix $\boldsymbol{A}^i$ formed by deleting the columns indexed by the set $\{\sigma(1), \sigma(2), \ldots, \sigma(i-1)\}$ has at least one p-identifiable column.

Let $\mathcal{B}$ be set of L unknown vectors in $\mathbb{R}^n$, and $\boldsymbol{A} \in \{0,1\}^{n \times L}$ be its support matrix. Let $\boldsymbol{B}$ be the matrix obtained by deleting duplicate columns of $\boldsymbol{A}$. The set $\mathcal{B}$ is called p-identifiable if $\boldsymbol{B}$ is p-identifiable.

Support matrix $\boldsymbol{A}$ is p-identifiable: Algorithm 3.13 uses the property that the support matrix $\boldsymbol{A}$ is p-identifiable for some known $p \leq \log L$ (See Theorem 3.10) to recover the supports of all the unknown vectors. First, we briefly describe the support recovery algorithm where we assume that the support of each unknown vector contains a unique identifying index, i.e., for each unknown vector $\boldsymbol{\beta} \in \mathcal{B}$, there exists a unique index $i \in [n]$ such that $\mathsf{occ}((i),1) = \{\boldsymbol{\beta}\}$, and hence $|\mathsf{occ}((i),1)| = 1$. Observe that if $|\mathsf{occ}((i),1)| = 1$, and $|\mathsf{occ}((i,j),(1,1))| = 1$ for some $i \neq j$, then it follows that both the indices i, j belong to the support of the same unknown vector. Therefore, we are able to recover the supports of all the unknown vectors by computing $|\mathsf{occ}((i),1)|$ and $|\mathsf{occ}((i,j),(1,1))|$ for all $i, j \in [n]$. Hence, the crux of this algorithm lies in computing all the n values of $|\mathsf{occ}((i),1)|$, and $O(n^2)$ values of $|\mathsf{occ}((i,j),(1,1))|$ using just $\mathsf{poly}(L,k)$ queries. We can generalize this aforementioned support recovery

technique by observing that if $\boldsymbol{A}$ is p-identifiable, then there exists an unknown vector $\boldsymbol{\beta} \in \mathcal{B}$ with a distinct support, a unique set $C \subseteq [n]$ and string $\boldsymbol{a} \in \{0,1\}^{|C|}$ satisfying $|C| \leq p$ such that $\mathsf{occ}(C \cup \{j\}, (\boldsymbol{a}, 1)) = \{\boldsymbol{\beta}\}$. By a similar observation as before, if $|\mathsf{occ}(C \cup \{j\}, (\boldsymbol{a}, 1))| = 1$ for some $j \in [n] \setminus C$, we can certify that $j \in \mathrm{supp}(\boldsymbol{\beta})$. The main technical challenge then lies in computing all the $O(2^p n^p)$ values $|\mathsf{occ}(C, \boldsymbol{a})|$ for every p and, $p+1$-sized ordered tuples of indices and all $\boldsymbol{a} \in \{0,1\}^p \cup \{0,1\}^{p+1}$ (Lemma 3.13) using few queries.

Definition 3.4 (flip-independent)**.** *A binary matrix $\boldsymbol{A}$ with all distinct columns is called flip-independent if there exists a subset of rows that if complemented (changing `0` to `1` and `1` to `0`) make all columns of $\boldsymbol{A}$ linearly independent.*

Let $\mathcal{B}$ be a set of L unknown vectors in $\mathbb{R}^n$, and $\boldsymbol{A} \in \{0,1\}^{n \times L}$ be its support matrix. Let $\boldsymbol{B}$ be the matrix obtained by deleting duplicate columns of $\boldsymbol{A}$. The set $\mathcal{B}$ has flip-independent supports if $\mathbf{B}$ is flip-independent.

Support matrix is flip-independent: Algorithm 3.14 uses the property that the support matrix $\boldsymbol{A}$ is flip-independent in order to recover the supports of the unknown vectors uniquely. As a pre-processing step, we identify the set $\mathcal{U} \triangleq \cup_{i \in [L]} \mathsf{supp}(\boldsymbol{\beta}^i)$ that represents the union of support of the unknown vectors. Let us define $\mathcal{U}' \triangleq \mathcal{U} \cup \{t\}$ where t is any index that does not belong to $\mathcal{U}$. This initial pre-processing step allows us to significantly reduce the computational complexity of this algorithm. Next, for each $\boldsymbol{a} \in \{0,1\}^3$, Algorithm 3.14 recovers $|\mathsf{occ}((i_1, i_2, i_3), \boldsymbol{a})|$ for every ordered tuple $(i_1, i_2, i_3) \in \mathcal{U}^3$. Using these recovered quantities, it is possible to construct the tensors

$$\mathcal{A}^{\mathcal{F}} = \sum_{i \in [L]} f(\mathcal{F}, \mathsf{supp}(\boldsymbol{\beta}^i)) \otimes f(\mathcal{F}, \mathsf{supp}(\boldsymbol{\beta}^i)) \otimes f(\mathcal{F}, \mathsf{supp}(\boldsymbol{\beta}^i))$$

for every subset $\mathcal{F} \subseteq \mathcal{U}'$. Since the matrix $\boldsymbol{A}$ is flip-independent, we know that there exists at least one subset $\mathcal{F}^\star \subseteq \mathcal{U}'$ such that the vectors $\{f(\mathcal{F}^\star, \mathsf{supp}(\boldsymbol{\beta}^i))\}_{i=1}^L$ are linearly independent. From Lemma 3.10, we know that by a CP decomposition of $\mathcal{A}^{\mathcal{F}^\star}$,

we can uniquely recover the vectors $\{f(\mathcal{F}^\star, \mathsf{supp}(\boldsymbol{\beta}^i))\}_{i=1}^L$; since the set $\mathcal{F}^\star$ is known, we can recover all the vectors $\{\mathsf{supp}(\boldsymbol{\beta}^i))\}_{i=1}^L$ by flipping all indices corresponding to $\mathcal{F}^\star$. However, a remaining challenge is to correctly identify a set $\mathcal{F}^\star$. Interestingly, Jennrich's algorithm (see Algorithm D.1 in Appendix D.2) throws an error if the tensor $\mathcal{A}^{\mathcal{F}}$ under consideration does not satisfy the uniqueness conditions for CP decomposition i.e. the underlying unknown vectors $\{f(\mathcal{F}, \mathsf{supp}(\boldsymbol{\beta}^i))\}_{i=1}^L$ are not linearly independent. Therefore Algorithm 3.14 is guaranteed to uniquely recover the supports of the unknown vectors.

Definition 3.5 (Kruskal rank)**.** *The Kruskal rank of a matrix $\boldsymbol{A}$ is defined as the maximum number r such that any r columns of $\boldsymbol{A}$ are linearly independent.*

Definition 3.6 (r-Kruskal rank support)**.** *Let $\mathcal{B}$ be a set of L unknown vectors in $\mathbb{R}^n$, and $\boldsymbol{A} \in \{0,1\}^{n \times L}$ be its support matrix. Let $\boldsymbol{B}$ be the matrix obtained by deleting duplicate columns of $\boldsymbol{A}$. The set $\mathcal{B}$ has r-Kruskal rank support if $\mathbf{B}$ has Kruskal rank r.*

Support matrix has Kruskal rank r: Algorithm 3.15 partially generalizes the flip-independence property by constructing higher order tensors. Again, as a pre-processing step, we identify the set $\mathcal{U} \triangleq \cup_{i \in [L]} \mathsf{supp}(\boldsymbol{\beta}^i)$ that represents the union of support of the unknown vectors. Note that $|\mathcal{U}| \leq Lk$ since each unknown vector is k-sparse. Since Jennrich's algorithm is not applicable for tensors of order more than 3, we will simply search over all $O((Lk)^{Lk})$ possibilities in order to compute the unique CP decomposition of an input tensor. Unfortunately though, if the sufficiency conditions (Lemma 3.10) for unique CP decomposition is not met, there can be multiple solutions and we will not be able to detect the correct one. This is the reason why it is not possible to completely generalize Algorithm 3.14 by constructing multiple tensors of higher order. To circumvent this issue, Algorithm 3.15 constructs only a single tensor $\mathcal{A}$ of rank L and order $w = \lceil \frac{2L-1}{r-1} \rceil$ by setting its $(i_1, \ldots, i_w)$-th

entry to $|\mathsf{occ}((i_1, \ldots, i_w), \mathbf{1}_w)|$ for every ordered tuple $(i_1, \ldots, i_w) \in [n]^w$. By using Theorem 3.10, the recovery of the supports of the unknown vectors via brute force CP decomposition of the constructed tensor is unique if the support matrix has Kruskal rank r.

All the above described algorithms require Lemma 3.13 that for any $s > 1$ computes $|\mathsf{occ}(C, \boldsymbol{a})|$ for every s-sized ordered tuple of indices C, and any $\boldsymbol{a} \in \{0, 1\}^s$ using few label queries. The key technical ingredient in Lemma 3.13 is to estimate $\mathsf{nzcount}(\boldsymbol{x})$ - the number of unknown vectors that have a non-zero inner product with $\boldsymbol{x}$. Note that even this simple task is non-trivial in the mixture model and more so with noisy label queries.

All the query complexity results below are valid with

$$\mathsf{SNR} = O(L^2 \max_{i \in [L]} \left|\left|\boldsymbol{\beta}^i\right|\right|_2^2 / \delta^2)$$

where δ is the minimum magnitude of any non-zero entry of any unknown vector in $\mathcal{B}$. In our first result, we recover the support of the unknown vectors with small number of label queries provided the support matrix of $\mathcal{B}$ is p-identifiable.

Theorem 3.9. *Let $\mathcal{B}$ be a set of L unknown vectors in $\mathbb{R}^n$ such that $\mathcal{B}$ is p-indentifiable. Then, Algorithm 3.13 recovers the support of all the unknown vectors in $\mathcal{B}$ with probability at least $1 - O(1/n)$ using $O(L^3(Lk)^{p+2} \log(Lkn) \log n)$ queries.*

In fact, all binary matrices with distinct columns are p-identifiable for some sufficiently large p.

Theorem 3.10. *Any $n \times L$, (with $n > L$) binary matrix with all distinct columns is p-identifiable for some $p \leq \log L$.*

Proof. Suppose $\boldsymbol{A}$ is the said matrix. Since all the columns of $\boldsymbol{A}$ are distinct, there must exist an index $i \in [n]$ which is not the same for all columns in $\boldsymbol{A}$. We must have

$|\mathsf{occ}((i), a)| \leq L/2$ for some $a \in \{0, 1\}$. Subsequently, we consider the columns of $\boldsymbol{A}$ indexed by the set $\mathsf{occ}((i), a)$ and can repeat the same step. Evidently, there must exist an index $j \in [n]$ such that $|\mathsf{occ}((i), \boldsymbol{a})| \leq L/4$ for some $\boldsymbol{a} \in \{0, 1\}^2$. Clearly, we can repeat this step at most $\log L$ times to find $C \subset [n]$ and $\boldsymbol{a} \in \{0, 1\}^{\leq \log L}$ such that $|\mathsf{occ}(C, \boldsymbol{a})| = 1$ and therefore the column in $\mathsf{occ}(C, \boldsymbol{a})$ is p-identifiable. We denote the index of this column as $\sigma(1)$ and form the sub-matrix $\boldsymbol{A}^1$ by deleting the column. Again, $\boldsymbol{A}^1$ has $L-1$ distinct columns and by repeating similar steps, $\boldsymbol{A}^1$ has a column that is $\log(L-1)$ identifiable. More generally, $\boldsymbol{A}^i$ formed by deleting the columns indexed in the set $\{\sigma(1), \sigma(2), \ldots, \sigma(i-1)\}$, has a column that is $\log(L-i)$ identifiable with the index (in $\boldsymbol{A}$) of the column having the unique sub-string (in $\boldsymbol{A}^i$) denoted by $\sigma(i)$. Thus the lemma is proved. □

Thus, we have the following corollary characterizing the unconditional worst-case guarantees for support recovery:

Corollary 3.1. *Let $\mathcal{B}$ be a set of L unknown vectors in $\mathbb{R}^n$. Then, Algorithm 3.13 recovers the support of all the unknown vectors in $\mathcal{B}$ with probability at least $1 - O(1/n)$ using $O(L^3(Lk)^{\log L+2} \log(Lkn) \log n)$ queries.*

Proof. The proof follows from the fact that any set $\mathcal{B}$ of L unknown vectors in $\mathbb{R}^n$ must have p-identifiable supports for $p \leq \log L$. □

Under some assumptions on the unknown support, e.g.flip-independence, we have better results.

Theorem 3.11. *Let $\mathcal{B}$ be a set of L unknown vectors in $\mathbb{R}^n$ such that $\mathcal{B}$ is flip-independent. Then, Algorithm 3.14 recovers the support of all the unknown vectors in $\mathcal{B}$ with probability at least $1 - O(1/n)$ using $O(L^3(Lk)^4 \log(Lkn) \log n)$ queries.*

We can also leverage the property of small Kruskal rank of the support matrix to show:

Theorem 3.12. *Let $\mathcal{B}$ be a set of L unknown vectors in $\mathbb{R}^n$ that has r-Kruskal rank support with $r \geq 2$. Let $w = \lceil \frac{2L-1}{r-1} \rceil$. Then, Algorithm 3.15 recovers the support of all the unknown vectors in $\mathcal{B}$ with probability at least $1 - O(1/n)$ using $O(L^3(Lk)^{w+1} \log(Lkn) \log n)$ queries.*

Discussion on Matrix Properties: Note that p-identifiability (Definition 3.3) property allows us to recover the supports of all the unknown vectors *in the worst-case without any assumptions* (Corollary 3.1). The flip-independence (Definition 3.4) and r-Kruskal rank support (Definition 3.6) properties are used for the tensor-decomposition based support recovery algorithms and follow naturally from Lemma 3.10. The flip-independence assumption is quite weak, however there do exist binary matrices that are not flip independent. For example

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

is not flip independent. The r-Kruskal rank support condition generalizes linear independence conditions considered in previous mixture model studies such as [152]. Note that this condition is always satisfied by the support vectors for some $r \geq 2$. *Essentially, we 1) provide algorithms for support recovery without any assumptions, 2) and also provide significantly better guarantees under extremely mild assumptions that we conjecture to be always true.*

Although we have not optimized the run-time of our algorithms in this work, we report the relevant computational complexities below:

Remark 3.3 (Computational Complexity)**.** *Note that Algorithm 3.13 has a computational complexity that is polynomial in the sparsity k, dimension n and scales as $O(L^p)$*

where $p \leq \log L$. On the other hand Algorithms 3.14, 3.15 has a computational complexity that scales exponentially with k, L while remaining polynomial in the dimension n. For the special case when the support matrix is known to be full rank, Algorithm 3.15 with $w = 3$ is polynomial in all parameters (by using Algorithm D.1 for the CP decomposition.)

3.4.2 Detailed Proofs and Algorithms

Recall the definition of $\mathsf{occ}(C, \boldsymbol{a})$ - the set of unknown vectors having $\boldsymbol{a} \in \{0,1\}^{|C|}$ as a substring in coordinates $C \subset [n]$. First, we observe that for any set $\mathcal{T} \subseteq \{0,1\}^s$, we can compute $|\mathsf{occ}(C, \boldsymbol{a})|$ for all $O(n^s)$ subsets of s indices $C \subset [n]$ and $\boldsymbol{a} \in \mathcal{T}$ using queries.

Lemma 3.13. *Let $\mathcal{T} \subseteq \{0,1\}^s$ be any set of binary vectors of length s. There exists an algorithm to compute $|\mathsf{occ}(C, \boldsymbol{a})|$ for all $C \subset [n]$ of size s, and all $\boldsymbol{a} \in \mathcal{T}$ with probability at least $1 - 1/n$ using*

$$O(L^3(Lk)^{s+1} \log(Lkn) \log n)$$

queries.

Lemma 3.13 (proved in Section 3.4.3) is an integral and non-trivial component of the proofs of all our main Theorems. In each of the sub-sections below, we go through each of them.

Recovery of p-identifiable support matrix In this section we present an algorithm for support recovery of all the L unknown vectors $\mathcal{B} \equiv \{\boldsymbol{\beta}^1, \ldots, \boldsymbol{\beta}^L\}$ when $\mathcal{B}$ is p-identifiable. In particular, we show that if $\mathcal{B}$ is p-identifiable, then computing $|\mathsf{occ}(C, \boldsymbol{a})|$ for every subset of p and $p+1$ indices is sufficient to recover the supports.

Proof of Theorem 3.9. The proof follows from the observation that for any subset of p indices $C \subset [n]$, index $j \in [n] \setminus C$ and $\boldsymbol{a} \in \{0,1\}^p$, $|\mathsf{occ}(C, \boldsymbol{a})| = |\mathsf{occ}(C \cup \{j\}, (\boldsymbol{a}, 1))| +$

$|\mathsf{occ}(C \cup \{j\}, (\boldsymbol{a}, 0))|$. Therefore if one of the terms in the RHS is 0 for all $j \in [n] \setminus C$, then all the vectors in $\mathsf{occ}(C, \boldsymbol{a})$ share the same support.

Also, if some two vectors $\boldsymbol{u}, \boldsymbol{v} \in \mathsf{occ}(C, \boldsymbol{a})$ do not have the same support, then there will exist at least one index $j \in [n] \setminus C$ such that $\boldsymbol{u} \in \mathsf{occ}(C \cup \{j\}, (\boldsymbol{a}, 1))|$ and $\boldsymbol{v} \in \mathsf{occ}(C \cup \{j\}, (\boldsymbol{a}, 0))$ or the other way round, and therefore $|\mathsf{occ}(C \cup \{j\}, (\boldsymbol{a}, 1))| \notin \{0, |\mathsf{occ}(C, \boldsymbol{a})|\}$. Algorithm 3.13 precisely checks for this condition. The existence of some vector $\boldsymbol{v} \in \mathcal{V}$ (p-identifiable column), a subset of indices $C \subset [n]$ of size p, and a binary sub-string $\boldsymbol{b} \in \{0,1\}^{\leq p}$ follows from the fact that $\mathcal{V}$ is p-identifiable. Let us denote the subset of unknown vectors with distinct support in $\mathcal{V}$ by $\mathcal{V}^1$.

Once we recover the p-identifiable column of $\mathcal{V}^1$, we mark it as $\boldsymbol{u}^1$ and remove it from the set (if there are multiple p-identifiable columns, we arbitrarily choose one of them). Subsequently, we can modify the $|\mathsf{occ}(\cdot)|$ values for all $S \subseteq [n], |S| \in \{p, p+1\}$ and $\boldsymbol{t} \in \{0,1\}^p \cup \{0,1\}^{p+1}$ as follows:

$$\left|\mathsf{occ}^2(S, \boldsymbol{t})\right| \triangleq |\mathsf{occ}(S, \boldsymbol{t})| - |\mathsf{occ}(C, \boldsymbol{b})| \times \mathbb{1}[\operatorname{supp} \boldsymbol{u}^1|_S = \boldsymbol{t}]. \tag{3.7}$$

Notice that, Equation 3.7 computes $|\mathsf{occ}^2(S, \boldsymbol{t})| = |\{\boldsymbol{v}^i \in \mathcal{V}^2 \mid \operatorname{supp} \boldsymbol{v}^i|_S = \boldsymbol{t}\}|$ where $\mathcal{V}^2$ is formed by deleting all copies of $\boldsymbol{u}^1$ from $\mathcal{V}$. Since $\mathcal{V}^1$ is p-identifiable, there exists a p-identifiable column in $\mathcal{V}^1 \setminus \{\boldsymbol{u}^1\}$ as well which we can recover. More generally for $q > 2$, if $\boldsymbol{u}^{q-1}$ is the p-identifiable column with the unique binary sub-string $\boldsymbol{b}^{q-1}$ corresponding to the set of indices C^{q-1}, we will have for all $S \subseteq [n], |S| \in \{p, p+1\}$ and $\boldsymbol{t} \in \{0,1\}^p \cup \{0,1\}^{p+1}$

$$|\mathsf{occ}^q(S, \boldsymbol{t})| \triangleq \left|\mathsf{occ}^{q-1}(S, \boldsymbol{t})\right| - \left|\mathsf{occ}^{q-1}(C^{q-1}, \boldsymbol{b}^{q-1})\right| \times \mathbb{1}[\operatorname{supp} \boldsymbol{u}^{q-1}|_S = \boldsymbol{t}]$$

implying $|\mathsf{occ}^q(S, \boldsymbol{t})| = |\{\boldsymbol{v}^i \in \mathcal{V}^q \mid \operatorname{supp} \boldsymbol{v}^i|_S = \boldsymbol{t}\}|$ where $\mathcal{V}^q$ is formed deleting all copies of $\boldsymbol{u}^1, \boldsymbol{u}^2, \ldots, \boldsymbol{u}^{q-1}$ from $\mathcal{V}$. Applying these steps recursively and repeatedly using the property that $\mathcal{V}$ is p-identifiable, we can recover all the vectors present in $\mathcal{V}$.

Algorithm 3.13 requires the values of $|\mathsf{occ}(C, \boldsymbol{a})|$, and $|\mathsf{occ}(\tilde{C}, \tilde{\boldsymbol{a}})|$ for every p and $p+1$ sized subset of indices $C, \tilde{C} \subset [n]$, and every $\boldsymbol{a} \in \{0,1\}^p$, $\tilde{\boldsymbol{a}} \in \{0,1\}^{p+1}$. Using Lemma 3.13, we can compute all these values using $O(\ell^3(\ell k)^{p+2}\log(\ell k n)\log n/(1-2\eta)^2)$ MLC queries or $O(\ell^3(\ell k)^{p+2}\log(\ell k n)\log n)$ MLR queries with probability at least $1 - O(n^{-1})$. □

Algorithm 3.13 RECOVER p-IDENTIFIABLE SUPPORTS

Require: $|\mathsf{occ}(C, \boldsymbol{a})|$ for every $C \subset [n]$, $|C| = t \; \forall \; t \leq p+1$, and every $\boldsymbol{a} \in \{0,1\}^{\leq p+1}$.

Set count $= 1, i = 1$.
while count $\leq L$ **do**
  if $|\mathsf{occ}(C, \boldsymbol{a})| = w$, and $|\mathsf{occ}(C \cup \{j\}, (\boldsymbol{a}, 1))| \in \{0, w\}$ for all $j \in [n] \setminus C$ **then**
    Set $\operatorname{supp} \boldsymbol{u}^i|_C = \boldsymbol{a}$
    For every $j \in [n] \setminus C$, set $\operatorname{supp} \boldsymbol{u}^i|_j = b$, where $|\mathsf{occ}(C \cup \{j\}, (\boldsymbol{a}, b))| = w$.
    Set $\mathsf{Multiplicity}^i = w$.
    For all $\boldsymbol{t} \in \{0,1\}^p \cup \{0,1\}^{p+1}, S \subseteq [n]$ such that $|S| \in \{p, p+1\}$, update

$$|\mathsf{occ}(S, \boldsymbol{t})| \leftarrow |\mathsf{occ}(S, \boldsymbol{t})| - |\mathsf{occ}(C, \boldsymbol{a})| \times \mathbb{1}[\operatorname{supp} \boldsymbol{u}^i|_S = \boldsymbol{t}]$$

    count $=$ count $+ w$.
    $i = i + 1$.
  end if
end while
Return $\mathsf{Multiplicity}^j$ copies of $\operatorname{supp} \boldsymbol{u}^j$ for all $j < i$.

Recovery of flip-independent support matrix In this section, we present an algorithm that recovers the support of all the L unknown vectors in $\mathcal{B}$ provided $\mathcal{B}$ is flip-independent .

Proof of Theorem 3.11. The query complexity of the algorithm follows from Lemma 3.13. For any subset C of 3 indices, with probability $1 - O(1/n)$, we can compute $|\mathsf{occ}(C, \cdot)|$ using $O(L^3(Lk)^4 \log(Lkn)\log n)$ queries.

For every subset $\mathcal{F} \subseteq [n]$, we construct the tensor $\mathcal{A}^{\mathcal{F}}$ as follows:

$$\mathcal{A}^{\mathcal{F}}_{(i_1,i_2,i_3)} = |\mathsf{occ}((i_1, i_2, i_3), (a_{i1}, a_{i2}, a_{i3}))|,$$

Algorithm 3.14 RECOVER FLIP-INDEPENDENT SUPPORTS

Require: $|\mathsf{occ}(C, \boldsymbol{a})|$ for every $C \subset [n]$, such that $|C| = 3$, and all $\boldsymbol{a} \in \{0,1\}^3$. $|\mathsf{occ}(i, 1)|$ for all $i \in [n]$.

Set $\mathcal{U} = \{i \in [n] : |\mathsf{occ}(i, 1)| \neq 0\}$ and $\mathcal{U}' = \mathcal{U} \cup \{t\}$ where $t \in [n] \setminus \mathcal{U}$.
for each $\mathcal{F} \subset \mathcal{U}'$ **do**
  Construct tensor $\mathcal{A}^{\mathcal{F}}$ as follows:
  for every $(i_1, i_2, i_3) \in [n]^3$ **do**
    Set $\mathcal{A}^{\mathcal{F}}_{(i_1,i_2,i_3)} = |\mathsf{occ}((i_1, i_2, i_3), (a_{i1}, a_{i2}, a_{i3}))|$, where $a_{ij} = 0$ if $i_j \in \mathcal{F}$ and 1 otherwise.
  end for
  if $\mathsf{Jenerich}(\mathcal{A}^{\mathcal{F}})$ (Algorithm D.1 with input $\mathcal{A}^{\mathcal{F}}$) succeeds: **then**
    Let $\mathcal{A}^{\mathcal{F}} = \sum_{i=1}^{R} \lambda_i \boldsymbol{a}^i \otimes \boldsymbol{a}^i \otimes \boldsymbol{a}^i$ be the tensor decomposition of $\mathcal{A}$ such that $\boldsymbol{a}^i \in \{0,1\}^n$.
    For all $i \in [R]$, modify $\boldsymbol{a}^i$ by flipping entries in $\mathcal{F}$.
    Return λ_i columns with modified $\boldsymbol{a}^i$, $\forall i \in [R]$.
    break
  end if
end for

for all $(i_1, i_2, i_3) \in [n]^3$ where $a_{ij} = 0$ if $i_j \in \mathcal{F}$ and 1 otherwise. We then run Jennrich's algorithm on each $\mathcal{A}^{\mathcal{F}}$. Observe that for any binary vector $\boldsymbol{b} \in \{0,1\}^n$, the (i_1, i_2, i_3)-th entry of the rank-1 tensor $\boldsymbol{b} \otimes \boldsymbol{b} \otimes \boldsymbol{b}$ is 1 if $b_{i_1} = b_{i_2} = b_{i_3} = 1$, and 0 otherwise. Therefore, the tensor $\mathcal{A}^{\mathcal{F}}$ can be decomposed as $\mathcal{A}^{\mathcal{F}} = \sum_{i=1}^{R} \lambda_i \boldsymbol{a}^i \otimes \boldsymbol{a}^i \otimes \boldsymbol{a}^i$, where the vectors $\boldsymbol{a}^i \in \{0,1\}^n$, $i \in R$ are the support vectors of the unknown vectors that are flipped at indices in $\mathcal{F}$ with multiplicity λ_i.

Now if the support matrix of the unknown vectors is flip-independent, then there exists a subset of rows indexed by some $\mathcal{F}^\star \subseteq [n]$ such that flipping the entries of those rows results in a modified support matrix with all its distinct columns being linearly independent. Since the all zero rows of the support matrix $\boldsymbol{A}$ are linearly independent (flipped or not), we can search for $\mathcal{F}^\star$ as a subset of $\mathcal{U}'$. Since, $|\mathcal{U}'| \leq Lk + 1$, this step improves the search space for $\mathcal{F}^\star$ from $O(2^n)$ to $O(2^{Lk})$.

Therefore, Jennrich's algorithm on input $\mathcal{A}^{\mathcal{F}^\star}$ is guaranteed to succeed and returns the decomposition $\mathcal{A}^{\mathcal{F}^\star} = \sum_{i=1}^{R} \lambda_i \boldsymbol{a}^i \otimes \boldsymbol{a}^i \otimes \boldsymbol{a}^i$ as the sum of R rank-one tensors, where, $\boldsymbol{a}^i \in \{0,1\}^n$, $i \in [R]$ are modified support vectors with multiplicity λ_i. Subsequently,

we can again flip the entries of the recovered vectors indexed by $\mathcal{F}^\star$ to return the original support vectors. $\square$

Algorithm 3.15 RECOVER r-KRUSKAL RANK SUPPORTS

Let w be smallest integer such that $w \cdot (r-1) \geq 2L-1$.
Require: $|\mathsf{occ}(C, \mathbf{1}_w)|$ for every $C \subset [n]$ with $|C| = w$. $|\mathsf{occ}((i), 1)|$ for all $i \in [n]$.
Set $\mathcal{U} \triangleq \{i \in [n] : |\mathsf{occ}((i), 1)| \neq 0\}$.
Construct tensor $\mathcal{A}$ as follows:
for every $(i_1, \ldots, i_w) \in [n]^w$ **do**
  Set $\mathcal{A}_{(i_1,\ldots,i_w)} = |\mathsf{occ}((i_1, \ldots, i_w), \mathbf{1}_w)|$.
end for
for every $(\boldsymbol{b}^1, \boldsymbol{b}^2 \ldots, \boldsymbol{b}^L) \in \{0,1\}^n$ satisfying $\mathsf{supp}(\boldsymbol{b}^i) \subseteq \mathcal{U}$ **do**
  if $\mathcal{A} = \sum_{i=1}^{L} \boldsymbol{b}^i \otimes \boldsymbol{b}^i \cdots \otimes \boldsymbol{b}^i$ (w times) **then**
    Set $(\boldsymbol{b}^1, \boldsymbol{b}^2 \ldots, \boldsymbol{b}^L)$ to be the CP decomposition of $\mathcal{A}$ and Break
  end if
end for
Return CP decomposition of $\mathcal{A}$

Recovery of r-Kruskal rank supports In this section, we present an algorithm that recovers the support of all the L unknown vectors provided they have r-Kruskal rank supports. Recall that for any set of w indices $C \subset [n]$, $\mathsf{occ}(C, \mathbf{1}_w)$ denotes the set of unknown vectors that are supported on all indices in C.

Proof of Theorem 3.12. To recover the supports we first construct the following order w tensor: $\mathcal{A}_{(i_1,\ldots,i_w)} = |\mathsf{occ}((i_1, \ldots, i_w), \mathbf{1}_w)|$, for $(i_1, \ldots, i_w) \in [n]^w$. Observe that the tensor $\mathcal{A}$ can be written as the sum of L' ($L' < L$) rank one tensors

$$\mathcal{A} = \sum_{i=1}^{L'} \lambda^i \underbrace{\operatorname{supp} \boldsymbol{\beta}^i \otimes \ldots \otimes \operatorname{supp} \boldsymbol{\beta}^i}_{w\text{-times}}. \tag{3.8}$$

where $\boldsymbol{\beta}^1, \boldsymbol{\beta}^2, \ldots, \boldsymbol{\beta}^{L'}$ are the unknown vectors with distinct supports in $\mathcal{B}$ with λ^i being the multiplicity of $\operatorname{supp} \boldsymbol{\beta}^i$. Since the support matrix $\boldsymbol{A}$ of $\mathcal{B}$ has r-Kruskal rank, for any w such that $w \cdot (r-1) \geq 2L-1$, the decomposition of Eq. (3.8) is unique (Lemma 3.10). Notice that by a pre-processing step, we compute $\mathcal{U} \triangleq \{i \in [n] : |\mathsf{occ}((i), 1)| \neq 0\}$ to be

the union of the supports of the unknown vectors. Since we know that the underlying vectors of the tensor that we construct are binary, we can simply search exhaustively over all the possibilities ($O((Lk)^{Lk})$ of them (Steps 7-10) to find the unique CP decomposition of the tensor $\mathcal{A}$. For the special case when $w = 3$, Jennrich's algorithm (Algorithm D.1) can be used to efficiently compute the unique CP decomposition of the tensor $\mathcal{A}$.

Algorithm 3.15 needs to know the values of $|\mathsf{occ}(C, \mathbf{1}_w)|$ for every $C \subset [n]$, such that $|C| = w$. Using Lemma 3.13, these can be computed using $O(L^3(Lk)^{w+1} \log(Lkn) \log n)$ queries with probability at least $1 - O(1/n)$. □

3.4.3 Computing $\mathsf{occ}(C, \mathbf{a})$

In this section, we provide the proof of Lemma 3.13 that follows from the correctness and performance guarantees of the subroutines that for any $s < n$, compute $|\bigcup_{i \in \mathcal{S}} \mathsf{occ}((i), 1)|$ for every subset of indices $\mathcal{S}$ of size s.

Let $s < n$, then using queries constructed from CFFs of appropriate parameters we compute $|\bigcup_{i \in \mathcal{S}} \mathsf{occ}((i), 1)|$ for all subsets $\mathcal{S} \subset [n]$ of size s.

Lemma 3.14. *For any $1 < s < n$, there exists an algorithm to compute $\left|\bigcup_{i \in \mathcal{S}} \mathsf{occ}((i), 1)\right|$ for all $\mathcal{S} \subseteq [n], |\mathcal{S}| = s$ with probability at least $1 - O(n^{-2})$ using*

$$O(L^3(Lk)^{s+1} \log(Lkn) \log n)$$

queries.

The proof of Lemma 3.14 follows from the guarantees of Algorithm 3.16 provided in Section 3.4.4.

For the special case of $s = 1$, we use queries given by a RUFF of appropriate parameters to compute $|\mathsf{occ}((i), 1)|$ for all $i \in [n]$ using Algorithm 3.17 in Section 3.4.4.

Lemma 3.15. *There exists an algorithm to compute* $|\mathsf{occ}((i),1)|\ \forall\ i \in [n]$ *with probability at least* $1 - O(n^{-2})$ *using* $O(L^4k^2\log(Lkn)\log n)$ *queries.*

Both the above mentioned algorithms crucially use a subroutine that counts the number of unknown vectors in $\mathcal{B}$ that have a non-zero inner product with a given query vector $\boldsymbol{x}$. For any $\boldsymbol{x} \in \mathbb{R}^n$, define $\mathsf{nzcount}(\boldsymbol{x}) := \sum_{i=1}^{L} \mathbb{1}[\langle \boldsymbol{\beta}^i, \boldsymbol{x}\rangle \neq 0]$.

The problem of estimating $\mathsf{nzcount}(\boldsymbol{x})$ in the mixed linear regression model is challenging due to the presence of additive noise. Note that one can scale the queries with some large positive constant to minimize the effect of the additive noise. However, we also aim to minimize the SNR, and hence need more sophisticated techniques to estimate $\mathsf{nzcount}(\boldsymbol{x})$. We restrict our attention to only binary vectors $\boldsymbol{x}$ to estimate $\mathsf{nzcount}$ in our model which is sufficient for support recovery.

Lemma 3.16. *There exists an algorithm to compute* $\mathsf{nzcount}(\boldsymbol{x})$ *for any vector* $\boldsymbol{x} \in \{0,1\}^n$*, with probability at least* $1 - 2\exp(-T/36\pi L^2)$ *using only* T *queries. Moreover,* $\mathsf{SNR} = O(L^2 \max_{i\in[L]} \left|\left|\boldsymbol{\beta}^i\right|\right|_2^2/\delta^2)$.

Proof of Lemma 3.13. Using Algorithm 3.16 $(s-1)$ times and Algorithm 3.17, we can compute $\left|\bigcup_{i\in\mathcal{S}} \mathsf{occ}((i),1)\right|$ for all $\mathcal{S} \subseteq [n]$ such that $|\mathcal{S}| \leq s$.

From Lemma 3.14, we know that each call to Algorithm 3.16 with any $t \leq s$ uses $O(L^3(Lk)^{t+1}\log(Lkn)\log n)$ queries, and each succeeds with probability at least $1 - O(1/n^2)$. Therefore, taking a union bound over all $t < s$, we can compute $\left|\bigcup_{i\in\mathcal{S}} \mathsf{occ}((i),1)\right|$ for all $\mathcal{S} \subseteq [n]$, $|\mathcal{S}| \leq s$ using $O(L^3(Lk)^{s+1}\log(Lkn)\log n)$ queries with probability $1 - O(1/n)$.

We now show using by induction on s that the quantities

$$\left\{ \left| \bigcup_{i\in\mathcal{S}} \mathsf{occ}((i),1) \right| \ \forall\ \mathcal{S} \subseteq [n], |\mathcal{S}| \leq s \right\}$$

are sufficient to compute $|\mathsf{occ}(C, \boldsymbol{a})|$ for all subsets C of indices of size at most s, and any binary vector $\boldsymbol{a} \in \{0,1\}^{\leq s}$.

Base case ($t = 1$): The base case follows since we can infer both $|\mathsf{occ}((i), 1)|$ and $|\mathsf{occ}((i), 0)| = L - |\mathsf{occ}((i), 1)|$ from $|\mathsf{occ}((i), 1)|$ computed using Algorithm 3.17 $\forall i \in [n]$.

Inductive Step: Let us assume that the statement is true for $r < s$ i.e. we can compute $|\mathsf{occ}(\mathcal{C}, \boldsymbol{a})|$ for all subsets $\mathcal{C}$ satisfying $|\mathcal{C}| \leq r$ and any binary vector $\boldsymbol{a} \in \{0,1\}^{\leq r}$ from the quantities $\left\{\left|\bigcup_{i\in\mathcal{S}} \mathsf{occ}((i), 1)\right| \ \forall\ \mathcal{S} \subseteq [n], |\mathcal{S}| \leq r\right\}$ provided as input. Now, we claim that the statement is true for $r + 1$. For simplicity of notation we will denote by $\mathcal{S}_i \triangleq \mathsf{occ}(i, 1)$ the set of unknown vectors which have a 1 in the i^{th} entry. Note that we can also rewrite $\mathsf{occ}(\mathcal{C}, \boldsymbol{a})$ for any set $\mathcal{C} \subseteq [n], \boldsymbol{a} \in \{0,1\}^{|\mathcal{C}|}$ as

$$\mathsf{occ}(\mathcal{C}, \boldsymbol{a}) = \bigcap_{j\in\mathcal{C}'} \mathcal{S}_j \bigcap_{j\in\mathcal{C}\setminus\mathcal{C}'} \mathcal{S}_j^c$$

where $\mathcal{C}' \subseteq \mathcal{C}$ corresponds to the indices in $\mathcal{C}$ for which the entries in $\boldsymbol{a}$ is $\mathtt{1}$. Fix any set $i_1, i_2, \ldots, i_{r+1} \in [n]$. Then we can compute $\left|\bigcap_{b=1}^{r+1} \mathcal{S}_{i_b}\right|$ using the following equation:

$$(-1)^{r+2}\left|\bigcap_{b=1}^{r+1} \mathcal{S}_{i_b}\right| = \sum_{u=1}^{r}(-1)^{u+1} \sum_{\substack{j_1,j_2,\ldots,j_u\in\{i_1,i_2,\ldots,i_{r+1}\}\\ j_1<j_2<\cdots<j_u}} \left|\bigcap_{b=1}^{u} \mathcal{S}_{j_b}\right| - \left|\bigcup_{b=1}^{r+1} \mathcal{S}_{i_b}\right|.$$

Finally for any subset $\mathcal{Y} \subseteq \{i_1, i_2, \ldots, i_{r+1}\}$, we can compute $\left|\bigcap_{i_b\notin\mathcal{Y}} \mathcal{S}_{i_b} \bigcap_{i_b\in\mathcal{Y}} \mathcal{S}_{i_b}^c\right|$ using the following set of equations:

$$\begin{aligned}
\left|\bigcap_{i_b\notin\mathcal{Y}} \mathcal{S}_{i_b} \bigcap_{i_b\in\mathcal{Y}} \mathcal{S}_{i_b}^c\right| &= \left|\bigcap_{i_b\notin\mathcal{Y}} \mathcal{S}_{i_b} \bigcap \Big(\bigcup_{i_b\in\mathcal{Y}} \mathcal{S}_{i_b}\Big)^c\right| \\
&= \left|\bigcap_{i_b\notin\mathcal{Y}} \mathcal{S}_{i_b}\right| - \left|\bigcap_{i_b\notin\mathcal{Y}} \mathcal{S}_{i_b} \bigcap \Big(\bigcup_{i_b\in\mathcal{Y}} \mathcal{S}_{i_b}\Big)\right| \\
&= \left|\bigcap_{i_b\notin\mathcal{Y}} \mathcal{S}_{i_b}\right| - \left|\bigcup_{i_b\in\mathcal{Y}} \Big(\bigcap_{i_b\notin\mathcal{Y}} \mathcal{S}_{i_b} \bigcap \mathcal{S}_{i_b}\Big)\right|.
\end{aligned}$$

The first term is already pre-computed and the second term is again a union of intersection of sets. For any $i_b \in \mathcal{Y}$, let us define $\mathcal{Q}_{i_b} := \bigcap_{i_b \notin \mathcal{Y}} \mathcal{S}_{i_b} \bigcap \mathcal{S}_{i_b}$. Therefore we have

$$\left| \bigcup_{i_b \in \mathcal{Y}} \mathcal{Q}_{i_b} \right| = \sum_{u=1}^{|\mathcal{Y}|} (-1)^{u+1} \sum_{\substack{j_1, j_2, \ldots, j_u \in \mathcal{Y} \\ j_1 < j_2 < \cdots < j_u}} \left| \bigcap_{b=1}^{u} \mathcal{Q}_{j_b} \right|.$$

We can compute $\left|\bigcup_{i_b \in \mathcal{Y}} \mathcal{Q}_{i_b}\right|$ because the quantities on the right hand side of the equation have already been pre-computed (using our induction hypothesis). Therefore, the lemma is proved.

Therefore, for any subset $\mathcal{T} \subset \{0,1\}^s$, we can compute

$$\{|\mathsf{occ}(C, \boldsymbol{a})| \mid \ \forall \boldsymbol{a} \in \mathcal{T}, C \subset [n], |C| = s\}$$

by computing $\left\{\left|\bigcup_{i \in \mathcal{S}} \mathsf{occ}((i), 1)\right| \ \forall\ \mathcal{S} \subseteq [n], |\mathcal{S}| \leq s\right\}$ just once. □

3.4.4 Missing Proofs and Algorithms in computing $\mathsf{occ}(C, \mathbf{a})$

Compute $\left|\bigcup_{i \in \mathcal{S}} \mathsf{occ}((i), 1)\right|$ using Algorithm 3.16. In this section we present an algorithm to compute $\left|\bigcup_{i \in \mathcal{S}} \mathsf{occ}((i), 1)\right|$, for every $\mathcal{S} \subseteq [n]$ of size $|\mathcal{S}| = s$, using $|\mathsf{occ}((i), 1)|$ computed in the Section 3.4.4.

We will need an (s, Lk)-CFF for this purpose. Let $\mathcal{G} \equiv \{\mathcal{H}_1, \mathcal{H}_2, \ldots, \mathcal{H}_n\}$ be the required (s, Lk)-CFF of size n over alphabet $m = O((Lk)^{s+1} \log n)$. We construct a set of $L+1$ matrices $\mathcal{B} = \{\boldsymbol{B^{(1)}}, \ldots, \boldsymbol{B^{(L+1)}}\}$ where, each $\boldsymbol{B^{(w)}} \in \mathbb{R}^{m \times n}, w \in [L+1]$, is obtained from the (s, Lk)-CFF $\mathcal{G}$. The construction of these matrices varies slightly for the model in question.

For the mixture of linear regressions, we avoid the scaling of non-zero entries by a uniform scalar. We set $\boldsymbol{B}^{(w)}_{i,j}$ to be 1 if $i \in H_j$, and 0 otherwise. Note that in this case each $\boldsymbol{B}^{(w)}$ is identical. We see that the scaling by uniform scalar is not necessary

for the mixtures of linear regressions since the procedure to compute $\mathsf{nzcount}$ in this model (see Algorithm 3.18) scales the query vectors by a Gaussian scalar which is sufficient for our purposes.

Let $\mathcal{U} := \cup_{i\in[L]}\mathsf{supp}(\boldsymbol{\beta}^i)$ denote the union of supports of all the unknown vectors. Since each unknown vector is k-sparse, it follows that $|\mathcal{U}| \leq Lk$. From the properties of (s, Lk)-CFF, we know that for any tuple of s indices $(i_1, i_2, \ldots, i_s) \subset \mathcal{U}$, the set $(\bigcap_{t=1}^{s} \mathcal{H}_{i_i}) \setminus \bigcup_{q\in\mathcal{U}\setminus\{i_1,i_2,\ldots,i_s\}} \mathcal{H}_q$ is non-empty. This implies that for every $w \in [L+1]$, there exists at least one row of $\boldsymbol{B}^{(w)}$ that has a non-zero entry in the $i_1^{\mathsf{th}}, i_2^{\mathsf{th}}, \ldots, i_s^{\mathsf{th}}$ index, and 0 in all other indices $p \in \mathcal{U} \setminus \{i_1, i_2, \ldots, i_s\}$. In Algorithm 3.16 we use these rows as queries to estimate their $\mathsf{nzcount}$. In Lemma 3.14, we show that this estimated quantity is exactly $|\bigcup_{j=1}^{s} \mathsf{occ}((i), 1)|$ for that particular tuple $(i_1, i_2, \ldots, i_s) \subset \mathcal{U}$.

Algorithm 3.16 RECOVER UNION- $\left|\bigcup_{i\in\mathcal{S}} \mathsf{occ}((i), 1)\right|$ for all $\mathcal{S} \subseteq [n], |\mathcal{S}| = s, s \geq 2$.

Require: $|\mathsf{occ}((i), 1)|$ for every $i \in [n]$. $s \geq 2$.
Require: Construct $\boldsymbol{B} \in \mathbb{R}^{m\times n}$ from (s, Lk)-CFF of size n over alphabet $m = c_3(Lk)^{s+1} \log n$.
1: Let $\mathcal{U} := \{i \in [n] \mid |\mathsf{occ}((i), 1)| > 0\}$
2: Let batchsize $T_R = 10 \cdot (36\pi)L^2 \log(nm)$.
3: **for** every $p \in [m]$ **do**
4: Let $\mathsf{count}(p) := \max_{w\in[L+1]}\{\mathsf{nzcount}(\boldsymbol{B}^{(w)}[p])\}$ (obtained using Algorithm 3.18 with batchsize T_R).
5: **end for**
6: **for** every set $\mathcal{S} \subseteq [n]$ with $|\mathcal{S}| = s$ **do**
7: Let $p \in [m]$ such that $\boldsymbol{B}_{\boldsymbol{p,t}} \neq 0$ for all $t \in \mathcal{S}$, and $\boldsymbol{B}_{\boldsymbol{p,t'}} = 0$ for all $q \in \mathcal{U} \setminus \mathcal{S}$.
8: Set $\left|\bigcup_{i\in\mathcal{S}} \mathsf{occ}((i), 1)\right| = \mathsf{count}(p)$.
9: **end for**

Proof of Lemma 3.14. Computing each count (see Algorithm 3.16, line 8) requires $O(TL)$ queries, where $T = T_R$. Therefore, the total number of queries made by Algorithm 3.16 is at most

$$O(mT_RL) = O((Lk)^{s+1}L^3 \log(Lkn) \log n)$$

for $m = O((Lk)^{s+1} \log n)$ and $T_R = O(L^2 \log(nm))$. Also, observe that each $\mathsf{nzcount}$ is estimated correctly with probability at least $1 - O\left(1/Lmn^2\right)$. Therefore from union bound it follows that all the $(L+1)m$ estimations of count are correct with probability at least $1 - O\left(1/n^2\right)$.

Recall that the set $\mathcal{U}$ denotes the union of supports of all the unknown vectors. This set is equivalent to $\{i \in [n] \mid |\mathsf{occ}((i), 1)| > 0\}$.

Since for every $w \in [L+1]$, the support of the columns of $\boldsymbol{B}^{(w)}$ are the indicators of sets in $\mathcal{G}$, the (s, Lk)-CFF property implies that there exists at least one row (say, with index $p \in [m]$) of every $\boldsymbol{B}^{(w)}$ which has a non-zero entry in the $i_1^{\mathsf{th}}, i_2^{\mathsf{th}}, \ldots, i_s^{\mathsf{th}}$ index, and 0 in all other indices $q \in U \setminus \{i_1, i_2, \ldots, i_s\}$, i.e.,

$$\begin{aligned} \boldsymbol{B}^{(w)}_{\boldsymbol{p},\boldsymbol{t}} &\neq 0 \text{ for all } t \in \{i_1, i_2, \ldots, i_s\}, \text{ and} \\ \boldsymbol{B}^{(w)}_{\boldsymbol{p},\boldsymbol{t'}} &= 0 \text{ for all } t' \in \mathcal{U} \setminus \{i_1, i_2, \ldots, i_s\}. \end{aligned}$$

To prove the correctness of the algorithm, we need to show the following:

$$\left| \bigcup_{p \in \{i_1, i_2, \ldots, i_s\}} \mathsf{occ}(p, 1) \right| = \max_{w \in [L+1]} \{\mathsf{nzcount}(\boldsymbol{B}^{(w)}[p])\}$$

First observe that using the row $\boldsymbol{B}^{(w)}[p]$ as query will produce non-zero value for only those unknown vectors $\boldsymbol{\beta} \in \bigcup_{p \in \{i_1, i_2, \ldots, i_s\}} \mathsf{occ}(p, 1)$. This establishes the fact that $|\bigcup_{p \in \{i_1, i_2, \ldots, i_s\}} \mathsf{occ}(p, 1)| \geq \mathsf{nzcount}(\boldsymbol{B}^{(w)}[p])$.

To show the other side of the inequality, consider the set of $(L+1)$ s-dimensional vectors obtained by the restriction of rows $\boldsymbol{B}^{(w)}[p]$ to the coordinates $(i_1, i_2, \ldots, i_s)$,

$$\{(\boldsymbol{B}^{(w)}_{\boldsymbol{p},\boldsymbol{i_1}}, \boldsymbol{B}^{(w)}_{\boldsymbol{p},\boldsymbol{i_2}}, \ldots, \boldsymbol{B}^{(w)}_{\boldsymbol{p},\boldsymbol{i_s}}) \mid w \in [L+1]\}.$$

Since the $\mathsf{nzcount}$ scales the non-zero entries of the query vector $\boldsymbol{B}^{(w)}[p]$ by a Gaussian, the pairwise linear independence still holds. Therefore, each $\boldsymbol{\beta} \in$

$\bigcup_{p\in\{i_1,i_2,\ldots,i_s\}} \mathsf{occ}(p,1)$ can have $\langle \boldsymbol{B}^{(w)}[p], \boldsymbol{\beta}\rangle = 0$ for at most 1 of the w queries. So by pigeonhole principle, at least one of the query vectors $\boldsymbol{B}^{(w)}[p]$ will have $\langle \boldsymbol{B}^{(w)}[p], \boldsymbol{\beta}\rangle \neq 0$ for all $\boldsymbol{\beta} \in \bigcup_{p\in\{i_1,i_2,\ldots,i_s\}} \mathsf{occ}(p,1)$. Hence, $|\bigcup_{p\in\{i_1,i_2,\ldots,i_s\}} \mathsf{occ}(p,1)| \leq \max_w\{\mathsf{nzcount}(\boldsymbol{B}^{(w)}[p])\}$.

□

Computing $|\mathsf{occ}((i),1)|$ In this section, we show how to compute $|\mathsf{occ}(i,1)|$ for every index $i \in [n]$.

Let $\mathcal{F} = \{\mathcal{H}_1, \mathcal{H}_2, \ldots, \mathcal{H}_n\}$ be a $(d, Lk, 0.5)$-RUFF of size n over alphabet $[m]$. Construct the binary matrix $\boldsymbol{A} \in \{0,1\}^{m\times n}$ from $\mathcal{F}$, as $\boldsymbol{A}_{i,j} = 1$ if and only if $i \in \mathcal{H}_j$. Each column $j \in [n]$ of $\boldsymbol{A}$ is essentially the indicator vector of the set $\mathcal{H}_j$.

We use the rows of matrix $\boldsymbol{A}$ as query vectors to compute $|\mathsf{occ}((i),1)|$ for each $i \in [n]$. For each such query vector $\boldsymbol{x}$, we compute the $\mathsf{nzcount}(\boldsymbol{x})$ using Algorithm 3.18 with batchsize T_R. We choose T_R to be sufficiently large to ensure that $\mathsf{nzcount}$ is correct for all the queries with very high probability.

For every $h \in \{0,\ldots,L\}$, let $\boldsymbol{b}^h \in \{0,1\}^m$ be the indicator of the queries that have $\mathsf{nzcount}$ at least h. We show in Lemma 3.15 that the set of columns of $\boldsymbol{A}$ that have large intersection with $\boldsymbol{b}^h$, exactly correspond to the indices $i \in [n]$ that satisfy $|\mathsf{occ}((i),1)| \geq h$. This allows us to recover $|\mathsf{occ}((i),1)|$ exactly for each $i \in [n]$.

Proof of Lemma 3.15. Since $\boldsymbol{A}$ has $m = O(L^2k^2\log n)$ distinct rows, and each row is queried $T_R = O(L^2\log(mn))$ times, the total query complexity of Algorithm 3.17 is $O(L^4k^2\log(Lkn)\log n)$ for our model.

To prove the correctness, we first see that the $\mathsf{nzcount}$ for each query is estimated correctly using Algorithm 4.1 with overwhelmingly high probability. From Lemma 3.16 with $T_R = 4\cdot(36\pi)\cdot L^2\log mn$, it follows that each $\mathsf{nzcount}$ is estimated correctly with probability at least $1 - \frac{1}{mn^2}$. Therefore, by taking a union bound over all rows of $\boldsymbol{A}$, we estimate all the counts accurately with probability at least $1 - \frac{1}{n^2}$.

Algorithm 3.17 COMPUTE–$|\mathsf{occ}((i),1)|$

Require: Construct binary matrix $\boldsymbol{A} \in \{0,1\}^{m \times n}$ from $(d, Lk, 0.5) - \mathsf{RUFF}$ of size n over alphabet $[m]$, with $m = c_1 L^2 k^2 \log n$ and $d = c_2 Lk \log n$.

Initialize $\boldsymbol{b}^0, \boldsymbol{b}^1, \boldsymbol{b}^2, \ldots, \boldsymbol{b}^L$ to all zero vectors of dimension m.
Let batchsize $T_R = 4 \cdot (36\pi) \cdot L^2 \log mn$.
for $i = 1, \ldots, m$ **do**
  Set $w := \mathsf{nzcount}(\boldsymbol{A}[i])$
  (obtained using Algorithm 3.18 with batchsize T_R.)
  for $h = 0, 1, \ldots, w$ **do**
    Set $\boldsymbol{b}_i^h = 1$.
  end for
end for
for $h = 0, 1, \ldots, L$ **do**
  Set $\mathcal{C}_h = \{i \in [n] \mid |\mathsf{supp}(\boldsymbol{b}^h) \cap \mathsf{supp}(\boldsymbol{A}_i)| \geq 0.5d\}$.
end for
for $i = 1, 2, \ldots, n$ **do**
  Set $|\mathsf{occ}((i),1)| = h$ if $i \in \{\mathcal{C}_h \setminus \mathcal{C}_{h+1}\}$ for some $h \in \{0, 1, \ldots, L-1\}$.
  Set $|\mathsf{occ}((i),1)| = L$ if $i \in \mathcal{C}_L$
end for

We now show, using the properties of RUFF, that $|\mathsf{supp}(\boldsymbol{b}^h) \cap \mathsf{supp}(\boldsymbol{A}_i)| \geq 0.5d$ if and only if $|\mathsf{occ}((i),1)| \geq h$, for any $0 \leq h \leq L$. Let $i \in [n]$ be an index such that $|\mathsf{occ}((i),1)| \geq h$, i.e., there exist at least h unknown vectors that have a non-zero entry in their i^{th} coordinate. Also, let $U := \cup_{i \in [L]} \mathsf{supp}(\boldsymbol{\beta}^i)$ denote the union of supports of all the unknown vectors. Since each unknown vector is k-sparse, it follows that $|U| \leq Lk$. To show that $|\mathsf{supp}(\boldsymbol{b}^h) \cap \mathsf{supp}(\boldsymbol{A}_i)| \geq 0.5d$, consider the set of rows of $\boldsymbol{A}$ indexed by $W := \{\mathsf{supp}(\boldsymbol{A}_i) \setminus \cup_{j \in U \setminus \{i\}} \mathsf{supp}(\boldsymbol{A}_j)\}$. Since $\boldsymbol{A}$ is a $(d, Lk, 0.5) - \mathsf{RUFF}$, we know that $|W| \geq 0.5d$. We now show that $\boldsymbol{b}_t^h = 1$ for every $t \in W$. This follows from the observation that for $t \in W$, and each unknown vector $\boldsymbol{\beta} \in \mathsf{occ}((i),1)$, the query $\langle \boldsymbol{A}[t], \boldsymbol{\beta} \rangle = \boldsymbol{\beta}_i \neq 0$. Since $|\mathsf{occ}((i),1)| \geq h$, we conclude that $\mathsf{nzcount}(\boldsymbol{A}[t]) \geq h$, and therefore, $\boldsymbol{b}_t^h = 1$.

To prove the converse, consider an index $i \in [n]$ such that $|\mathsf{occ}((i),1)| < h$. Using a similar argument as above, we now show that $|\mathsf{supp}(\boldsymbol{b}^h) \cap \mathsf{supp}(\boldsymbol{A}_i)| < 0.5d$. Consider the set of rows of $\boldsymbol{A}$ indexed by $W := \{\mathsf{supp}(\boldsymbol{A}_i) \setminus \cup_{j \in U \setminus \{i\}} \mathsf{supp}(\boldsymbol{A}_j)\}$. Now observe that for each $t \in W$, and any unknown vector $\boldsymbol{\beta} \notin \mathsf{occ}((i),1)$, $\langle \boldsymbol{A}[t], \boldsymbol{\beta} \rangle = 0$. Therefore

$\mathsf{nzcount}(\boldsymbol{A}[t]) \leq |\mathsf{occ}((i),1)| < h$, and $\boldsymbol{b}_t^h = 0$ for all $t \in W$. Since $|W| \geq 0.5d$, it follows that $|\mathsf{supp}(\boldsymbol{b}^h) \cap \mathsf{supp}(\boldsymbol{A}_i)| < 0.5d$. For any $0 \leq h \leq L$, Algorithm 3.17. therefore correctly identifies the set of indices $i \in [n]$ such that $|\mathsf{occ}((i),1)| \geq h$. In particular, the set $C_h := \{i \in [n] \mid |\mathsf{occ}((i),1)| \geq h\}$. Therefore, the set $\mathcal{C}_h \setminus \mathcal{C}_{h+1}$ is exactly the set of indices $i \in [n]$ such that $|\mathsf{occ}((i),1)| = h$. □

3.4.5 Estimating nzcount

The main subroutine used to compute both $|\mathsf{occ}((i),1)|$ and $|\cup_j \mathsf{occ}((j),1)|$ is to estimate $\mathsf{nzcount}(\boldsymbol{x})$ - the number of unknown vectors that have a non-zero inner product with $\boldsymbol{x} \in \mathbb{R}^n$. We now provide an algorithm to estimate $\mathsf{nzcount}(\boldsymbol{x})$ using few queries.

We restrict our attention to only binary queries in this section which is sufficient for support recovery. Algorithm 3.18 queries repeatedly with a carefully crafted transformation $\mathbf{Tr}_{\boldsymbol{\gamma}}(\boldsymbol{x})$ of the input vector $\boldsymbol{x}$, and counts the number of responses that lie within a fixed range $[-a, a]$. This estimates count the number of unknown vectors that have a zero inner product with $\boldsymbol{x}$, and thereby estimates $\mathsf{nzcount}(\boldsymbol{x})$.

For any binary vector $\boldsymbol{x} \in \{0,1\}^n$, define as follows: $\mathbf{Tr}_{\boldsymbol{\gamma}} : \{0,1\}^n \to \mathbb{R}^n$

$$\mathbf{Tr}_{\boldsymbol{\gamma}}(\boldsymbol{x})_i = \begin{cases} 0 \text{ if } \ \boldsymbol{x}_i = 0 \\ \\ \mathcal{N}(0, \gamma^2) \text{ if } \ \boldsymbol{x}_i \neq 0. \end{cases}$$

For any $a, \sigma \in \mathbb{R}$, let us also define

$$\phi_1(a, \sigma) := \Pr_{W \sim \mathcal{N}(0,\sigma^2)}(W \in [-a, a]) \quad \text{and}$$

$$\phi_2(a, \sigma, \gamma) := \Pr_{W \sim \mathcal{N}(0,\sigma^2+\gamma^2)}(W \in [-a, a]).$$

From standard Gaussian concentration bounds, we know that

$$\phi_1(a, \sigma) = \mathsf{erf}\left(\tfrac{a}{\sqrt{2}\sigma}\right) \geq \tfrac{\sqrt{2}}{\sqrt{\pi}}\left(\tfrac{a}{\sigma} - \tfrac{a^3}{6\sigma^3}\right). \tag{3.9}$$

$$\phi_2(a,\sigma,\gamma) = \mathsf{erf}\Big(\tfrac{a}{\sqrt{2(\sigma^2+\gamma^2)}}\Big) \le a\sqrt{\tfrac{2}{\pi(\sigma^2+\gamma^2)}}. \tag{3.10}$$

Algorithm 3.18 QUERY($\boldsymbol{x} \in \{0,1\}^n, T, a, \gamma$)

Require: Query access to $\mathcal{O}$ and known σ, L.

1: **for** $i = 1, 2, \ldots, T$ **do**
2: Query with vector $\mathbf{Tr}_{\boldsymbol{\gamma}}(\boldsymbol{x})$ and obtain response $y_i \in \mathbb{R}$.
3: **end for**
4: Let $\hat{\mathsf{z}} = \mathsf{round}\Big(\frac{L\sum_{i=1}^{T} \mathbb{1}[y_i \in [-a,a]]}{T\phi_1(a,\sigma)}\Big)$.
5: Return $\hat{\mathsf{nz}} = L - \hat{\mathsf{z}}(\boldsymbol{x})$.

Proof of Lemma 3.16. Similar to the proof of Lemma 4.1 define $\mathsf{zcount}(\boldsymbol{x})$ denote the number of unknown vectors that have a zero inner product with $\boldsymbol{x}$. We show that Algorithm 3.18 estimates this quantity accurately, and hence $\mathsf{nzcount}(\boldsymbol{x}) = L - \mathsf{zcount}(\boldsymbol{x})$ can be inferred from it.

For the set of T responses $y_1, \ldots, y_T$ obtained from $\mathcal{O}$, define $U := \frac{\sum_i \mathbb{1}\left[y^i \in [-a,a]\right]}{T}$. Then,

$$\mathop{\mathbb{E}}_{\mathcal{V},\mathbf{Tr}_{\boldsymbol{\gamma}},Z}[U] = \Pr_{\mathcal{V},\mathbf{Tr}_{\boldsymbol{\gamma}},Z}\Big(\langle \mathbf{Tr}_{\boldsymbol{\gamma}}(\boldsymbol{x}), \boldsymbol{\beta}\rangle + Z \in [-a,a]\Big). \tag{3.11}$$

Note that for any $a \in \mathbb{R}$ and $\boldsymbol{x} \in \{0,1\}^n$, we have

$$\begin{aligned}
&\Pr_{\mathcal{V},\mathbf{Tr}_{\boldsymbol{\gamma}},Z}\Big(\langle \mathbf{Tr}_{\boldsymbol{\gamma}}(\boldsymbol{x}), \boldsymbol{\beta}\rangle + Z \in [-a,a]\Big) \\
&= \frac{1}{L}\Bigg(\sum_{i:\langle \boldsymbol{x},\boldsymbol{\beta}^i\rangle = 0} \Pr_{\mathbf{Tr}_{\boldsymbol{\gamma}},Z}\Big(\langle \mathbf{Tr}_{\boldsymbol{\gamma}}(\boldsymbol{x}), \boldsymbol{\beta}^i\rangle + Z \in [-a,a]\Big) \\
&+ \sum_{i:\langle \boldsymbol{x},\boldsymbol{\beta}^i\rangle \neq 0} \Pr_{\mathbf{Tr}_{\boldsymbol{\gamma}},Z}\Big(\langle \mathbf{Tr}_{\boldsymbol{\gamma}}(\boldsymbol{x}), \boldsymbol{\beta}^i\rangle + Z \in [-a,a]\Big)\Bigg)
\end{aligned}$$

Observe that if $\langle \boldsymbol{x}, \boldsymbol{\beta}^i \rangle = 0$, then $\langle \mathbf{Tr}_{\boldsymbol{\gamma}}(\boldsymbol{x}), \boldsymbol{\beta}^i \rangle + Z \sim \mathcal{N}(0, \sigma^2)$, and if $\langle \boldsymbol{x}, \boldsymbol{\beta}^i \rangle \neq 0$, then $\langle \mathbf{Tr}_{\boldsymbol{\gamma}}(\boldsymbol{x}), \boldsymbol{\beta}^i \rangle \sim \mathcal{N}(0, \gamma^2 \left|\left| \boldsymbol{x} \odot \boldsymbol{\beta}^i \right|\right|_2^2 + \sigma^2)$, where $\boldsymbol{u} \odot \boldsymbol{\beta}$ denotes the entry-wise product of $\boldsymbol{u}, \boldsymbol{\beta}$. It then follows that

$$\begin{aligned} \frac{\mathsf{zcount}(\boldsymbol{x})}{L} \cdot \phi_1(a, \sigma) &\leq \Pr_{\mathcal{V}, \mathbf{Tr}_{\boldsymbol{\gamma}}, Z} \Big(\langle \mathbf{Tr}_{\boldsymbol{\gamma}}(\boldsymbol{x}), \boldsymbol{\beta} \rangle + Z \in [-a, a] \Big) \\ &\leq \frac{\mathsf{zcount}(\boldsymbol{x})}{L} \cdot \phi_1(a, \sigma) + \phi_2(a, \sigma, \gamma\delta). \end{aligned} \tag{3.12}$$

Setting the parameters $a = \sigma/2$ and $\gamma = 2\sqrt{2L}\sigma/\delta$, from Equation 3.9, we get that

$$\phi_1(a, \sigma) \geq \frac{23\sqrt{2}}{48\sqrt{\pi}} \quad \text{and} \quad \phi_2(a, \sigma, \gamma\delta) \leq \frac{\sqrt{2}}{4L\sqrt{\pi}}.$$

and therefore, $4L\phi_2(a, \sigma, \gamma\delta) \leq \phi_1(a, \sigma)$.

Combining this observation with Equation 3.11 and Equation 3.12, we then get that

$$\begin{aligned} \frac{\mathsf{zcount}(\boldsymbol{x})}{L} \cdot \phi_1(a, \sigma) &\leq \mathop{\mathbb{E}}_{\mathcal{V}, \mathbf{Tr}_{\boldsymbol{\gamma}}, Z}[U] \\ &\leq \frac{\mathsf{zcount}(\boldsymbol{x})}{L} \cdot \phi_1(a, \sigma) + \frac{1}{4L} \cdot \phi_1(a, \sigma). \end{aligned} \tag{3.13}$$

From Equation 3.13, we observe that if $|U - \mathbb{E}[U]| \leq \frac{1}{4L} \cdot \phi_1(a, \sigma)$, then $\mathsf{zcount}(\boldsymbol{x}) - \frac{1}{4} \leq \frac{LU}{\phi_1(a,\sigma)} \leq \mathsf{zcount}(\boldsymbol{x}) + \frac{1}{2}$. Since $\mathsf{zcount}(\boldsymbol{x})$ is integral, it follows that if $|U - \mathbb{E}[U]| \leq \frac{1}{4L} \cdot \phi_1(a, \sigma)$, the estimate $\hat{\mathsf{z}} = \mathsf{round}\big(\frac{LU}{\phi_1(a,\sigma)}\big)$ computed in Algorithm 3.18 will correctly estimate $\mathsf{zcount}(\boldsymbol{x})$.

The correctness of the algorithm then follows from Chernoff bound[26]

$$\begin{aligned} \Pr\Big(|U - \mathbb{E}U| \geq \frac{\phi_1(a, \sigma)}{4L} \Big) &\leq 2 \exp\Big(- \frac{T \phi_1(a, \sigma)^2}{8L^2} \Big) \\ &\leq 2 \exp\Big(- \frac{T}{36\pi L^2} \Big). \end{aligned}$$

Moreover, From the definition of SNR, and the fact that $\mathbb{E}Z^2 = \sigma^2$, we have

$$\begin{aligned}
\mathsf{SNR} &\leq \frac{1}{\sigma^2} \cdot \max_{\boldsymbol{x} \in \{0,1\}^n} \max_{i \in [L]} \mathbb{E} \langle \mathbf{Tr}_{\boldsymbol{\gamma}}(\boldsymbol{x}), \boldsymbol{\beta}^i \rangle^2 \\
&\leq \frac{1}{\sigma^2} \cdot \gamma^2 \max_{i \in [L]} \left|\left| \boldsymbol{\beta}^i \right|\right|_2^2 \\
&= O(L^2 \max_{i \in [L]} \left|\left| \boldsymbol{\beta}^i \right|\right|_2^2 / \delta^2) \text{ for } \gamma = 2\sqrt{2} L \sigma / \delta.
\end{aligned}$$

□

CHAPTER 4

MIXTURES OF SPARSE LINEAR CLASSIFIERS

4.1 Introduction

Continuing the work presented in chapter 3, we study the similar problem of modeling the separating hyperplane corresponding to a binary output variable by a mixture of linear function of the explanatory features. Again, as in chapter 3, we study the setting where the weight vectors characterizing the linear functions are sparse, and further, we can query the label of a particular feature from an oracle. Our setting generalizes the well-known 1-bit compressed sensing setting but it is significantly more difficult since the tag of which linear function the oracle response corresponds to is latent. Quite interestingly, we find subtle similarities in this problem and semi-supervised clustering with overlapping clusters [84]; namely, the prima facie assumptions required for recovery of the latent clusters in the adversarial setting in [84] is exactly the same as the assumptions required for recovery of the sparse weight vectors in the setting. This algorithms and results of this chapter can be used for designing recommendation systems or advertisement engines where there are multiple users sharing the same account and the objective is to recommend items to every user based on their consumption history without compromising their privacy. The results of this chapter can be found in [67] and [69].

Let $\mathcal{B} \equiv \{\boldsymbol{\beta}^1, \boldsymbol{\beta}^2, \ldots, \boldsymbol{\beta}^\ell \in \mathbb{R}^n\}$ be a set of ℓ unknown k-sparse vectors. Let $\mathsf{sign} : \mathbb{R} \to \{-1, +1\}$ be the sign function that takes a real number and returns its

sign. We consider an oracle $\mathcal{O} : \mathbb{R}^n \to \{-1, +1\}$ that takes as input a query vector $\boldsymbol{x} \in \mathbb{R}^n$ and returns

$$\mathsf{sign}(\langle \boldsymbol{x}, \boldsymbol{\beta} \rangle) \cdot (1 - 2Z)$$

where $\boldsymbol{\beta}$ is sampled uniformly at random from $\mathcal{B}$ and $Z \sim \mathsf{Ber}(\eta)$, the noise, is a Bernoulli random variable that is 1 with probability η and 0 with probability $1 - \eta$.

As described above, our work focuses on recovering the unknown classifiers in the *query model* that was used in [154] and in Chapter 3 to study the mixtures of sparse linear regressions. In the query model, we assume the existence of an oracle $\mathcal{O}$ which when queried with a vector $\boldsymbol{v} \in \mathbb{R}^n$, samples one of the classifiers $\boldsymbol{\beta} \in \{\boldsymbol{\beta}^1, \boldsymbol{\beta}^2, \ldots, \boldsymbol{\beta}^\ell\}$ uniformly at random and returns the label of $\boldsymbol{v}$ assigned by the sampled classifier $\boldsymbol{\beta}$. The response of the oracle might be altered with a probability of η due to the presence of Bernoulli Noise. The goal of approximate recovery is to reconstruct each of the unknown classifiers using small number of oracle queries. The problem can be formalized as follows:

Problem 4.1 (ϵ-recovery)**.** *Given $\epsilon > 0$, and query access to oracle $\mathcal{O}$, find k-sparse vectors $\{\hat{\boldsymbol{\beta}}^{\mathbf{1}}, \hat{\boldsymbol{\beta}}^{\mathbf{2}}, \ldots, \hat{\boldsymbol{\beta}}^{\boldsymbol{\ell}}\}$ such that for some permutation $\sigma : [\ell] \to [\ell]$*

$$\left\| \frac{\boldsymbol{\beta}^{\boldsymbol{i}}}{\|\boldsymbol{\beta}^{\boldsymbol{i}}\|_2} - \frac{\hat{\boldsymbol{\beta}}^{\boldsymbol{\sigma(i)}}}{\|\hat{\boldsymbol{\beta}}^{\boldsymbol{\sigma(i)}}\|_2} \right\|_2 \leq \epsilon \quad \forall\, i \in [\ell].$$

Since from the classification labels, we lose the magnitude information of the unknown vectors, we assume each $\boldsymbol{\beta}^{\boldsymbol{i}}$ and the estimates $\hat{\boldsymbol{\beta}}^{\boldsymbol{i}}$ to have a unit norm.

Similar to the literature on one-bit compressed sensing, one of our proposed solutions employs a two-stage algorithm to recover the unknown vectors. In the first stage the algorithm recovers the support of every vector, and then in the second stage, approximately recovers the vectors using the support information.

For any vector $\boldsymbol{v} \in \mathbb{R}^n$, let $\mathsf{supp}(\boldsymbol{v}) := \{i \in [n] \mid \boldsymbol{v}_i \neq 0\}$ denote the support of $\boldsymbol{v}$. The problem of support recovery is then defined as follows:

Problem 4.2 (Support Recovery)**.** *Given query access to oracle $\mathcal{O}$, construct*

$$\{\hat{\boldsymbol{\beta}}^{\mathbf{1}}, \hat{\boldsymbol{\beta}}^{\mathbf{2}}, \ldots, \hat{\boldsymbol{\beta}}^{\boldsymbol{\ell}}\}$$

such that for some permutation $\sigma : [\ell] \to [\ell]$

$$\mathsf{supp}(\hat{\boldsymbol{\beta}}^{\boldsymbol{\sigma}(\boldsymbol{i})}) = \mathsf{supp}(\boldsymbol{\beta}^{\boldsymbol{i}}) \quad \forall\, i \in [\ell]$$

For both these problems, we primarily focus on minimizing the query complexity of the problem, *i.e.*, minimizing the number of queries that suffice to approximately recover all the sparse unknown vectors or their supports. However, all the algorithms proposed in this work also run in time $O(\mathrm{poly}(q))$, where q is the query complexity of the algorithm.

Mixture of sparse linear classifiers. The main technical difficulty that arises in recovering multiple sparse hyperplanes using 1-bit measurements (labels) is to *align* the responses of different queries concerning a fixed unknown hyperplane. To understand this better, let us consider the case when $\ell = 2$ (see Figure 4.1). Let $\boldsymbol{\beta}^{\mathbf{1}}, \boldsymbol{\beta}^{\mathbf{2}}$ be two unknown k-sparse vectors corresponding to two sparse linear classifiers. On each query, the oracle samples a $\boldsymbol{\beta}^i$, for $i \in \{1, 2\}$, uniformly at random and returns the binary label corresponding to it ($+$ or $-$). One can query the oracle repeatedly with the same query vector to ensure a response from both the classifiers with overwhelmingly high probability.

For any query vector if the responses corresponding to the two classifiers are the same (*i.e.*, $(+,+)$ or $(-,-)$), then we do not gain any information separating the two classifiers. We might still be able to reconstruct some sparse hyperplanes, but the

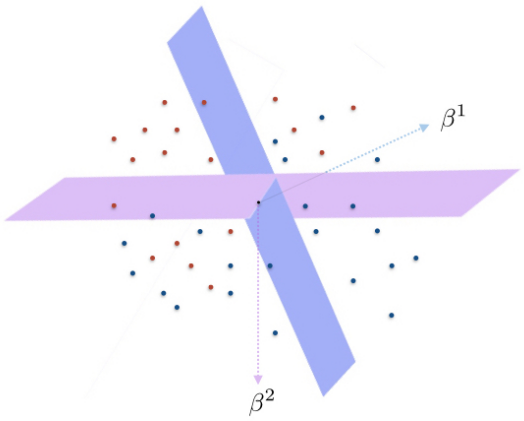


Figure 4.1: Recover the two classifiers given red and blue dots. How many such points do we require in order to recover both $\boldsymbol{\beta}^1$ and $\boldsymbol{\beta}^2$?

recovery guarantees of such an algorithm will be poor. On the other hand, if both the responses are different (*i.e.*, $(+,-)$), then we do not know which labels correspond to a particular classifier. For example, if the responses are $(+,-)$ and $(+,-)$ for two distinct query vectors, then we do not know if the 'plusses' correspond to the same classifier. This issue of alignment makes the problem challenging. Such alignment issues are less damning in the case of mixture of linear regressions even in the presence of noise (see [154] and the approximate recovery guarantees presented in Chapter 3) since we can utilize the magnitude information of the inner products (labels) to our advantage.

One of the challenges in our study is to recover the supports of the two unknown vectors. The support recovery results in Mixture of Linear Classifiers follow from the techniques presented in Chapter 3. Once we obtain the supports, we use an additional $O(\frac{k}{\epsilon}\log nk)$ Gaussian queries (with a slight modification) to approximately recover the individual vectors.

We then extend this two-step process (using more general classes of UFFs) to recover a mixture of ℓ different sparse vectors under the assumption that the support of no vector is contained in the union of supports of the remaining ones (Assumption 4.1).

The assumption implies that if the sparse vectors are arranged as columns of a matrix, then the matrix contains the identity matrix as a permutation of the rows. This separability condition appears before in [8, 56, 129] in the context of nonnegative integer matrix factorization, which is a key tool that we will subsequently use to prove our results. To quote [8] in the context of matrix factorization, "an approximate separability condition is regarded as a fairly benign assumption and is believed to hold in many practical contexts in machine learning." We believe this observation holds for our context as well (each classifier uses some unique feature).

We show that with this support separability condition, $\tilde{O}(\ell^6 k^3)$ queries suffice for support recovery of ℓ different k-sparse vectors. Further, using $\tilde{O}((\ell^3 k/\epsilon))$ queries, we can recover each of the $\boldsymbol{\beta^i}$'s, for $i \in \{1, \ldots, \ell\}$ up to ϵ precision (see Theorem 4.2 and Theorem 4.5). The two-stage procedure described above, can be made completely non-adaptive using queries from union free families (see Theorem 4.6). Note that for the problem of support recovery, we do not need this assumption of support separability; our results provide worst case guarantees and significantly improved results under extremely mild conditions.

Furthermore, for $\ell = 2$, we see that the support condition (Assumption 4.1) is not necessary. We can approximately recover the two unknown vectors provided a) they are not extremely sparse and, b) each $\boldsymbol{\beta^i} \in \delta\mathbb{Z}^n$ for some $\delta > 0$. To prove this, we borrow the tools from [4] who give guarantees for 1-bit compressed sensing using sub-Gaussian vectors. In particular, we use queries with independent Bernoulli coordinates which are sub-Gaussian. These discrete random queries (as opposed to continuous Gaussians) along with condition (b), enables us to align the labels corresponding to the two unknown vectors. (see Theorem 4.7 for more details). Note that condition (a) is due to the result by [4] and is necessary for recovery using sub-Gaussian queries and (b) is a mild assumption on the precision of the unknown vectors, which was also necessary [97, 154] for learning the mixture of sparse linear regressions.

We leave the problem of designing a query scheme that works for approximate recovery for any general ℓ without any assumptions as an open problem. For large ℓ, finding out the dependence of query complexity on ℓ is also a natural question. Overall, this study leads to an interesting set of questions that are technically demanding as well as quite relevant to practical modeling of heterogeneous data that are ubiquitous in applications. For instance, in recommendation systems, where the goal is to identify the factors governing the preferences of individual members of a group via crowdsourcing while preserving the anonymity of their responses.

Organization: The rest of the chapter is organizes as follows: in Section 4.2, we describe our contributions in this chapter namely our results for support recovery and approximate recovery in mixtures of linear classifiers. In Section 4.3, we provide details on estimating different counts and on family of sets. In Section 4.4, we prove our results on support recovery without any assumptions. In Section 4.5 and Section 4.6, we outline the details of our two-stage algorithm and single-stage algorithm for approximate recovery for any number of unknown vectors under the separability assumption. In Section 4.7, we relax the separability assumption and provide guarantees for approximate recovery under extremely mild assumptions. Finally, in Section 4.8, we provide simulations and experiments on real world data to complete our theoretical results.

4.2 Our contributions

4.2.1 Support Recovery

Recall that the set of unknown vectors is denoted by $\mathcal{B} \equiv \{\boldsymbol{\beta}^1, \boldsymbol{\beta}^2, \ldots, \boldsymbol{\beta}^\ell\}$. Let $\mathbf{A} \in \{0,1\}^{n \times \ell}$ denote the support matrix corresponding to $\mathcal{B}$ where each column vector $\boldsymbol{A}_i \in \{0,1\}^n$ represents the support of the i^{th} unknown vector $\boldsymbol{\beta}^i$. Our results for the support recovery in Mixtures of Linear Classifiers are very similar to the support recovery guarantees presented in Mixtures of Linear Regressions (see Chapter 3). For

the necessary definitions of p-identifiability, flip-independence and r-Kruskal rank support, we ask the reader to refer to Chapter 3. Below, we describe our results (note that they explicitly show the scaling with noise):

In our first result, we recover the support of the unknown vectors with small number of oracle queries provided the support matrix of $\mathcal{B}$ is p-identifiable.

Theorem 4.1. *Let $\mathcal{B}$ be a set of ℓ unknown vectors in $\mathbb{R}^n$ such that $\mathcal{B}$ is p-identifiable. Then, Algorithm 3.13 recovers the support of all the unknown vectors in $\mathcal{B}$ with probability at least $1 - O(1/n)$ using $O\left(\frac{\ell^3(\ell k)^{p+2}\log(\ell k n)\log n}{(1-2\eta)^2}\right)$ queries.*

Recall that in fact, all binary matrices with distinct columns are p-identifiable for some sufficiently large p.

Theorem 4.2 (Restatement of Theorem 3.10)**.** *Any $n \times \ell$, (with $n > \ell$) binary matrix with all distinct columns is p-identifiable for some $p \leq \log \ell$.*

Thus, we have the following corollary characterizing the unconditional worst-case guarantees for support recovery:

Corollary 4.1. *Let $\mathcal{B}$ be a set of ℓ unknown vectors in $\mathbb{R}^n$. Then, Algorithm 3.13 recovers the support of all the unknown vectors in $\mathcal{B}$ with probability at least $1-O(1/n)$ using $O\left(\frac{\ell^3(\ell k)^{\log \ell+2}\log(\ell k n)\log n}{(1-2\eta)^2}\right)$ queries.*

Proof. The proof follows from the fact that any set $\mathcal{B}$ of ℓ unknown vectors in $\mathbb{R}^n$ must have p-identifiable supports for $p \leq \log \ell$. □

As in Chapter 3, under some assumptions on the unknown support, e.g.flip-independence, we have better results.

Theorem 4.3. *Let $\mathcal{B}$ be a set of ℓ unknown vectors in $\mathbb{R}^n$ such that $\mathcal{B}$ is flip-independent. Then, Algorithm 3.14 recovers the support of all the unknown vectors in $\mathcal{B}$ with probability at least $1 - O(1/n)$ using $O\left(\frac{\ell^3(\ell k)^4\log(\ell k n)\log n}{(1-2\eta)^2}\right)$ queries.*

We also leverage the property of small Kruskal rank of the support matrix to show:

Theorem 4.4. *Let $\mathcal{B}$ be a set of ℓ unknown vectors in $\mathbb{R}^n$ that has r-Kruskal rank support with $r \geq 2$. Let $w = \lceil \frac{2\ell-1}{r-1} \rceil$. Then, Algorithm 3.15 recovers the support of all the unknown vectors in $\mathcal{B}$ with probability at least $1 - O(1/n)$ using $O\left(\frac{\ell^3(\ell k)^{w+1}\log(\ell k n)\log n}{(1-2\eta)^2}\right)$ queries*

From our discussion on these aforementioned matrix properties in Chapter 3, essentially, we 1) provide algorithms for support recovery without any assumptions, 2) and also provide significantly better guarantees under extremely mild assumptions that we conjecture to be always true.

4.2.2 Approximate Recovery in Noiseless Setting

For simplicity, we provide the results for approximate recovery in the noiseless setting i.e. when $\eta = 0$. However this assumption is not necessary as our results can be easily extended to the noisy case. In order to present our first set of results, we need certain assumption regarding the separability of supports of the unknown vectors. In particular, we want each component of the mixture to have a unique identifying coordinate. More formally, it can be stated as follows:

Assumption 4.1. *For every $i \in [\ell]$, $\mathsf{supp}(\boldsymbol{\beta}^i) \not\subseteq \bigcup_{j:j\neq i} \mathsf{supp}(\boldsymbol{\beta}^j)$, i.e. the support of any unknown vector is not contained in the union of the support of the other unknown vectors.*

Two-stage algorithm: First, we propose a two-stage algorithm for ϵ-recovery of the unknown vectors. In the first stage of the algorithm, we recover the support of the unknown vectors (Theorem 4.2), followed by ϵ-recovery using the deduced supports (Theorem 4.5) in the second stage. Each stage in itself is non-adaptive, *i.e.*, the queries do not depend on the responses of previously made queries.

Corollary 4.2. *Let $\mathcal{B}$ be a set of ℓ unknown k-sparse vectors in $\mathbb{R}^n$ that satisfy Assumption 4.1. There exists an algorithm to recover the support of every unknown vector in $\mathcal{B}$ with probability at least $1 - O(1/n^2)$, using $O(\ell^6 k^3 \log^2 n)$ non-adaptive queries to oracle $\mathcal{O}$.*

Proof. The separability assumption 4.1 implies that the support matrix $\boldsymbol{A}$ of the set of unknown vectors $\mathcal{B}$ is 1-identifiable. □

Now using this support information, we can approximately recover the unknown vectors using an additional $\tilde{O}(\ell^3 k)$ non-adaptive queries.

Theorem 4.5. *Let $\mathcal{B}$ be a set of ℓ unknown k-sparse vectors in $\mathbb{R}^n$ that satisfy Assumption 4.1. There exists a two-stage algorithm that uses*

$$O\Big(\ell^6 k^3 \log^2 n + (\ell^3 k/\epsilon) \log(nk/\epsilon) \log(k/\epsilon)\Big)$$

oracle queries for the ϵ-recovery of all the unknown vectors in $\mathcal{B}$ with probability at least $1 - O(1/n)$.

Remark 4.1. *We note that for the two-stage recovery algorithm to be efficient, we require the magnitude of non-zero entries of the unknown vectors to be non-negligible (at least $1/poly(n)$). This assumption however is not required to bound the query complexity of the algorithm which is the main focus of this work.*

Completely non-adaptive algorithm: Next, we show that the entire ϵ-recovery algorithm can be made non-adaptive (single-stage) at the cost of increased query complexity.

Theorem 4.6. *Let $\mathcal{B}$ be a set of ℓ unknown k-sparse vectors in $\mathbb{R}^n$ that satisfy Assumption 4.1. There exists an algorithm that uses $O\Big((\ell^{\ell+3} k^{\ell+2}/\epsilon) \log n \log(n/\epsilon) \log(k/\epsilon)\Big)$ non-adaptive oracle queries for the ϵ-recovery of all the unknown vectors in $\mathcal{B}$ with probability at least $1 - O(1/n)$.*

Note that even though the one-stage algorithm uses many more queries than the two-stage algorithm, a completely non-adaptive is highly parallelizable as one can choose all the query vectors in advance. Also, in the $\ell = O(1)$ regime, the query complexity is comparable to its two-stage analogue.

While we mainly focus on minimizing the query complexity, all the algorithms proposed in this work run in $\mathsf{poly}(n)$ time assuming every oracle query takes $\mathsf{poly}(n)$ time and $\ell = o(\log n)$.

Non-adaptive algorithm for $\ell = 2$ without Assumption 4.1: We observe that for $\ell = 2$, we do not need the separability condition (Assumption 4.1) required earlier. Instead we just need a mild assumption on the precision δ, and the sparsity of the unknown vectors. In particular, we propose an algorithm for the ϵ-recovery of the two unknown vectors using $\tilde{O}(k^3 + k/\epsilon)$ queries provided the unknown vectors have some finite precision and are not extremely sparse.

Assumption 4.2. *For $\boldsymbol{\beta} \in \mathcal{B}$, $\|\boldsymbol{\beta}\|_\infty = o(1)$.*

Assumption 4.2 ensures that we can safely invoke the result of [4] who use the exact same assumption in the context of 1-bit compressed sensing using sub-Gaussian queries.

Theorem 4.7. *Let $\boldsymbol{\beta}^1, \boldsymbol{\beta}^2$ be two k-sparse vectors in $\mathbb{R}^n$ that satisfy Assumption 4.2. Let $\delta > 0$ be the largest real such that $\boldsymbol{\beta^1}, \boldsymbol{\beta^2} \in \delta\mathbb{Z}^n$. There exists an algorithm that uses $O(k^3 \log^2 n + (k^2/\epsilon^4\delta^2) \log^2(n/k\delta^2))$ (adaptive) oracle queries for the ϵ-recovery of $\boldsymbol{\beta^1}, \boldsymbol{\beta^2}$ with probability at least $1 - O(1/n)$.*

Moreover, if $\mathsf{supp}(\boldsymbol{\beta^1}) \neq \mathsf{supp}(\boldsymbol{\beta^2})$, then there exists a two-stage algorithm for the ϵ-recovery of the two vectors using only $O(k^3 \log^2 n + (k/\epsilon) \log(nk/\epsilon) \log(k/\epsilon))$ non-adaptive oracle queries.

Also, the ϵ-recovery algorithm proposed for Theorem 4.7 runs in time $\mathsf{poly}(n, 1/\delta)$.

No sparsity constraint: We can infact avoid the sparsity constraint altogether for the case of $\ell = 2$. Since in this setting, we consider the support of both unknown vectors to include all coordinates, we do not need a support recovery stage. We then get a single stage and therefore completely non-adaptive algorithm for ϵ-recovery of the two unknown vectors.

Corollary 4.3. *Let $\boldsymbol{\beta^1}, \boldsymbol{\beta^2}$ be two unknown vectors in $\mathbb{R}^n$ that satisfy Assumption 4.2. Let $\delta > 0$ be the largest real such that $\boldsymbol{\beta^1}, \boldsymbol{\beta^2} \in \delta\mathbb{Z}^n$. There exists an algorithm that uses $O((n^2/\epsilon^4\delta^2)\log(1/\delta))$ non-adaptive oracle queries for the ϵ-recovery of $\boldsymbol{\beta^1}, \boldsymbol{\beta^2}$ with probability at least $1 - O(1/n)$.*

4.3 Preliminaries

Let $[n]$ to denote the set $\{1, 2, \ldots, n\}$. For any vector $\boldsymbol{v} \in \mathbb{R}^n$, $\mathsf{supp}(\boldsymbol{v})$ denotes the support and $\boldsymbol{v}_i$ denote the i^{th} entry (coordinate) of the vector $\boldsymbol{v}$. We will use $\boldsymbol{e_i}$ to denote a vector which has $\mathtt{1}$ only in the i^{th} position and is $\mathtt{0}$ everywhere else. We will use the notation $\langle \boldsymbol{a}, \boldsymbol{b} \rangle$ to denote the inner product between two vectors $\boldsymbol{a}$ and $\boldsymbol{b}$ of the same dimension. For a matrix $\boldsymbol{A} \in \mathbb{R}^{m \times n}$, let $\boldsymbol{A_i} \in \mathbb{R}^n$ be its i^{th} column and $\boldsymbol{A}[j]$ denote its j^{th} row. and let $\boldsymbol{A}_{i,j}$ be the (i,j)-th entry of $\boldsymbol{A}$. We will denote by Inf a *very large positive number*. Also, let $\mathcal{N}(0,1)$ denote the standard normal distribution. We will use $\mathcal{P}_n$ to denote a the set of all $n \times n$ permutation matrices, *i.e.*, the set of all $n \times n$ binary matrices that are obtained by permuting the rows of an $n \times n$ identity matrix (denoted by $\boldsymbol{I}_n$). Let $\mathsf{round} : \mathbb{R} \to \mathbb{Z}$ denote a function that returns the closest integer to a given real input.

Let us further introduce a few definitions that will be used throughout the paper.

Definition 4.1. *For a particular entry $i \in [n]$, define $\mathcal{S}(i)$ to be the set of all unknown vectors whose i^{th} entry is non-zero.*

$$\mathcal{S}(i) := \{\boldsymbol{\beta^j}, j \in [\ell] \mid \boldsymbol{\beta^j}_i \neq 0\}$$

Definition 4.2. *For a particular query vector* $\boldsymbol{v}$*, define* $\mathsf{poscount}(\boldsymbol{v}), \mathsf{negcount}(\boldsymbol{v})$ *and* $\mathsf{nzcount}(\boldsymbol{v})$ *to be the number of unknown vectors that assign a positive, negative, and non-zero label to* $\boldsymbol{v}$ *respectively.*

$$\begin{aligned}
\mathsf{poscount}(\boldsymbol{v}) &:= |\{\boldsymbol{\beta^j} \mid \langle \boldsymbol{v}, \boldsymbol{\beta^j} \rangle > 0, j \in [\ell]\}| \\
\mathsf{negcount}(\boldsymbol{v}) &:= |\{\boldsymbol{\beta^j} \mid \langle \boldsymbol{v}, \boldsymbol{\beta^j} \rangle < 0, j \in [\ell]\}| \\
\mathsf{nzcount}(\boldsymbol{v}) &:= \mathsf{poscount}(\boldsymbol{v}) + \mathsf{negcount}(\boldsymbol{v}) \\
&= |\{\boldsymbol{\beta^j} \mid \langle \boldsymbol{v}, \boldsymbol{\beta^j} \rangle \neq 0, j \in [\ell]\}|.
\end{aligned}$$

Definition 4.3 (Gaussian query)**.** *A vector* $\boldsymbol{v} \in \mathbb{R}^n$ *is called a Gaussian query vector if each entry* $\boldsymbol{v}_i$ *of* $\boldsymbol{v}$ *is sampled independently from the standard Normal distribution,* $\mathcal{N}(0, 1)$*.*

Estimating the counts Now, we show how to accurately estimate each of the counts *i.e.*, $\mathsf{poscount}(\boldsymbol{v})$, $\mathsf{negcount}(\boldsymbol{v})$ and $\mathsf{nzcount}(\boldsymbol{v})$ with respect to any query vector $\boldsymbol{v}$, with high probability (see Algorithm 4.1).

The idea is to simply query the oracle with $\boldsymbol{v}$ and $-\boldsymbol{v}$ repeatedly and estimate the counts empirically using the responses of the oracle. Let T denote the number of times a fixed query vector $\boldsymbol{v}$ is repeatedly queried. We refer to this quantity as the *batchsize*. We now design estimators of each of the counts which equals the real counts with high probability. Let $\mathbb{E}_{\boldsymbol{\beta}}(\cdot)$ and $\Pr_{\boldsymbol{\beta}}(\cdot)$ denote the expectation and the probability respectively when $\boldsymbol{\beta}$ is chosen uniformly from the set of unknown vectors $\{\boldsymbol{\beta}^1, \boldsymbol{\beta}^2, \ldots, \boldsymbol{\beta}^\ell\}$.

We must have

$$\begin{aligned}
\mathbb{E}_{\boldsymbol{\beta}}[\mathsf{sign}(\langle \boldsymbol{v}, \boldsymbol{\beta} \rangle)] &= \mathbb{E}_{\boldsymbol{\beta}}[\mathbb{1}[\langle \boldsymbol{v}, \boldsymbol{\beta} \rangle \geq 0]] - \mathbb{E}_{\boldsymbol{\beta}}[\mathbb{1}[\langle \boldsymbol{v}, \boldsymbol{\beta} \rangle < 0]] \\
&= \Pr_{\boldsymbol{\beta}}\Big(\langle \boldsymbol{v}, \boldsymbol{\beta} \rangle \geq 0\Big) - \Pr_{\boldsymbol{\beta}}\Big(\langle \boldsymbol{v}, \boldsymbol{\beta} \rangle < 0\Big)
\end{aligned}$$

$$= \frac{1}{\ell} \cdot \sum_{i=1}^{\ell} \mathbb{1}[\langle \boldsymbol{v}, \boldsymbol{\beta}^i \rangle \geq 0] - \frac{1}{\ell} \cdot \sum_{i=1}^{\ell} \mathbb{1}[\langle \boldsymbol{v}, \boldsymbol{\beta}^i \rangle < 0].$$

Notice that

$$\mathbb{1}[\langle \boldsymbol{v}, \boldsymbol{\beta}^i \rangle \geq 0] - \mathbb{1}[\langle \boldsymbol{v}, \boldsymbol{\beta}^i \rangle < 0] = \mathbb{1}[\langle -\boldsymbol{v}, \boldsymbol{\beta}^i \rangle \geq 0] - \mathbb{1}[\langle -\boldsymbol{v}, \boldsymbol{\beta}^i \rangle < 0] \quad \text{if} \quad \langle \boldsymbol{v}, \boldsymbol{\beta}^i \rangle = 0$$

and

$$\mathbb{1}[\langle \boldsymbol{v}, \boldsymbol{\beta}^i \rangle \geq 0] - \mathbb{1}[\langle \boldsymbol{v}, \boldsymbol{\beta}^i \rangle < 0] = \mathbb{1}[\langle -\boldsymbol{v}, \boldsymbol{\beta}^i \rangle < 0] - \mathbb{1}[\langle -\boldsymbol{v}, \boldsymbol{\beta}^i \rangle \geq 0] \quad \text{if} \quad \langle \boldsymbol{v}, \boldsymbol{\beta}^i \rangle \neq 0.$$

Therefore, we must have

$$\frac{\mathbb{E}_{\boldsymbol{\beta}}[\mathsf{sign}(\langle \boldsymbol{v}, \boldsymbol{\beta} \rangle) + \mathsf{sign}(\langle -\boldsymbol{v}, \boldsymbol{\beta} \rangle)]}{2} = \frac{1}{\ell} \cdot \sum_{i=1}^{\ell} \mathbb{1}[\langle \boldsymbol{v}, \boldsymbol{\beta}^i \rangle = 0]$$

Suppose we query the oracle with the pair of query vectors $\boldsymbol{v}, -\boldsymbol{v}$ repeatedly for T times. Let us denote the the T responses from the oracle $\mathcal{O}$ by $y_1, y_2, \ldots, y_T$ and $z_1, z_2, \ldots, z_T$ corresponding to the query vectors $\boldsymbol{v}$ and $-\boldsymbol{v}$ respectively. Hence, we design the following estimator (denoted by $\hat{\mathsf{z}}$) to estimate the number of unknown vectors that have zero projection on the query vector $\boldsymbol{v}$ i.e. $\sum_{i=1}^{\ell} \mathbb{1}[\langle \boldsymbol{v}, \boldsymbol{\beta}^i \rangle = 0]$:

$$\hat{\mathsf{z}} \triangleq \mathsf{round}\Big(\frac{\ell \sum_{i=1}^{T} y_i + z_i}{2T}\Big)$$

where $\mathsf{round} : \mathbb{R} \to \mathbb{Z}$ denotes a function that returns the closest integer to a given real input. Again, we have

$$\mathbb{E}_{\boldsymbol{\beta}}[\mathbb{1}[\mathsf{sign}(\langle \boldsymbol{v}, \boldsymbol{\beta} \rangle) = -1]] = \Pr_{\boldsymbol{\beta}}[\mathsf{sign}(\langle \boldsymbol{v}, \boldsymbol{\beta} \rangle < 0] = \frac{1}{\ell} \cdot \sum_{i=1}^{\ell} \mathbb{1}[\langle \boldsymbol{v}, \boldsymbol{\beta}^i \rangle < 0]$$

and therefore we design the estimator

$$\hat{\mathsf{neg}} \triangleq \mathsf{round}\Big(\frac{\ell \sum_{i=1}^{T} \mathbb{1}[y_i = -1]}{T}\Big)$$

of $\mathsf{negcount}(\boldsymbol{v})$. Subsequently, let $\hat{\mathsf{nz}} \triangleq \ell - \hat{\mathsf{z}}$ and $\hat{\mathsf{pos}} \triangleq \hat{\mathsf{nz}} - \hat{\mathsf{neg}}$ be the estimators of $\mathsf{nzcount}(\boldsymbol{v})$ and $\mathsf{poscount}(\boldsymbol{v})$ respectively.

Lemma 4.1. *For any query vector $\boldsymbol{v}$, Algorithm 4.1 with batchsize T provides the correct estimates of $\mathsf{poscount}(\boldsymbol{v})$, $\mathsf{negcount}(\boldsymbol{v})$ and $\mathsf{nzcount}(\boldsymbol{v})$ with probability at least $1 - 4e^{-T/2\ell^2}$.*

Proof. The proof of the lemma follows from a simple application of Chernoff bound.

Let $Z = \sum_i \mathbb{1}[y^i = -1]$, and therefore $\mathbb{E}Z = \frac{T \times \mathsf{negcount}}{\ell}$.

Note that Algorithm 4.1 makes a mistake in estimating $\mathsf{negcount}$ only if

$$|Z - \frac{T \times \mathsf{negcount}}{\ell}| \geq \frac{T}{2\ell}.$$

Since the responses in each batch are independent, using Chernoff bound [26], we get an upper bound on the probability that Algorithm 4.1 makes a mistake in estimating $\mathsf{negcount}$ as

$$\Pr\Big(|Z - \mathbb{E}Z| \geq \frac{T}{2\ell}\Big) \leq 2e^{-\frac{T}{2\ell^2}}.$$

Similarly, let $Z' = \frac{\sum_i y^i + z^i}{2}$, and therefore

$$\mathbb{E}Z' = \frac{T}{\ell} \cdot \sum_{i=1}^{\ell} \mathbb{1}[\langle \boldsymbol{x}, \boldsymbol{v}^i \rangle = 0].$$

Again, Algorithm 4.1 makes a mistake in estimating $\sum_{i=1}^{\ell} \mathbb{1}[\langle \boldsymbol{x}, \boldsymbol{v}^i \rangle = 0]$ only if

$$\frac{|Z' - \mathbb{E}Z'|}{T} \geq \frac{1}{2\ell}.$$

Using Chernoff bound [26] as before, the probability of making a mistake is bounded from above as

$$\Pr\left(|Z' - \mathbb{E}Z'| \geq \frac{T}{2\ell}\right) \leq 2e^{-\frac{T}{2\ell^2}}.$$

By taking a union bound, both $\hat{\mathsf{z}}, \hat{\mathsf{neg}}$ are computed correctly with probability at least $1 - 2e^{-\frac{T}{2\ell^2}}$. Finally, computing $\hat{\mathsf{z}}, \hat{\mathsf{neg}}$ correctly implies that $\hat{\mathsf{nz}}, \hat{\mathsf{pos}}$ are also correct thus proving our claim.

□

Algorithm 4.1 QUERY$(\boldsymbol{v}, T)$

Require: Query access to oracle $\mathcal{O}$.

1: **for** $i = 1, 2, \ldots, T$ **do**
2: Query the oracle with vector $\boldsymbol{v}$ and obtain response $y^i \in \{-1, +1\}$.
3: Query the oracle with vector $-\boldsymbol{v}$ and obtain response $z^i \in \{-1, +1\}$.
4: **end for**
5: Let $\hat{\mathsf{z}} := \mathsf{round}\left(\frac{\ell \sum_{i=1}^{T} y_i + z_i}{2T}\right)$.
6: Let $\hat{\mathsf{neg}} := \mathsf{round}\left(\frac{\ell \sum_i \mathbb{1}[y^i = -1]}{T}\right)$
7: Let $\hat{\mathsf{nz}} = \ell - \hat{\mathsf{z}}$ and $\hat{\mathsf{pos}} = \hat{\mathsf{nz}} - \hat{\mathsf{neg}}$
8: Return $\hat{\mathsf{pos}}, \hat{\mathsf{neg}}, \hat{\mathsf{nz}}$.

Family of sets: We will also use the families of sets introduced in Chapter 3 namely $(d, t, \alpha) - \mathsf{RUFF}$ and (r, t)-CFF. The $(2, t)$-CFF is of particular interest to us in this chapter and will henceforth be referred to as the *pairwise union free family* (PUFF). From Lemma 3.12 we know the existence of PUFF of size n with $m = O(t^3 \log n)$.

Corollary 4.4. *For any given integer t, there exists a $(2, t)$-CFF, $\mathcal{F}$ of size n with $m = O(t^3 \log n)$.*

4.4 Detailed Proofs and Algorithms (Support Recovery)

Recall the definition of $\mathsf{occ}(C, \boldsymbol{a})$ - the set of unknown vectors having $\boldsymbol{a} \in \{0, 1\}^{|C|}$ as a substring in coordinates $C \subset [n]$. First, we observe that for any set $\mathcal{T} \subseteq \{0, 1\}^s$,

we can compute $|\mathsf{occ}(C, \boldsymbol{a})|$ for all $O(n^s)$ subsets of s indices $C \subset [n]$ and $\boldsymbol{a} \in \mathcal{T}$ using queries.

Lemma 4.2. *Let $\mathcal{T} \subseteq \{0,1\}^s$ be any set of binary vectors of length s. There exists an algorithm to compute $|\mathsf{occ}(C, \boldsymbol{a})|$ for all $C \subset [n]$ of size s, and all $\boldsymbol{a} \in \mathcal{T}$ with probability at least $1 - 1/n$ using $O(\ell^3(\ell k)^{s+1} \log(\ell k n) \log n (1-2\eta)^2)$ queries.*

Given Lemma 4.2, the proofs of Theorems 4.1, 4.4 and 4.3 are identical to the proofs of Theorem 3.9, 3.12 and 3.11 presented in Chapter 3 respectively. Moreover, the proof of Lemma 4.2 is identical to the proof of Lemma 3.13.

Let $s < n$, then using queries constructed from CFFs of appropriate parameters we compute $|\bigcup_{i \in \mathcal{S}} \mathsf{occ}((i), 1)|$ for all subsets $\mathcal{S} \subset [n]$ of size s.

Lemma 4.3. *For any $1 < s < n$, there exists an algorithm to compute $\left|\bigcup_{i \in \mathcal{S}} \mathsf{occ}((i), 1)\right|$ for all $\mathcal{S} \subseteq [n], |\mathcal{S}| = s$ with probability at least $1 - O(n^{-2})$ using*

$$O(\ell^3(\ell k)^{s+1} \log(\ell k n) \log n / (1-2\eta)^2)$$

queries.

The proof of Lemma 4.3 follows from the guarantees of Algorithm 4.2. For the special case of $s = 1$, we use queries given by a RUFF of appropriate parameters to compute $|\mathsf{occ}((i), 1)|$ for all $i \in [n]$ using Algorithm 3.17 in Section 3.4.4.

Lemma 4.4. *There exists an algorithm to compute $|\mathsf{occ}((i), 1)|$ $\forall$ $i \in [n]$ with probability at least $1 - O(n^{-2})$ using $O(\ell^4 k^2 \log(\ell k n) \log n / (1-2\eta)^2)$ queries.*

Both the above mentioned algorithms crucially use a subroutine that counts the number of unknown vectors in $\mathcal{B}$ that have a non-zero inner product with a given query vector $\boldsymbol{x}$. For any $\boldsymbol{x} \in \mathbb{R}^n$, define $\mathsf{nzcount}(\boldsymbol{x}) := \sum_{i=1}^{L} \mathbb{1}[\langle \boldsymbol{\beta}^i, \boldsymbol{x} \rangle \neq 0]$.

Note that Lemma 4.1 (see Algorithm 4.1) provides the sufficient query complexity for computing $\mathsf{nzcount}(\boldsymbol{x})$ for any vector $\boldsymbol{x} \in \{0,1\}^n$.

Compute $\left|\bigcup_{i\in\mathcal{S}} \mathsf{occ}((i),1)\right|$ using Algorithm 4.2. In this section we present an algorithm to compute $\left|\bigcup_{i\in\mathcal{S}} \mathsf{occ}((i),1)\right|$, for every $\mathcal{S} \subseteq [n]$ of size $|\mathcal{S}| = s$, using $|\mathsf{occ}((i),1)|$ computed in the Section 3.4.4.

We will need an (s, Lk)-CFF for this purpose. Let $\mathcal{G} \equiv \{\mathcal{H}_1, \mathcal{H}_2, \ldots, \mathcal{H}_n\}$ be the required (s, Lk)-CFF of size n over alphabet $m = O((Lk)^{s+1} \log n)$. We construct a set of $L+1$ matrices $\mathcal{B} = \{\boldsymbol{B^{(1)}}, \ldots, \boldsymbol{B^{(L+1)}}\}$ where, each $\boldsymbol{B^{(w)}} \in \mathbb{R}^{m\times n}, w \in [L+1]$, is obtained from the (s, Lk)-CFF $\mathcal{G}$.

For the mixture of linear classifiers, we construct the sequence of matrices as follows: For every $(i, j) \in [m] \times [n]$, set $\boldsymbol{B}^{(w)}_{i,j}$ to be a random number sampled uniformly from $[0, 1]$ if $i \in H_j$, and 0 otherwise. We remark that the choice of uniform distribution in $[0, 1]$ is arbitrary, and any continuous distribution works. Since every $\boldsymbol{B^{(w)}}$ is generated identically, they have the exact same support, though the non-zero entries are different. Also, by definition, the support of the columns of every $\boldsymbol{B^{(w)}}$ corresponds to the sets in $\mathcal{G}$.

Let $\mathcal{U} := \cup_{i\in[L]} \mathsf{supp}(\boldsymbol{\beta}^i)$ denote the union of supports of all the unknown vectors. Since each unknown vector is k-sparse, it follows that $|\mathcal{U}| \leq Lk$. From the properties of (s, Lk)-CFF, we know that for any tuple of s indices $(i_1, i_2, \ldots, i_s) \subset \mathcal{U}$, the set $(\bigcap_{t=1}^{s} \mathcal{H}_{i_i}) \setminus \bigcup_{q\in\mathcal{U}\setminus\{i_1,i_2,\ldots,i_s\}} \mathcal{H}_q$ is non-empty. This implies that for every $w \in [L+1]$, there exists at least one row of $\boldsymbol{B^{(w)}}$ that has a non-zero entry in the $i_1^{\mathsf{th}}, i_2^{\mathsf{th}}, \ldots, i_s^{\mathsf{th}}$ index, and 0 in all other indices $p \in \mathcal{U} \setminus \{i_1, i_2, \ldots, i_s\}$. In Algorithm 4.2 we use these rows as queries to estimate their $\mathsf{nzcount}$. In Lemma 4.3, we show that this estimated quantity is exactly $|\bigcup_{j=1}^{s} \mathsf{occ}((i),1)|$ for that particular tuple $(i_1, i_2, \ldots, i_s) \subset \mathcal{U}$.

The rest of the proofs of Lemma 4.3 and Lemma 4.4 is identical to the proof of Lemma 3.14 and Lemma 3.15 respectively with T_R replaced by T_C.

Computing $|\mathsf{occ}((i),1)|$ In this section, we show how to compute $|\mathsf{occ}(i,1)|$ for every index $i \in [n]$.

Algorithm 4.2 RECOVER UNION- $\left|\bigcup_{i\in\mathcal{S}} \mathsf{occ}((i),1)\right|$ for all $\mathcal{S}\subseteq[n], |\mathcal{S}|=s, s\geq 2$.

Require: $|\mathsf{occ}((i),1)|$ for every $i\in[n]$. $s\geq 2$.
Require: Construct $\boldsymbol{B}\in\mathbb{R}^{m\times n}$ from (s,Lk)-CFF of size n over alphabet $m = c_3(Lk)^{s+1}\log n$.
1: Let $\mathcal{U} := \{i\in[n] \mid |\mathsf{occ}((i),1)|>0\}$
2: Let batchsize $T_C = 10L^2\log(nm)/(1-2\eta)^2$,
3: **for** every $p\in[m]$ **do**
4: Let $\mathsf{count}(p) := \max_{w\in[L+1]}\{\mathsf{nzcount}(\boldsymbol{B}^{(w)}[p])\}$ (obtained using Algorithm 4.1 with batchsize T_C).
5: **end for**
6: **for** every set $\mathcal{S}\subseteq[n]$ with $|\mathcal{S}|=s$ **do**
7: Let $p\in[m]$ such that $\boldsymbol{B}_{\boldsymbol{p},\boldsymbol{t}}\neq 0$ for all $t\in\mathcal{S}$, and $\boldsymbol{B}_{\boldsymbol{p},\boldsymbol{t'}}=0$ for all $q\in\mathcal{U}\setminus\mathcal{S}$.
8: Set $\left|\bigcup_{i\in\mathcal{S}}\mathsf{occ}((i),1)\right| = \mathsf{count}(p)$.
9: **end for**

Let $\mathcal{F}=\{\mathcal{H}_1,\mathcal{H}_2,\ldots,\mathcal{H}_n\}$ be a $(d,Lk,0.5)$-RUFF of size n over alphabet $[m]$. Construct the binary matrix $\boldsymbol{A}\in\{0,1\}^{m\times n}$ from $\mathcal{F}$, as $\boldsymbol{A}_{i,j}=1$ if and only if $i\in\mathcal{H}_j$. Each column $j\in[n]$ of $\boldsymbol{A}$ is essentially the indicator vector of the set $\mathcal{H}_j$.

We use the rows of matrix $\boldsymbol{A}$ as query vectors to compute $|\mathsf{occ}((i),1)|$ for each $i\in[n]$. For each such query vector $\boldsymbol{x}$, we compute the $\mathsf{nzcount}(\boldsymbol{x})$ using Algorithm 4.1 with batchsize T_C. We choose T_C to be sufficiently large to ensure that $\mathsf{nzcount}$ is correct for all the queries with very high probability.

For every $h\in\{0,\ldots,L\}$, let $\boldsymbol{b}^h\in\{0,1\}^m$ be the indicator of the queries that have $\mathsf{nzcount}$ at least h. We show in Lemma 3.15 that the set of columns of $\boldsymbol{A}$ that have large intersection with $\boldsymbol{b}^h$, exactly correspond to the indices $i\in[n]$ that satisfy $|\mathsf{occ}((i),1)|\geq h$. This allows us to recover $|\mathsf{occ}((i),1)|$ exactly for each $i\in[n]$.

4.5 Two-stage Approximate Recovery

In this section, we present the proof of Theorem 4.5. The two stage approximate recovery algorithm, as the name suggests, proceeds in two sequential steps. In the first stage, we recover the support of all the ℓ unknown vectors. In the second stage, we use

Algorithm 4.3 COMPUTE–$|\mathsf{occ}((i),1)|$

Require: Construct binary matrix $\boldsymbol{A} \in \{0,1\}^{m \times n}$ from $(d, Lk, 0.5) - \mathsf{RUFF}$ of size n over alphabet $[m]$, with $m = c_1 L^2 k^2 \log n$ and $d = c_2 Lk \log n$.

Initialize $\boldsymbol{b}^0, \boldsymbol{b}^1, \boldsymbol{b}^2, \ldots, \boldsymbol{b}^L$ to all zero vectors of dimension m.
Let batchsize $T_C = 4\ell^2 \log mn/(1-2\eta)^2$.
for $i = 1, \ldots, m$ **do**
  Set $w := \mathsf{nzcount}(\boldsymbol{A}[i])$
  (obtained using Algorithm 4.1 with batchsize T_C.)
  for $h = 0, 1, \ldots, w$ **do**
    Set $\boldsymbol{b}_i^h = 1$.
  end for
end for
for $h = 0, 1, \ldots, L$ **do**
  Set $\mathcal{C}_h = \{i \in [n] \mid |\mathsf{supp}(\boldsymbol{b}^h) \cap \mathsf{supp}(\boldsymbol{A}_i)| \geq 0.5d\}$.
end for
for $i = 1, 2, \ldots, n$ **do**
  Set $|\mathsf{occ}((i),1)| = h$ if $i \in \{\mathcal{C}_h \setminus \mathcal{C}_{h+1}\}$ for some $h \in \{0, 1, \ldots, L-1\}$.
  Set $|\mathsf{occ}((i),1)| = L$ if $i \in \mathcal{C}_L$
end for

these deduced supports to approximately recover the unknown vectors (Algorithm 4.4 described below.

Once we have the obtained the support of all unknown vectors, the task of approximate recovery can be achieved using a set of *Gaussian queries*. Recall from Definition 4.3, a Gaussian query refers to an oracle query with vector $\boldsymbol{v} = (\boldsymbol{v}_1, \ldots, \boldsymbol{v}_n) \in \mathbb{R}^n$ where each $\boldsymbol{v}_i$ is sampled independently from the standard Normal distribution, $\boldsymbol{v}_i \sim \mathcal{N}(0,1)$. The use of Gaussian queries in the context of 1-bit compressed sensing ($\ell = 1$) was studied by [85].

Lemma 4.5 ([85])**.** *For any $\epsilon > 0$, there exists an ϵ-recovery algorithm to efficiently recover an unknown vector in $\mathbb{R}^n$ using $O\left(\frac{n}{\epsilon} \log \frac{n}{\epsilon}\right)$ Gaussian queries.*

In the current query model however, the approximate recovery is a bit intricate since we do not possess the knowledge of the particular unknown vector that was sampled by the oracle. To circumvent this problem, we will leverage the special support structure of the unknown vectors. From Assumption 4.1, we know that every

unknown vector $\boldsymbol{\beta}^t, t \in [\ell]$, has at least one coordinate which is not contained in the support of the other unknown vectors. We will denote the first such coordinate by $\mathsf{rep}(\boldsymbol{\beta}^t)$. Define,

$$\mathsf{rep}(\boldsymbol{\beta}^t) := \mathsf{min}_p\{ \quad p \in \mathsf{supp}(\boldsymbol{\beta}^t) \setminus \bigcup_{q \in [\ell]\setminus\{t\}} \mathsf{supp}(\boldsymbol{\beta}^q)\} \in [n].$$

For ϵ-recovery of a fixed unknown vector $\boldsymbol{\beta}^t$, we will use the set of representative coordinates $\{\mathsf{rep}(\boldsymbol{\beta}^{t'})\}_{t' \neq t}$, to correctly identify its responses with respect to a set of Gaussian queries. In order to achieve this, we first have to recover the sign of $\boldsymbol{\beta}^t_{\mathsf{rep}(\boldsymbol{\beta}^t)}$ for every $t \in [\ell]$, using an RUFF, which is described in Algorithm 4.5.

Lemma 4.6. *Algorithm 4.5 recovers* $\mathsf{sign}(\boldsymbol{\beta}^t_{\mathsf{rep}(\boldsymbol{\beta}^t)})$ *for all* $t \in [\ell]$.

With the knowledge of all the supports, and the sign of every representative coordinate, we are now ready to prove Theorem 4.5. The details are presented in the Algorithm 4.4.

Proof of Theorem 4.5. For the ϵ-recovery of a fixed unknown vector $\boldsymbol{\beta}^t, t \in [\ell]$, we will generate its correct response with respect to a set of $\tilde{O}(k/\epsilon)$ Gaussian queries using *modified* Gaussian queries. A modified Gaussian query $\boldsymbol{v}^t$ for the t-th unknown vector, is a Gaussian query with a large positive entry in the coordinates indexed by $\mathsf{rep}(\boldsymbol{\beta}^{t'})$, for every $t' \neq t$.

Consider a fixed unknown vector $\boldsymbol{\beta}^t$. Let $\boldsymbol{v} \in \mathbb{R}^n$ be a Gaussian query, i.e., every entry of $\boldsymbol{v}$ is sampled independently from $\mathcal{N}(0,1)$. Algorithm 4.4 constructs a modified Gaussian query $\boldsymbol{v}^t$ from $\boldsymbol{v}$ as follows:

$$\boldsymbol{v}^t_j = \begin{cases} \mathsf{Inf} & \text{if} \quad j = \mathsf{rep}(\boldsymbol{\beta}^{t'}) \quad \text{for some } t' \neq t \\ \boldsymbol{v}_j & \text{otherwise} \end{cases}.$$

Algorithm 4.4 ϵ-RECOVERY, TWO STAGE

Require: Query access to oracle $\mathcal{O}$.
Require: Assumption 4.1 to be true.
Estimate $\mathsf{supp}(\boldsymbol{\beta}^t)$ for all $t \in [\ell]$.
Estimate $\mathsf{sign}(\boldsymbol{\beta}^t_{\mathsf{rep}(\boldsymbol{\beta}^t)})$ for all $t \in [\ell]$ using Algorithm 4.5.
Let Inf be a large positive number.
Let batchsize $T = 4\ell^2 \log(nk/\epsilon)$.
for $t = 1, \ldots, \ell$ **do**
  for $i = 1, \ldots, \tilde{O}(k/\epsilon)$ **do**
    Define $\boldsymbol{v}^t_j := \begin{cases} \mathsf{Inf} & \text{if } j = \mathsf{rep}(\boldsymbol{\beta}^{t'}), \text{for some } t' \neq t \\ \mathcal{N}(0,1) & \text{otherwise} \end{cases}$
    Obtain $\mathsf{poscount}(\boldsymbol{v}^t)$ using Algorithm 4.1 with batchsize T.
    Let $p_t := |\{t' \neq t \mid \mathsf{sign}(\boldsymbol{\beta}^{t'}_{\mathsf{rep}(\boldsymbol{\beta}^{t'})}) = +1\}|$
    if $\mathsf{poscount}(\boldsymbol{v}^t) \neq p_t$ **then**
      Set $y^t_i = +1$.
    else
      Set $y^t_i = -1$.
    end if
  end for
  From $\{y^t_1, y^t_2, \ldots, y^t_{\tilde{O}(k/\epsilon)}\}$, and $\mathsf{supp}(\boldsymbol{\beta}^t)$ recover $\hat{\boldsymbol{\beta}}^t$ by using Lemma 4.5.
end for
Return $\{\hat{\boldsymbol{\beta}}^t, t \in [\ell]\}$.

From construction, we know that $\boldsymbol{v}^t_j = \boldsymbol{v}_j$ for all $j \in \mathsf{supp}(\boldsymbol{\beta}^t)$. Therefore,

$$\langle \boldsymbol{v}^t, \boldsymbol{\beta}^t \rangle = \langle \boldsymbol{v}, \boldsymbol{\beta}^t \rangle \qquad \text{and therefore} \qquad \mathsf{sign}(\langle \boldsymbol{v}^t, \boldsymbol{\beta}^t \rangle) = \mathsf{sign}(\langle \boldsymbol{v}, \boldsymbol{\beta}^t \rangle).$$

On the other hand, if Inf is chosen to be large enough,

$$\mathsf{sign}(\langle \boldsymbol{v}^t, \boldsymbol{\beta}^{t'} \rangle) = \mathsf{sign}(\boldsymbol{\beta}^{t'}_{\mathsf{rep}(\boldsymbol{\beta}^{t'})}) \qquad \forall t' \neq t,$$

since $\mathsf{Inf} \cdot \boldsymbol{\beta}^{t'}_{\mathsf{rep}(\boldsymbol{\beta}^{t'})}$ dominates the sign of the inner product. Note that in order to obtain an upper bound on the value of Inf, we have to assume that the non-zero entries of every unknown vector have some non-negligible magnitude (at least $1/\mathrm{poly}(n)$).

Note that the $\mathsf{sign}(\boldsymbol{\beta}^{t'}_{\mathsf{rep}(\boldsymbol{\beta}^{t'})})$ was already computed using Algorithm 4.5, and therefore, the response of the modified Gaussian query with each $\boldsymbol{\beta}^{t'}, t' \neq t$ is known. Now if

$\mathsf{poscount}(\boldsymbol{v}^t)$ is different from the number of positive instances of $\mathsf{sign}(\boldsymbol{\beta}^{t'}_{\mathsf{rep}(\boldsymbol{\beta}^{t'})}), t' \neq t$, then it follows that $\mathsf{sign}(\langle \boldsymbol{v}^t, \boldsymbol{\beta}^t \rangle) = +1$. From this we can successfully obtain the response of $\boldsymbol{\beta}^t$ corresponding to a Gaussian query $\boldsymbol{v}$.

Algorithm 4.4 simulates $O(k/\epsilon \cdot \log(k/\epsilon))$ Gaussian queries for every $\boldsymbol{\beta}^t, t \in [\ell]$ using the modified Gaussian queries $\boldsymbol{v}^t$. Approximate recovery is then possible using Lemma 4.5 (restricted to the k-non zero coordinates in the $\mathsf{supp}(\boldsymbol{\beta}^t)$).

We now argue about the query complexity and the success probability of Algorithm 4.4.

For every unknown vector $\boldsymbol{\beta}^t, t \in [\ell]$, we simulate $O(k/\epsilon \cdot \log(k/\epsilon))$ Gaussian queries. Simulating each Gaussian query involves $T = O(\ell^2 \log(nk/\epsilon))$ oracle queries to estimate the $\mathsf{poscount}$. Note that Algorithm 4.5 can be run simultaneously with Algorithm 3.17 since they use the same set of queries. The sign recovery algorithm, therefore, does not increase the query complexity of approximate recovery. The total query complexity of Algorithm 4.4 after the support recovery procedure is at most $O\left((\ell^3 k/\epsilon) \log(nk/\epsilon) \log(k/\epsilon)\right)$.

From Lemma 4.1, each $\mathsf{poscount}$ is correct with probability at least $1 - O(\epsilon/(n^2k^2))$ and therefore by a union bound over all the $O(\ell k/\epsilon \cdot log(k/\epsilon))$ $\mathsf{poscount}$ estimates, the algorithm succeeds with probability at least $1 - O(1/n)$. □

Proof of Lemma 4.6. Consider the $(d, \ell k, 0.5) - \mathsf{RUFF}$, $\mathcal{F} = \{\mathcal{H}_1, \mathcal{H}_2, \ldots, \mathcal{H}_n\}$, of size n over alphabet $m = O(\ell^2 k^2 \log n)$ used in Algorithm 3.17. Let $\boldsymbol{A} \in \{0,1\}^{m \times n}$ be the binary matrix constructed from the RUFF in a similar manner, i.e., $\boldsymbol{A}_{i,j} = 1$ if and only if $i \in \mathcal{H}_j$. From the properties of RUFF, we know that for every $t \in [\ell]$, there exists a row (indexed by $i \in [m]$) of $\boldsymbol{A}$ such that $\boldsymbol{A}_{i,u(\boldsymbol{\beta}^t)} \neq 0$, and $\boldsymbol{A}_{i,j} = 0$ for all $j \in U \setminus \{u(\boldsymbol{\beta}^t)\}$, where, $U = \cup_{i \in [\ell]} \mathsf{supp}(\boldsymbol{\beta}^i)$. Therefore, the query with $\boldsymbol{A}[i]$ yields non-zero sign with only $\boldsymbol{\beta}^t$. Since,

$$\mathsf{sign}(\langle \boldsymbol{A}[i], \boldsymbol{\beta}^t \rangle) = \mathsf{sign}(\langle \boldsymbol{e}_{u(\boldsymbol{\beta}^t)}, \boldsymbol{\beta}^t \rangle) = \mathsf{sign}(\boldsymbol{\beta}^t_{u(\boldsymbol{\beta}^t)})$$

$\mathsf{sign}(\boldsymbol{\beta}^t_{u(\boldsymbol{\beta}^t)})$ can be deduced.

Algorithm 4.5 COMPUTE–$\mathsf{sign}(\boldsymbol{\beta}^t_{\mathsf{rep}(\boldsymbol{\beta}^t)})$

Require: Binary matrix $\boldsymbol{A} \in \{0,1\}^{m\times n}$ from $(d, \ell k, 0.5)-\mathsf{RUFF}$ of size n over alphabet $[m]$, with $m = O(\ell^2 k^2 \log n)$ and $d = O(\ell k \log n)$.
Require: $\mathsf{rep}(\boldsymbol{\beta}^t) \in [n]$ for all $t \in [\ell]$.
1: Let batchsize $T = 4\ell^2 \log mn$.
2: Let $U := \cup_{i\in[\ell]}\mathsf{supp}(\boldsymbol{\beta}^i)$.
3: **for** $t = 1, \ldots, \ell$ **do**
4: Let $i \in \{\mathsf{supp}(\boldsymbol{A}_{\mathsf{rep}(\boldsymbol{\beta}^t)}) \setminus \cup_{j\in U\setminus\{\mathsf{rep}(\boldsymbol{\beta}^t)\}}\mathsf{supp}(\boldsymbol{A}_j)\}$
5: **if** $\mathsf{poscount}(\boldsymbol{A}[i]) > 0$ (obtained using Algorithm 4.1 with batchsize T.) **then**
6: $\mathsf{sign}(\boldsymbol{\beta}^t_{\mathsf{rep}(\boldsymbol{\beta}^t)}) = +1$.
7: **else**
8: $\mathsf{sign}(\boldsymbol{\beta}^t_{\mathsf{rep}(\boldsymbol{\beta}^t)}) = -1$.
9: **end if**
10: **end for**

□

4.6 Single stage process for Approximate recovery

In this section, we present the proof of Theorem 4.6. Note that the approximate recovery procedure (Algorithm 4.4), described in Section 4.5, crucially utilizes the support information of every unknown vector to design its queries. This requirement forces the algorithm to proceed in two sequential stages.

In particular, Algorithm 4.4, with the knowledge of the support and the representative coordinates of all the unknown vectors, designed modified Gaussian queries that in turn simulated Gaussian queries for a fixed unknown vector. In this section, we achieve this by using the rows of a matrix obtained from an $(\ell, \ell k) - \mathsf{CFF}$. The property of the CFF allows us to simulate enough Gaussian queries for every unknown vector without the knowledge of their supports. This observation gives us a completely non-adaptive algorithm for approximate recovery of all the unknown vectors.

Consider a matrix $\boldsymbol{A}$ of dimension $m \times n$ constructed from an $(\ell, \ell k) - \mathsf{CFF}$, $\mathcal{F} = \{\mathcal{H}_1, \mathcal{H}_2, \ldots, \mathcal{H}_n\}$ of size n over alphabet m, as follows:

$$\boldsymbol{A}_{i,j} = \begin{cases} \mathsf{Inf} & \text{if } i \in \mathcal{H}_j \\ v \sim \mathcal{N}(0,1) & \text{otherwise} \end{cases}.$$

In Lemma 4.7, we show that for every unknown vector $\boldsymbol{\beta}^t$, there exists a row of $\boldsymbol{A}$ that simulates the Gaussian query for it. Therefore, using $\tilde{O}(k/\epsilon)$ independent blocks of such queries will ensure sufficient Gaussian queries for every unknown vector which then allows us to approximately recover these vectors.

Recall the definition of a representative coordinate of an unknown vector $\boldsymbol{\beta}^t$,

$$\mathsf{rep}(\boldsymbol{\beta}^t) := \mathsf{min}_p\{ \quad p \in \mathsf{supp}(\boldsymbol{\beta}^t) \setminus \bigcup_{q \in [\ell] \setminus \{t\}} \mathsf{supp}(\boldsymbol{\beta}^q)\} \in [n].$$

Lemma 4.7. *For every $t \in [\ell]$, there exists at least one row $\boldsymbol{v}^t$ in $\boldsymbol{A}$ that simulates a Gaussian query for $\boldsymbol{\beta}^t$, and $\mathsf{sign}(\langle \boldsymbol{v}^t, \boldsymbol{\beta}^{t'} \rangle) = \mathsf{sign}(\boldsymbol{\beta}^{t'}_{\mathsf{rep}(\boldsymbol{\beta}^{t'})})$ for all $t' \neq t$.*

Proof of Lemma 4.7. For any fixed $t \in [\ell]$, consider the set of indices

$$\mathcal{X} = \{\mathsf{rep}(\boldsymbol{\beta}^{t'}) \mid t' \in [\ell] \setminus \{t\}\}.$$

Recall that from the property of $(\ell, \ell k) - \mathsf{CFF}$, we must have

$$\bigcap_{j \in \mathcal{X}} \mathsf{supp}(\boldsymbol{A}_j) \not\subseteq \bigcup_{j \in \cup_{q \in [\ell]} \mathsf{supp}(\boldsymbol{\beta}^q) \setminus \mathcal{X}} \mathsf{supp}(\boldsymbol{A}_j).$$

Therefore, there must exist at least one row $\boldsymbol{v}^t$ in $\boldsymbol{A}$ which has a large positive entry, Inf, in all the coordinates indexed by $\mathcal{X}$. Moreover, $\boldsymbol{v}^t$ has a random Gaussian entry in all the other coordinates indexed by the union of support of all unknown vectors. Since $\boldsymbol{\beta}^t$ is 0 for all coordinates in $\mathcal{X}$, the query $\mathsf{sign}(\langle \boldsymbol{v}^t, \boldsymbol{\beta}^t \rangle)$ simulates a Gaussian query. Also,

$$\mathsf{sign}(\langle \boldsymbol{v}, \boldsymbol{\beta}^{t'} \rangle) = \mathsf{sign}(\mathsf{rep}(\boldsymbol{\beta}^{t'})) \qquad \forall t' \neq t$$

since $\mathsf{Inf} \times \boldsymbol{\beta}^{t'}_{\mathsf{rep}(\boldsymbol{\beta}^{t'})}$ dominates the inner product. □

We are now ready to present the completely non-adaptive algorithm for the approximate recovery of all the unknown vectors.

Algorithm 4.6 ϵ-RECOVERY, SINGLE STAGE

Require: Assumption 4.1 to be true.
Require: Binary matrix $\tilde{\boldsymbol{A}} \in \{0,1\}^{m \times n}$ from $(\ell, \ell k) - \mathsf{CFF}$ of size n over alphabet $m = O((\ell k)^{\ell+1} \log n)$.
Estimate $\mathsf{supp}(\boldsymbol{\beta}^t)$ and $\mathsf{sign}(\boldsymbol{\beta}^t_{\mathsf{rep}(\boldsymbol{\beta}^t)})$ for all $t \in [\ell]$ using Support Recovery Algorithm and Algorithm 4.5 respectively.
Set Inf to be a large positive number.
Set $\mathsf{D} = O(k/\epsilon \cdot \log(k/\epsilon))$.
Set batchsize $T = 4\ell^2 \log(mnk/\epsilon)$.
for $i = 1, \ldots, m$ **do**
  for $w = 1, 2, \ldots, \mathsf{D}$ **do**
    Construct query vector $\boldsymbol{v}$, where $\boldsymbol{v}_j = \begin{cases} \mathsf{Inf} & \text{if } \tilde{\boldsymbol{A}}_{i,j} = 1 \\ \mathcal{N}(0,1) & \text{otherwise} \end{cases}$.
    $\mathsf{Query}\Big(\boldsymbol{v}, T\Big)$ and set $\boldsymbol{P}_{i,w} = \mathsf{poscount}(\boldsymbol{v})$.
  end for
end for
for $t = 1, \ldots, \ell$ **do**
  Let $\mathcal{X} := \{\mathsf{rep}(\boldsymbol{\beta}^{t'}) \mid t' \in [\ell] \setminus t\}$ and $U := \cup_q \mathsf{supp}(\boldsymbol{\beta}^q)$
  Let $i \in \{\cap_{j \in \mathcal{X}} \mathsf{supp}(\tilde{\boldsymbol{A}}_j) \setminus \bigcup_{j \in U \setminus \mathcal{X}} \mathsf{supp}(\tilde{\boldsymbol{A}}_j)\} \subset [m]$.
  Let $p := |\{t' \neq t \mid \mathsf{sign}(\boldsymbol{\beta}^t_{\mathsf{rep}(\boldsymbol{\beta}^t)}) = +1\}|$
  for $w = 1, \ldots, \mathsf{D}$ **do**
    if $\boldsymbol{P}_{i,w} \neq p$ **then**
      Set $y^t_w = +1$
    else
      Set $y^t_w = -1$
    end if
  end for
  From $\{y^t_w \mid w \in [\mathsf{D}]\}$ and $\mathsf{supp}(\boldsymbol{\beta}^t)$ recover $\hat{\boldsymbol{\beta}}^t$ by using Lemma 4.5.
end for
Return $\{\hat{\boldsymbol{\beta}}^t \mid t \in [\ell]\}$.

Proof of Theorem 4.6. The proof of Theorem 4.6 follows from the guarantees of Algorithm 4.6. The query vectors of Algorithm 4.6 can be represented by the rows of the following matrix:

$$\boldsymbol{R} = \begin{bmatrix} \boldsymbol{A} \\ \tilde{\boldsymbol{A}} + \boldsymbol{B}^{(1)} \\ \tilde{\boldsymbol{A}} + \boldsymbol{B}^{(2)} \\ \vdots \\ \tilde{\boldsymbol{A}} + \boldsymbol{B}^{(\mathsf{D})} \end{bmatrix}$$

where, $\mathsf{D} = O(k/\epsilon \cdot \log k/\epsilon)$ and $\boldsymbol{A}$ is the matrix obtained from the $(d, \ell k, 0.5) - \mathsf{RUFF}$ required by the Support Recovery Algorithm and Algorithm 4.5. The matrix $\tilde{\boldsymbol{A}}$ is obtained from an $(\ell, \ell k) - \mathsf{CFF}$, $\mathcal{F} = \{\mathcal{H}_1, \mathcal{H}_2, \ldots, \mathcal{H}_n\}$ by setting $\tilde{\boldsymbol{A}}_{i,j} = \mathsf{Inf}$ if $i \in \mathcal{H}_j$ and 0 otherwise, and each matrix $\boldsymbol{B}^{(w)}$ for $w \in [\mathsf{D}]$ is a Gaussian matrix with every entry $\boldsymbol{B}^{(w)}_{i,j}$ drawn uniformly at random from standard Normal distribution.

Algorithm 4.6 decides all its query vectors at the start and hence is completely non-adaptive. It first invokes the Support Recovery Algorithm and Algorithm 4.5 to recover the support and the sign of the representative coordinate of every unknown vector $\boldsymbol{\beta}^t$. Now using the queries from the rows of the matrix $\boldsymbol{R}$, the algorithm generates at least $\mathsf{D} = \tilde{O}(k/\epsilon)$ Gaussian queries for each unknown vector.

It follows from Lemma 4.7 that each matrix $\tilde{\boldsymbol{A}} + \boldsymbol{B}^{(w)}$, for $w \in [\mathsf{D}]$, contains at least one Gaussian query for every unknown vector. Therefore, in total, $\boldsymbol{R}$ contains at least $D = O(k/\epsilon \cdot \log k/\epsilon)$ Gaussian queries for every unknown vector $\boldsymbol{\beta}^t$. Using the responses of these Gaussian queries, we can then approximately recover every $\boldsymbol{\beta}^t$ using Lemma 4.5.

The total query complexity is therefore the sum of query complexities of support recovery process (which from Theorem 4.2 we know to be at most $O(\ell^6 k^3 \log(n) \log(\ell k n)))$, and the total number of queries needed to generate $O(k/\epsilon \cdot \log(k/\epsilon))$ Gaussian queries (which is mTD) for each unknown vector. Therefore the net query complexity is $O\Big((\ell^{\ell+3} k^{\ell+2}/\epsilon) \log n \log(k/\epsilon) \log(n/\epsilon))\Big)$. Each of the algorithms namely 4.5 and the

Gaussian query generation succeed with probability at least $1 - O(1/n)$, therefore from union bound, Algorithm 4.6 succeeds with probability at least $1 - O(1/n)$. $\square$

4.7 Relaxing Separability Assumption for two unknown vectors

In this section, we will circumvent the necessity for Assumption 4.1 when there are only two unknown vectors - $\{\boldsymbol{\beta}^1, \boldsymbol{\beta}^2\}$. We present a two-stage algorithm to approximately recover both the unknown vectors and present the proof of Theorem 4.7. In the first stage, the algorithm recovers the support of both the vectors, and then using the support information it approximately recovers the two vectors.

We would like to mention that if $\mathsf{supp}(\boldsymbol{\beta}^1) \neq \mathsf{supp}(\boldsymbol{\beta}^2)$, we do not need any further assumptions on the unknown vectors for their approximate recovery. However, if the two vectors have the exact same support, then we need to impose some mild assumptions in order to approximately recover the vectors.

We show that supports of both the unknown vectors can be inferred directly from $\{|\mathcal{S}(i)|\}_{i\in[n]}$ and $\{|\mathcal{S}(i) \cap \mathcal{S}(j)|\}_{i,j\in[n]}$.

Lemma 4.8. *There exists an algorithm to recover the support of any two k-sparse unknown vectors using $O(k^3 \log^2 n)$ oracle queries with probability at least $1 - O(1/n^2)$.*

Proof of Lemma 4.8. Consider Algorithm 4.7. The proof follows directly from Theorem 4.1 since the support matrix for two vectors must be 1-identifiable.

$\square$

Now, we present the approximate recovery algorithm. The queries are designed based on the supports of the two vectors.

We split the analysis in two parts. First, we consider the case when the two vectors have different supports, i.e. $\mathsf{supp}(\boldsymbol{\beta}^1) \neq \mathsf{supp}(\boldsymbol{\beta}^2)$. In this case, we use Lemma 4.9 to approximately recover the two vectors.

Algorithm 4.7 RECOVER–SUPPORT $\ell = 2$

Require: Access to oracle $\mathcal{O}$

Estimate $|S(i)|$ for every $i \in [n]$ using Algorithm 3.17.
Estimate $|S(i) \cap S(j)|$ for every $i, j \in [n]$ using Algorithm 3.13.
if $|S(i)| \in \{0, 2\}$ for all $i \in [n]$ **then**
    $\mathsf{supp}(\boldsymbol{\beta}^1) = \mathsf{supp}(\boldsymbol{\beta}^2) = \{i \in [n] || \mathcal{S}(i)| \neq 0\}$.
else
    Let $i_0 = \min\{i || S(i)| = 1\}$, and let $i_0 \in \mathsf{supp}(\boldsymbol{\beta}^1)$
    for $j \in [n] \setminus \{i_0\}$ **do**
        if $|S(j)| = 2$ **then**
            Add j to $\mathsf{supp}(\boldsymbol{\beta}^1)$, and $\mathsf{supp}(\boldsymbol{\beta}^2)$.
        else if $|S(j)| = 1$ and $|\mathcal{S}(i_0) \cap \mathcal{S}(j)| = 0$ **then**
            Add j to $\mathsf{supp}(\boldsymbol{\beta}^2)$.
        else if $|S(j)| = 1$ and $|\mathcal{S}(i_0) \cap \mathcal{S}(j)| = 1$ **then**
            Add j to $\mathsf{supp}(\boldsymbol{\beta}^1)$.
        end if
    end for
end if

Lemma 4.9. *If* $\mathsf{supp}(\boldsymbol{\beta}^1) \neq \mathsf{supp}(\boldsymbol{\beta}^2)$*, then there exists an algorithm for* ϵ*-approximate recovery of any two* k*-sparse unknown vectors using* $O\left(\frac{k}{\epsilon} \cdot \log(\frac{nk}{\epsilon})\right)$ *oracle queries with probability at least* $1 - O(1/n)$*.*

When the two vectors have the exact same support, we use a set of sub-Gaussian queries to recover the two vectors. This is slightly tricky, and our algorithms succeeds under some mild assumption on the two unknown vectors (Assumption 4.2).

Lemma 4.10. *If* $\mathsf{supp}(\boldsymbol{\beta}^1) = \mathsf{supp}(\boldsymbol{\beta}^2)$*, then there exists an algorithm for* ϵ*-approximate recovery of any two* k*-sparse unknown vectors using* $O(\frac{k^2}{\epsilon^4 \delta^2} \log^2(\frac{nk}{\delta}))$ *oracle queries with probability at least* $1 - O(1/n)$*.*

Algorithm 4.8 ϵ-APPROXIMATE-RECOVERY

Estimate $\mathsf{supp}(\boldsymbol{\beta}^1), \mathsf{supp}(\boldsymbol{\beta}^2)$ using Algorithm 4.7.
if $\mathsf{supp}(\boldsymbol{\beta}^1) \neq \mathsf{supp}(\boldsymbol{\beta}^2)$ **then**
    Return $\hat{\boldsymbol{\beta}}^1, \hat{\boldsymbol{\beta}}^2$ using Algorithm 4.9.
else
    Return $\hat{\boldsymbol{\beta}}^1, \hat{\boldsymbol{\beta}}^2$ using Algorithm 4.10.
end if

Proof of Theorem 4.7. The guarantees of Algorithm 4.8 prove Theorem 4.7. The total query complexity after support recovery is the maximum of the query complexities of Algorithm 4.9 and Algorithm 4.10, which is $O(\frac{k^2}{\epsilon\delta^2}\log^2(\frac{nk}{\delta}))$.

Moreover from Lemma 4.9 and Lemma 4.10, we know that both these algorithms succeed with a probability at least $1 - O(1/n)$, therefore, Algorithm 4.8 is also guaranteed to succeed with probability at least $1 - O(1/n)$. □

We now prove Lemma 4.9 and Lemma 4.10.

4.7.1 Case 1: Different Support

In this subsection, we assume that $\mathsf{supp}(\boldsymbol{\beta}^1) \neq \mathsf{supp}(\boldsymbol{\beta}^2)$.

Proof of Lemma 4.9. Consider a coordinate $p \in \mathsf{supp}(\boldsymbol{\beta}^1) \,\Delta\, \mathsf{supp}(\boldsymbol{\beta}^2)$, where Δ denotes the symmetric difference of the two support sets. Without loss of generality we can assume $p \in \mathsf{supp}(\boldsymbol{\beta}^1)$. We first identify the $\mathsf{sign}(\boldsymbol{\beta}^1_p)$ simply using the query vector $\boldsymbol{e}_p$. For the sake of simplicity let us assume $\mathsf{sign}(\boldsymbol{\beta}^1_p) = +1$.

We use two types of queries to recover the two unknown vectors. The *Type 1* queries are modified Gaussian queries, of the form $\boldsymbol{v} + \mathsf{Inf} \cdot \boldsymbol{e}_p$, where $\boldsymbol{v}$ is a Gaussian query vector. *Type 2* query is the plain Gaussian query $\boldsymbol{v}$.

Since $p \in \mathsf{supp}(\boldsymbol{\beta}^1) \setminus \mathsf{supp}(\boldsymbol{\beta}^2)$, the Type 1 queries will always have a positive response with the unknown vector $\boldsymbol{\beta}^1$. Moreover, they will simulate a Gaussian query with $\boldsymbol{\beta}^2$. Therefore from the responses of the oracle, we can correctly identify the response of $\boldsymbol{\beta}^2$ with a set of $O(k/\epsilon \cdot \log(k/\epsilon))$ Gaussian queries. Now, using Lemma 4.5, we can approximately recover it.

Now since the response of $\boldsymbol{\beta}^2$ with the Type 1 query $\boldsymbol{v}+\mathsf{Inf}\cdot\boldsymbol{e}_p$ and the corresponding Type 2 query $\boldsymbol{v}$, remains the same, we can also obtain correct responses of $\boldsymbol{\beta}^1$ with a set of $O(k/\epsilon \cdot \log(k/\epsilon))$ Gaussian queries. By invoking Lemma 4.5 again, we can approximately recover $\boldsymbol{\beta}^1$.

Algorithm 4.9 ϵ-APPROXIMATE-RECOVERY: CASE 1

Require: $\mathsf{supp}(\boldsymbol{\beta}^1) \neq \mathsf{supp}(\boldsymbol{\beta}^2)$

Set $m = O(k/\epsilon \cdot \log(k/\epsilon))$

Set batchsize $T = 10 \log mn$.

Let Inf be a large positive number.

Let $p \in \mathsf{supp}(\boldsymbol{\beta}^1) \setminus \mathsf{supp}(\boldsymbol{\beta}^2)$, and $s := \mathsf{sign}(\boldsymbol{\beta}^1_p)$.

for $i = 1, \ldots, m$ **do**

  Construct query vector $\boldsymbol{v}$, where $\boldsymbol{v}_j = \mathcal{N}(0,1)$ for all $j \in [n]$.

  Construct query vector $\tilde{\boldsymbol{v}} := \boldsymbol{v} + s \cdot \mathsf{Inf} \cdot \boldsymbol{e}_p$

  $\mathsf{Query}\Big(\boldsymbol{v}, T\Big)$, and $\mathsf{Query}\Big(\tilde{\boldsymbol{v}}, T\Big)$.

  Set $y^i = \begin{cases} +1 & \text{if } \mathsf{poscount}(\tilde{\boldsymbol{v}}) == 2 \\ -1 & \text{if } \mathsf{negcount}(\tilde{\boldsymbol{v}}) == 1 \\ 0 & \text{otherwise} \end{cases}$

  Set $z^i = \begin{cases} +1 & \text{if } y^i = +1 \text{ and } \mathsf{poscount}(\boldsymbol{v}) == 2 \\ -1 & \text{if } y^i = +1 \text{ and } \mathsf{negcount}(\boldsymbol{v}) == 1 \\ +1 & \text{if } y^i = -1 \text{ and } \mathsf{poscount}(\boldsymbol{v}) == 1 \\ -1 & \text{if } y^i = -1 \text{ and } \mathsf{negcount}(\boldsymbol{v}) == 2 \\ +1 & \text{if } y^i = 0 \text{ and } \mathsf{poscount}(\boldsymbol{v}) == 1 \\ -1 & \text{if } y^i = 0 \text{ and } \mathsf{negcount}(\boldsymbol{v}) == 1 \\ 0 & \text{otherwise} \end{cases}$

end for

From $\{y^i \mid i \in [m]\}$ and $\mathsf{supp}(\boldsymbol{\beta}^2)$ recover $\hat{\boldsymbol{\beta}}^2$ by using Lemma 4.5.

From $\{z^i \mid i \in [m]\}$ and $\mathsf{supp}(\boldsymbol{\beta}^1)$ recover $\hat{\boldsymbol{\beta}}^1$ by using Lemma 4.5.

The total query complexity of the algorithm is $O(kT/\epsilon \cdot \log(k/\epsilon)) = O(k/\epsilon \cdot \log(nk/\epsilon) \cdot \log(k/\epsilon))$. Also, from Lemma 4.1, it follows that each oracle query succeeds with probability at least $1 - O(1/mn)$. Therefore by union bound over all $2m$ queries, the algorithm succeeds with probability at least $1 - O(1/n)$. □

4.7.2 Case 2: Same Support

We now propose an algorithm for approximate recovery of the two unknown vectors when their supports are exactly the same i.e. $\mathsf{supp}(\boldsymbol{\beta}^1) = \mathsf{supp}(\boldsymbol{\beta}^2)$. Until now for ϵ-recovery, we were using a representative coordinate to generate enough responses to Gaussian queries. However, when the supports are exactly the same, the same trick does not work.

For the approximate recovery in this case, we use sub-Gaussian queries instead of Gaussian queries. In particular, we consider queries whose entries are sampled uniformly from $\{-1, 1\}$. The equivalent of Lemma 4.5 proved by [4] for sub-Gaussian queries enables us to achieve similar bounds.

Lemma 4.11 (Corollary of Theorem 1.1 of [4]). *Let $\boldsymbol{x} \in \mathbb{S}^{n-1}$ be a k-sparse unknown vector of unit norm. Let $\boldsymbol{v_1}, \ldots, \boldsymbol{v_m}$ be independent random vectors in $\mathbb{R}^n$ whose coordinates are drawn uniformly from $\{-1, 1\}$. There exists an algorithm that recovers $\hat{\boldsymbol{x}} \in \mathbb{S}^{n-1}$ using the 1-bit sign measurements $\{\mathsf{sign}(\langle \boldsymbol{v_i}, \boldsymbol{x} \rangle)\}_{i \in [m]}$, such that with probability at least $1 - 4e^{-\alpha^2}$ (for any $\alpha > 0$), it satisfies*

$$\|\boldsymbol{x} - \hat{\boldsymbol{x}}\|_2^2 \leq O\left(\|x\|_\infty^{\frac{1}{2}} + \frac{1}{2\sqrt{m}} (\sqrt{k \log(2n/k)} + \alpha) \right).$$

In particular, for $m = O(\frac{k}{\epsilon^4} \log n)$, we get $O(\epsilon + \|x\|_\infty^{\frac{1}{2}})$ - approximate recovery with probability at least $1 - O(1/n)$. Therefore, if the unknown vectors are not *extremely* sparse (Assumption 4.2), we can get good guarantees on their approximate recovery with sufficient number of sub-Gaussian queries.

The central idea of ϵ-recovery algorithm (Algorithm 4.10) is therefore to identify the responses of a particular unknown vector $\boldsymbol{\beta}$ with respect to a set of sub-Gaussian queries $\boldsymbol{v} \sim \{-1, 1\}^n$. Then using Lemma 4.11, we can approximately reconstruct $\boldsymbol{\beta}$.

Let us denote by $\mathsf{response}(\boldsymbol{v})$, the set of distinct responses of the oracle with a query vector $\boldsymbol{v}$. Since there are only two unknown vectors, $|\mathsf{response}(\boldsymbol{v})| \leq 2$. If both unknown vectors have the same response with respect to a given query vector $\boldsymbol{v}$, i.e., $|\mathsf{response}(\boldsymbol{v})| = 1$ then we can trivially identify the correct responses with respect both the unknown vectors by setting $\mathsf{sign}(\langle \boldsymbol{v}, \boldsymbol{\beta}^2 \rangle) = \mathsf{sign}(\langle \boldsymbol{v}, \boldsymbol{\beta}^2 \rangle) = \mathsf{response}(\boldsymbol{v})$.

However if $|\mathsf{response}(\boldsymbol{v})| = 2$, we need to identify the correct response with respect to a fixed unknown vector. This *alignment* constitutes the main technical challenge in approximate recovery. To achieve this, Algorithm 4.10 fixes a pivot query say $\boldsymbol{v}_0$ with

$|\mathsf{response}(\boldsymbol{v}_0)| = 2$, and aligns all the other queries with respect to it by making some additional oracle queries.

Let W denote the set of queries such that $|\mathsf{response}(\boldsymbol{v})| = 2$. Also, for any pair of query vectors, $\boldsymbol{v}_1, \boldsymbol{v}_2 \in W$, we denote by $\mathsf{align}_{\boldsymbol{\beta}}(\boldsymbol{v}_1, \boldsymbol{v}_2)$ to be an ordered tuple of responses with respect to the unknown vector $\boldsymbol{\beta}$.

$$\mathsf{align}_{\boldsymbol{\beta}}(\boldsymbol{v}_1, \boldsymbol{v}_2) = (\mathsf{sign}(\langle \boldsymbol{v}_1, \boldsymbol{\beta} \rangle), \mathsf{sign}(\langle \boldsymbol{v}_2, \boldsymbol{\beta} \rangle)).$$

We fix a pivot query $\boldsymbol{v}_0 \in W$ to be one that satisfies $\mathsf{response}(\boldsymbol{v}_0) = \{-1, 1\}$. We can assume without loss of generality that there always exists one such query, otherwise all queries $\boldsymbol{v} \in W$ have $0 \in \mathsf{response}(\boldsymbol{v})$, and Proposition 4.1 aligns all such responses using $O(\log n)$ additional oracle queries.

Proposition 4.1. *Suppose for all queries $\boldsymbol{v} \in W$, $0 \in \mathsf{response}(\boldsymbol{v})$. There exists an algorithm that estimates $\mathsf{align}_{\boldsymbol{\beta}^1}(\boldsymbol{v}_0, \boldsymbol{v})$ and $\mathsf{align}_{\boldsymbol{\beta}^2}(\boldsymbol{v}_0, \boldsymbol{v})$ for any $\boldsymbol{v}, \boldsymbol{v}_0 \in W$ using $O(\log n)$ oracle queries with probability at least $1 - O(1/n)$.*

For a fixed pivot query $\boldsymbol{v}_0 \in W$ such that $\mathsf{response}(\boldsymbol{v}_0) = \{-1, 1\}$, Proposition 4.2 and Proposition 4.3 compute $\mathsf{align}_{\boldsymbol{\beta}}(\boldsymbol{v}_0, \boldsymbol{v})$ for all queries $\boldsymbol{v} \in W$ such that $0 \in \mathsf{response}(\boldsymbol{v})$ and $0 \notin \mathsf{response}(\boldsymbol{v})$ respectively.

Proposition 4.2. *Let $\boldsymbol{v}_0 \in W$ such that $\mathsf{response}(\boldsymbol{v}_0) = \{-1, 1\}$. For any query vector $\boldsymbol{v} \in W$ such that $0 \in \mathsf{response}(\boldsymbol{v})$, there exists an algorithm that computes $\mathsf{align}_{\boldsymbol{\beta}^1}(\boldsymbol{v}_0, \boldsymbol{v})$ and $\mathsf{align}_{\boldsymbol{\beta}^2}(\boldsymbol{v}_0, \boldsymbol{v})$ using $O(\log n)$ oracle queries with probability at least $1 - O(1/n)$.*

Proposition 4.3. *Let $\delta > 0$, be the largest real number such that $\boldsymbol{\beta}^1, \boldsymbol{\beta}^2 \in \delta\mathbb{Z}^n$. Let $\boldsymbol{v}_0 \in W$ such that $\mathsf{response}(\boldsymbol{v}_0) = \{-1, 1\}$. For any query vector $\boldsymbol{v} \in W$ such that $\mathsf{response}(\boldsymbol{v}) = \{-1, 1\}$, there exists an algorithm that computes $\mathsf{align}_{\boldsymbol{\beta}^1}(\boldsymbol{v}_0, \boldsymbol{v})$ and $\mathsf{align}_{\boldsymbol{\beta}^2}(\boldsymbol{v}_0, \boldsymbol{v})$ using $O(\frac{k}{\delta^2} \log(\frac{nk}{\delta}))$ oracle queries with probability at least $1 - O(1/n)$.*

Using the alignment process and Lemma 4.11, we can now approximately recover both the unknown vectors.

Proof of Lemma 4.10. Consider Algorithm 4.10, which basically collects enough responses of an unknown vector for a set of sub-Gaussian queries by aligning all responses.

Without loss of generality, we fix $\boldsymbol{v}_0$ such that $\mathsf{response}(\boldsymbol{v}_0) = \{+1, -1\}$, and also enforce that $\mathsf{sign}(\boldsymbol{v}_0, \boldsymbol{\beta}^1) = +1$. Now, we align all other responses with respect to $\boldsymbol{v}_0$. The proof of Lemma 4.10 then follows from the guarantees of Lemma 4.11. For $m = O(\frac{k}{\epsilon^4} \log n)$, along with the assumptions that $\|\boldsymbol{\beta^1}\|_\infty, \|\boldsymbol{\beta^2}\|_\infty = o(1)$, the algorithm approximately recovers $\boldsymbol{\beta^1}, \boldsymbol{\beta^2}$.

Algorithm 4.10 ϵ-Approximate Recovery: Case 2

Require: $\mathsf{supp}(\boldsymbol{\beta}^1) = \mathsf{supp}(\boldsymbol{\beta}^2)$, Assumption 4.2.

Set $m = O(\frac{k}{\epsilon^4} \log(n))$
Set batchsize $T = O(\log mn)$
for $i = 1, \ldots, m$ **do**
  Sample query vector $\boldsymbol{v}$ as: $\boldsymbol{v}_j = \begin{cases} +1 & \text{w.p. } 1/2 \\ -1 & \text{w.p. } 1/2 \end{cases}$
  $\mathsf{Query}(\boldsymbol{v}, T)$, and store $\mathsf{response}(\boldsymbol{v})$.
  if $|\mathsf{response}(\boldsymbol{v})| == 1$ **then**
    Set $y^{\boldsymbol{v}} = \mathsf{response}(\boldsymbol{v})$.
    Set $z^{\boldsymbol{v}} = \mathsf{response}(\boldsymbol{v})$.
  else
    Add $\boldsymbol{v}$ to W.
  end if
  Let $\boldsymbol{v}_0$ be an arbitrary $\boldsymbol{v} \in W$.
  for every $\boldsymbol{v} \in W$ **do**
    Set $(y^{\boldsymbol{v}_0}, y^{\boldsymbol{v}}) = \mathsf{align}_{\boldsymbol{\beta}^1}(\boldsymbol{v}_0, \boldsymbol{v})$.
    Set $(z^{\boldsymbol{v}_0}, z^{\boldsymbol{v}}) = \mathsf{align}_{\boldsymbol{\beta}^2}(\boldsymbol{v}_0, \boldsymbol{v})$.
  end for
end for
Using $\{y^{\boldsymbol{v}}\}_{\boldsymbol{v}}$, estimate $\hat{\boldsymbol{\beta}}^1$.
Using $\{z^{\boldsymbol{v}}\}_{\boldsymbol{v}}$, estimate $\hat{\boldsymbol{\beta}}^2$.

The number of queries made by Algorithm 4.10 is at most mT to generate responses and $O(m\frac{k}{\delta^2} \log(\frac{nk}{\delta}))$ to align all the m responses with respect to a fixed pivot query $\boldsymbol{v}_0$. Therefore the total query complexity of Algorithm 4.10 is $O(\frac{k^2}{\epsilon^4 \delta^2} \log^2(\frac{nk}{\delta}))$.

All parts of the algorithm succeed with probability at least $1 - O(1/n)$, and therefore the algorithm succeeds with probability at least $1 - O(1/n)$. □

Finally, we prove Proposition 4.1, Proposition 4.2 and Proposition 4.3.

Proof of Proposition 4.1. For the proof of Proposition 4.1, we simply use the query vector $\boldsymbol{v}_0 + \boldsymbol{v}$ to reveal whether the 0's in the two response sets correspond to the same unknown vector or different ones. The correctness of Algorithm 4.11 follows from the fact that there will be a 0 in the response set of $\boldsymbol{v}_0 + \boldsymbol{v}$ if and only if both the 0's correspond to the same unknown vector.

To obtain the complete response set for the query $\boldsymbol{v}_0 + \boldsymbol{v}$ with probability at least $1 - 1/n$, Algorithm 4.11 makes at most $O(\log n)$ queries.

Algorithm 4.11 ALIGN QUERIES, CASE 1

Require: $\boldsymbol{v}_0, \boldsymbol{v} \in \{-1, 1\}^n$, $0 \in \mathsf{response}(\boldsymbol{v}_0) \cap \mathsf{response}(\boldsymbol{v})$.
1: Set batchsize $T = O(\log n)$.
2: $\mathsf{Query}(\boldsymbol{v}_0 + \boldsymbol{v}, T)$.
3: **if** $0 \in \mathsf{response}(\boldsymbol{v}_0 + \boldsymbol{v})$ **then**
4: $\quad \mathsf{align}_{\boldsymbol{\beta}^1}(\boldsymbol{v}_0, \boldsymbol{v}) = (0, 0)$
5: $\quad \mathsf{align}_{\boldsymbol{\beta}^2}(\boldsymbol{v}_0, \boldsymbol{v}) = (\mathsf{response}(\boldsymbol{v}_0) \setminus \{0\}, \mathsf{response}(\boldsymbol{v}) \setminus \{0\})$
6: **else**
7: $\quad \mathsf{align}_{\boldsymbol{\beta}^1}(\boldsymbol{v}_0, \boldsymbol{v}) = (0, \mathsf{response}(\boldsymbol{v}) \setminus \{0\})$
8: $\quad \mathsf{align}_{\boldsymbol{\beta}^2}(\boldsymbol{v}_0, \boldsymbol{v}) = (\mathsf{response}(\boldsymbol{v}_0) \setminus \{0\}, 0)$
9: **end if**

□

Proof of Proposition 4.2. In this case, we observe that the response set corresponding to the query $\mathsf{Inf} \cdot \boldsymbol{v} + \boldsymbol{v}_0$ can reveal the correct alignment. To see this, let the response of $\boldsymbol{v}_0$ and $\boldsymbol{v}$ be $\{+1, -1\}$ and $\{\mathsf{s}, 0\}$ respectively for some $\mathsf{s} \in \{\pm 1\}$. The response set corresponding to $\mathsf{Inf} \cdot \boldsymbol{v} + \boldsymbol{v}_0$ will be the set (or multi-set) of the form $\{\mathsf{s}, \mathsf{t}\}$. Since we know $\mathsf{s} = \mathsf{response}(\boldsymbol{v}) \setminus \{0\}$, we can deduce t from the $\mathsf{poscount}(\mathsf{Inf} \cdot \boldsymbol{v} + \boldsymbol{v}_0)$, and $\mathsf{negcount}(\mathsf{Inf} \cdot \boldsymbol{v} + \boldsymbol{v}_0)$.

Now, if $\mathsf{t} = +1$, then $(+1, 0)$ are aligned together (response of the same unknown vector) and $(\mathsf{s}, -1)$ are aligned together. Similarly, if $\mathsf{t} = -1$, then $(-1, 0)$ and $(+1, \mathsf{s})$ are aligned together respectively.

The alignment algorithm is presented in Algorithm 4.12. It makes $O(\log n)$ queries and succeeds with probability at least $1 - 1/n$.

Algorithm 4.12 ALIGN QUERIES, CASE 2

Require: $\boldsymbol{v}_0, \boldsymbol{v} \in \{-1, 1\}^n$, $0 \in \mathsf{response}(\boldsymbol{v})$, $\mathsf{response}(\boldsymbol{v}_0) = \{\pm 1\}$.

Set batchsize $T = O(\log n)$.
Set Inf to be a large positive number.
$\mathsf{Query}(\boldsymbol{v}_0 + \mathsf{Inf} \cdot \boldsymbol{v}, T)$.
if $\mathsf{response}(\boldsymbol{v}_0 + \mathsf{Inf} \cdot \boldsymbol{v}) = \{\mathsf{response}(\boldsymbol{v}) \setminus \{0\}, +1\}$ **then**
  $\mathsf{align}_{\boldsymbol{\beta}^1}(\boldsymbol{v}_0, \boldsymbol{v}) = (+1, 0)$
  $\mathsf{align}_{\boldsymbol{\beta}^2}(\boldsymbol{v}_0, \boldsymbol{v}) = (-1, \mathsf{response}(\boldsymbol{v}) \setminus \{0\})$
else
  $\mathsf{align}_{\boldsymbol{\beta}^1}(\boldsymbol{v}_0, \boldsymbol{v}) = (+1, \mathsf{response}(\boldsymbol{v}) \setminus \{0\})$
  $\mathsf{align}_{\boldsymbol{\beta}^2}(\boldsymbol{v}_0, \boldsymbol{v}) = (-1, 0)$
end if

□

Proof of Proposition 4.3. The objective of Proposition 4.3 is to align the responses of queries $\boldsymbol{v}_0$ and $\boldsymbol{v}$ by identifying which among the following two hypotheses is true:

- $\mathbb{H}_1$: The response of the unknown vectors with both the query vectors $\boldsymbol{v}_0$ and $\boldsymbol{v}$ is same. Since we fixed the $\mathsf{sign}(\langle \boldsymbol{v}_0, \boldsymbol{\beta}^1 \rangle) = 1$, this corresponds to the case when $\mathsf{align}_{\boldsymbol{\beta}^1}(\boldsymbol{v}_0, \boldsymbol{v}) = (+1, +1)$ and $\mathsf{align}_{\boldsymbol{\beta}^1}(\boldsymbol{v}_0, \boldsymbol{v}) = (-1, -1)$.

  In this case, we observe that for any query of the form $\eta \boldsymbol{v}_0 + \zeta \boldsymbol{v}$ with $\eta, \zeta > 0$, the response set will remain $\{+1, -1\}$.

- $\mathbb{H}_2$: The response of each unknown vector with both the query vectors $\boldsymbol{v}_0$ and $\boldsymbol{v}$ is different, i.e., $\mathsf{align}_{\boldsymbol{\beta}^1}(\boldsymbol{v}_0, \boldsymbol{v}) = (+1, -1)$ and $\mathsf{align}_{\boldsymbol{\beta}^1}(\boldsymbol{v}_0, \boldsymbol{v}) = (-1, +1)$.

  In this case, we note that the response for the queries of the form $\eta \boldsymbol{v}_0 + \zeta \boldsymbol{v}$ changes from $\{-1, 1\}$ to either $\{+1\}, \{-1\}$, or $\{0\}$ for an appropriate choice of

$\eta, \zeta > 0$. In particular, the cardinality of the response set for queries of the form $\eta \boldsymbol{v}_0 + \zeta \boldsymbol{v}$ changes from 2 to 1 if $\frac{\eta}{\zeta} \in \left[-\frac{\langle \boldsymbol{\beta}^1, \boldsymbol{v}\rangle}{\langle \boldsymbol{\beta}^1, \boldsymbol{v}_0\rangle}, -\frac{\langle \boldsymbol{\beta}^2, \boldsymbol{v}\rangle}{\langle \boldsymbol{\beta}^2, \boldsymbol{v}_0\rangle}\right] \cup \left[-\frac{\langle \boldsymbol{\beta}^2, \boldsymbol{v}\rangle}{\langle \boldsymbol{\beta}^2, \boldsymbol{v}_0\rangle}, -\frac{\langle \boldsymbol{\beta}^1, \boldsymbol{v}\rangle}{\langle \boldsymbol{\beta}^1, \boldsymbol{v}_0\rangle}\right]$.

Algorithm 4.13 ALIGN QUERIES, CASE 3

Require: $\boldsymbol{v}_0, \boldsymbol{v} \in \{0, -1, 1\}^n$, $\mathsf{response}(\boldsymbol{v}) = \mathsf{response}(\boldsymbol{v}_0) = \{\pm 1\}$.
1: Set batchsize $T = O(\log nk/\delta)$.
2: **for** $\eta \in \{\frac{c}{d} \mid c, d \in \mathbb{Z} \setminus \{0\}, |c|, |d| \leq \frac{\sqrt{k}}{\delta}\}$ **do**
3: $\quad \mathsf{Query}(\eta \boldsymbol{v}_0 + \boldsymbol{v}, T)$.
4: $\quad$ **if** $|\mathsf{response}(\eta \boldsymbol{v}_0 + \boldsymbol{v})| == 1$ **then**
5: $\quad\quad$ Return $\mathsf{align}_{\boldsymbol{\beta}^1}(\boldsymbol{v}_0, \boldsymbol{v}) = (+1, -1)$, $\mathsf{align}_{\boldsymbol{\beta}^2}(\boldsymbol{v}_0, \boldsymbol{v}) = (-1, +1)$
6: $\quad$ **end if**
7: **end for**
8: Return $\mathsf{align}_{\boldsymbol{\beta}^1}(\boldsymbol{v}_0, \boldsymbol{v}) = (+1, +1)$, $\mathsf{align}_{\boldsymbol{\beta}^2}(\boldsymbol{v}_0, \boldsymbol{v}) = (-1, -1)$

In order to distinguish between these two hypotheses, Algorithm 4.13 makes sufficient queries of the form $\eta \boldsymbol{v}_0 + \zeta \boldsymbol{v}$ for varying values of $\eta, \zeta > 0$. If for some η, ζ the cardinality of the response set changes from 2 to 1, then we claim that $\mathbb{H}_2$ holds, otherwise $\mathbb{H}_1$ is true. Algorithm 4.13 then returns the appropriate alignment.

Note that for any query vector $\boldsymbol{v} \in \{-1, 1\}^n$, and any k-sparse unknown vector $\boldsymbol{\beta} \in \mathbb{S}^{n-1}$ the inner product $\langle \boldsymbol{\beta}, \boldsymbol{v}\rangle \in [-\sqrt{k}, \sqrt{k}]$. Moreover, if we assume that the unknown vectors have precision δ, the ratio $\frac{\langle \boldsymbol{\beta}^2, \boldsymbol{v}\rangle}{\langle \boldsymbol{\beta}^2, \boldsymbol{v}_0\rangle}$ can assume at most $4k/\delta^2$ distinct values. Algorithm 4.13 therefore iterates through all such possible values of η/ζ in order to decide which among the two hypothesis is true.

The total number of queries made by Algorithm 4.13 is therefore $4kT/\delta^2 = O(\frac{k}{\delta^2} \log(\frac{nk}{\delta}))$. From Lemma 4.1, all the responses are recovered correctly with probability $1 - O(1/n)$. □

4.8 Experiments

Similar to the mixed regression model, the problem of learning mixed linear classifiers can be used to model heterogenous data with categorical labels. We provide

some simulation results to show the efficacy of our proposed algorithms to reconstruct the component classifiers in the mixture.

Moreover, the algorithm suggested in this work can be used to learn the set of discriminative features of a group of people in a crowd sourcing model using simple queries with binary responses. Each person's preferences represents a sparse linear classifier, and the oracle queries here correspond to the crowdsourcing model. To exemplify this, we provide experimental results using the MovieLens [74] dataset to recover the movie genre preferences of two different users (that may use the same account, thus generating mixed responses) using small number of queries.

4.8.1 Simulations

We perform simulations that recover the support of $\ell = 2$, k-sparse vectors in $\mathbb{R}^n$ using Algorithm 4.7. We use random sparse matrices with sufficient number of rows to construct an RUFF. Error is measured in terms of relative hamming distance between the actual and the reconstructed support vectors.

The simulations show an improvement in the accuracy with increasing number of rows allocated to construct the RUFF for different values of $n = 1000, 2000, 3000$ with fixed $k = 5$. This is evident since the increasing number of rows improve the probability of getting an RUFF.

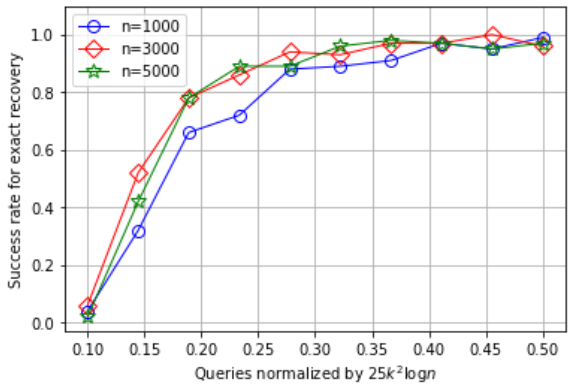


Figure 4.2: Support Recovery for $\ell = 2, k = 5$ and $n = 1000, 2000, 3000$.

4.8.2 Movie Lens

The MovieLens [74] database contains the user ratings for movies across various genres. Our goal in this set of experiments is to learn the movie genre preferences of two ($\ell = 2$) unknown users using a small set of commonly rated movies.

We first preprocess the set of all movies from the dataset to obtain a subset that have an average rating between 2.5 to 3.5. This is done to avoid biased data points that correspond to movies that are liked (or not liked at all) by almost everyone. For the rest of the experiment, we work with this pre-processed set of movies.

We consider $n = 20$ movie genres in some arbitrary, but predetermined order. The genre preference of each user i is depicted as an (unknown) indicator vector $\boldsymbol{\beta}^i \in \{0,1\}^n$, i.e., $\boldsymbol{\beta}^i_j = 1$ if and only if user i likes the movies in genre j. We assume that a user *likes* a particular movie if they rate it 3 or above. Also, we assume that the user likes a genre if they like at least half the movies they rated in a particular genre.

We consider two users, say U_1, U_2 who have commonly rated at least 500 movies. The preference vectors for both the users is obtained using Algorithm 4.7. We query the *oracle* with a movie, and obtain its rating from one of the two users at random. For the algorithm, we consider each query to correspond to the indicator of genres that the queried movie belongs to. Using small number of such randomly chosen movie queries, we show that Algorithm 4.7 approximately recovers the movie genre preference of both the users.

First, we pick a random subset of m movies that were rated by both the users, and partition them into two subsets of size m_1, and m_2 respectively. The first set of m_1 movies are used to partition the list of genres into three classes - genres liked by exactly one of the users, genres liked by both the users, and the genres liked by neither user. These set of m_1 randomly chosen movies essentially correspond to the rows of a RUFF used in Algorithm 4.7.

We then align the genres liked by exactly one of the users, we use the other set of m_2 randomly chosen movies and obtain two genre preference vectors $\boldsymbol{s_1}, \boldsymbol{s_2}$. Since we do not know whether $\boldsymbol{s_1}$ corresponds the preference vector of U_1 or U_2, we validate it against both, i.e., we validate $\boldsymbol{s_1}$ with U_1, $\boldsymbol{s_2}$ with U_2 and vice versa and select the permutation with higher average accuracy.

Validation: In order to validate our results, we use our recovered preference vectors to predict the movies that U_1 and U_2 will like. For each user U_i, we select the set of movies that were rated by U_i, but were not selected in the set of m movies used to recover their preference vector. The accuracy of our recovered preference vectors are measured by correctly predicting whether a user will like a particular movie from the test set.

Results: We obtain the accuracy, precision and recall for three random user pairs who have together rated at least 500 movies. The results show that our algorithm predicts the movie genre preferences of the user pair with high accuracy even with small m. Each of the quantities are obtained by averaging over 100 runs.

id: (U_1, U_2)	m_1	m_2	$\mathsf{A}(U_1)$	$\mathsf{P}(U_1)$	$\mathsf{R}(U_1)$	$\mathsf{A}(U_2)$	$\mathsf{P}(U_2)$	$\mathsf{R}(U_2)$
	0	0	0.300	0.000	0.000	0.435	0.000	0.000
(68, 448)	10	20	0.670	0.704	0.916	0.528	0.550	0.706
	30	60	0.678	0.700	0.944	0.533	0.548	0.791
	0	0	0.269	0.000	0.000	0.107	0.000	0.000
(274, 380)	10	20	0.686	0.733	0.902	0.851	0.893	0.946
	30	60	0.729	0.737	0.982	0.872	0.891	0.976
	0	0	0.250	0.000	0.000	0.197	0.000	0.000
(474, 606)	10	20	0.665	0.752	0.827	0.762	0.804	0.930
	30	60	0.703	0.750	0.910	0.787	0.806	0.970

CHAPTER 5

SUPPORT RECOVERY IN SPARSE MIXTURE MODELS

5.1 Introduction

Mixture models, that correspond to probabilistic modeling of subpopulations in data/observations, are widely used in practice to model heterogeneous data and have been studied theoretically for more than a century. Mixture models with a finite number of components have provided a mathematically rigorous approach to the statistical modeling of a wide variety of random phenomena. As finite mixture models are quite flexible and can be used to model complex data, they have been proved to be extremely useful for modeling data in many different fields including astrology, genetics, medicine, psychiatry, economics, marketing and engineering among many others [115]. Finite mixture models are especially successful in modeling data-sets having a group structure or the presence of a sub-population within the overall population. Often, finite mixture models can handle situations where a single parametric family cannot provide a satisfactory model for local variations in the observed data.

In this work, we investigate the support recovery problem in finite mixture models when the latent parameters of a mixture model are known to be sparse. Note that although sparsity is a very natural constraint on model parameters to design efficient algorithms for learning via reducing the effective dimension of the latent space, it has been relatively unexplored in mixture models. In particular, we study the support recovery problem in three different canonical mixture models namely mixtures of distributions (MD), mixtures of linear regressions (MLR) and mixtures of linear classifiers (MLC). We provide two flavors of results for support recovery namely 1)

Exact support recovery where we recover the support of all unknown sparse latent parameter corresponding to each component of the mixture 2) *Deduplicated Support recovery* where we recover the support of a crucial subset of latent parameters.

For *exact support recovery*, we use a general framework that was proposed in Chapter 3 for support recovery in MLR and MLC in the experimental design setting. A crucial difference between the setting in this chapter and the one studied in Chapter 3 is that in Chapter 3, the learning algorithm is able to design the covariates and assumes access to an oracle which can provide a noisy response corresponding to the designed oracle; on the other hand, for MLR and MLC, we assume that the covariates are sampled according to an isotropic Gaussian. Nevertheless, in Chapter 3, we showed that to recover the support exactly, it is sufficient to obtain correct estimates of the number of unknown vectors having non-zero entries at least one index in sets of small size. In this work, we extend this framework and provide non-trivial approaches to use this connection and compute the aforementioned sufficient statistic for MD, MLR and MLC settings. Similar to Chapter 3, we also introduce a general framework for *Deduplicated Support recovery* that we use to provide sample complexity guarantees in the MD and MLR settings.

5.1.1 Notations

We write $[n]$ to denote the set $\{1, 2, \ldots, n\}$. We will use $\mathbf{1}_n, \mathbf{0}_n$ to denote an all one vector and all zero vector of dimension n respectively. We will use $\mathcal{Q}([n])$ to denote the power set of $[n]$ i.e. $\mathcal{Q}([n]) = \{\mathcal{C} \mid \mathcal{C} \subseteq [n]\}$.

For any vector $\boldsymbol{v} \in \mathbb{R}^n$, we use $\boldsymbol{v}_i$ to denote the i^{th} coordinate of $\boldsymbol{v}$ and for any ordered set $\mathcal{S} \subseteq [n]$, we will use the notation $\boldsymbol{v}_{|\mathcal{S}} \in \mathbb{R}^{|\mathcal{S}|}$ to denote the vector $\boldsymbol{v}$ restricted to the indices in $\mathcal{S}$. Furthermore, we will use $\mathsf{supp}(\boldsymbol{v}) \triangleq \{i \in [n] : \boldsymbol{v}_i \neq 0\}$ to denote the support of $\boldsymbol{v}$ and $||\boldsymbol{v}||_0 \triangleq |\mathsf{supp}(\boldsymbol{v})|$ to denote the size of the support. Let $\mathsf{sign} : \mathbb{R} \to \{-1, +1\}$ be a function that returns the sign of a real number i.e. for

any input $x \in \mathbb{R}$,

$$\mathsf{sign}(x) = \begin{cases} 1 & \text{if } x \geq 0 \\ -1 & \text{if } x < 0 \end{cases}.$$

Consider a multi-set of n-dimensional vectors $\mathcal{U} \equiv \{\boldsymbol{u}^{(1)}, \boldsymbol{u}^{(2)}, \ldots, \boldsymbol{u}^{(L)}\}$. We will write $\mathcal{S}_{\mathcal{U}}(i) \triangleq \{\boldsymbol{u} \in \mathcal{U} : \boldsymbol{u}_i \neq 0\}$ to denote the multi-set of vectors in $\mathcal{U}$ that has a non-zero entry at the i^{th} index. Furthermore, for an ordered set $\mathcal{C} \subseteq [n]$ and vector $\boldsymbol{a} \in \{0,1\}^{|\mathcal{C}|}$, we will also write $\mathsf{occ}_{\mathcal{U}}(\mathcal{C}, \boldsymbol{a}) \triangleq \sum_{\boldsymbol{u} \in \mathcal{U}} \mathbf{1}[\boldsymbol{u}_{|\mathcal{C}} = \boldsymbol{a}]$ to denote the number of vectors in $\mathcal{U}$ that equal $\boldsymbol{a}$ when restricted to the indices in $\mathcal{C}$. For a matrix $\boldsymbol{M} \in \mathbb{R}^{m \times n}$, we will use $\boldsymbol{M}_i$ to denote the i^{th} column of $\boldsymbol{M}$. Let $\boldsymbol{A}_{\mathcal{U}} \in \{0,1\}^{n \times L}$ denote the support matrix of $\mathcal{U}$ where each column vector $\boldsymbol{A}_i \in \{0,1\}^n$ represents the support of the vector $\boldsymbol{u}^i \in \mathcal{U}$. For ease of notation, we will omit the subscript $\mathcal{U}$ when the set of vectors is clear from the context.

We write $\mathcal{N}(\mu, \sigma^2)$ to denote a Gaussian distribution with mean μ and variance σ^2. We will denote the cumulative distribution function of a random variable $Z \sim \mathcal{N}(0,1)$ by $\phi : \mathbb{R} \to [0,1]$ i.e. $\phi(a) = \int_{-\infty}^{a} p(z)dz$ where $p(\cdot)$ is the density function of Z. Also, we will denote $\mathsf{erf} : \mathbb{R} \to \mathbb{R}$ to be the error function defined by $\mathsf{erf}(z) = \frac{2}{\sqrt{\pi}} \int_0^z \exp(-t^2)dt$. Since the error function erf is bijective, we define $\mathsf{erf}^{-1}(\cdot)$ to be the inverse of the $\mathsf{erf}(\cdot)$ function. Finally, for a fixed set $\mathcal{B}$ we will write $X \sim_{\mathsf{Unif}} \mathcal{B}$ to denote a random variable X that is uniformly sampled from the elements in $\mathcal{B}$.

5.1.2 Formal Problem Statements

Let $\mathcal{V}$ be a multi-set of L unknown k-sparse vectors $\boldsymbol{v}^{(1)}, \boldsymbol{v}^{(2)}, \ldots, \boldsymbol{v}^{(L)} \in \mathbb{R}^n$ such that $\left|\left|\boldsymbol{v}^{(i)}\right|\right|_0 \leq k$ for all $i \in [L]$. We consider the following problems described below:

Mixtures of Distributions with Sparse Latent Parameters (MD): Consider a class of distributions $\mathcal{P} \equiv \{\boldsymbol{P}(\theta)\}_{\theta \in \Theta}$ parameterized by some $\theta \in \Theta$ where $\Theta \subseteq \mathbb{R}$.

We assume that all distributions in $\mathcal{P}$ satisfy the following property: $\mathbb{E}_{x\sim \boldsymbol{P}(\theta)} x^L$ can be written as a polynomial in θ of degree exactly L, From Table 2 in [18], we know that many well-known distributions satisfy this property (further discussion later). A sample $\boldsymbol{x} \sim \mathcal{P}_d$ is generated as follows:

$$t \sim_{\mathsf{Unif}} [L] \quad \text{and} \quad \boldsymbol{x}_i \mid t \sim \boldsymbol{P}(\boldsymbol{v}_i^{(t)}) \text{ independently for all } i \in [n].$$

In other words, $\boldsymbol{x}$ is generated according to a uniform mixture of distributions each having a sparse unknown parameter vector. Consider $\boldsymbol{x}^{(1)}, \boldsymbol{x}^{(2)}, \ldots, \boldsymbol{x}^{(m)} \in \mathbb{R}^n$, m i.i.d. copies of $\boldsymbol{x}$, that we can use to recover $\mathcal{V}$.

Here are some examples of this setting:

1. $\boldsymbol{P}(\theta)$ can be a Gaussian distribution with mean θ. This setting corresponds to a mixture of high-dimensional isotropic Gaussian distributions with sparse means.
2. $\boldsymbol{P}(\theta)$ can be a uniform distribution with range $[\theta, b]$ for a fixed and known b.
3. $\boldsymbol{P}(\theta)$ can be a Poisson distribution with mean θ.

Mixtures of Sparse Linear Regressions (MLR). Consider m samples

$$(\boldsymbol{x}^{(1)}, y^{(1)}), (\boldsymbol{x}^{(2)}, y^{(2)}), \ldots, (\boldsymbol{x}^{(m)}, y^{(m)}) \in \mathbb{R}^n \times \mathbb{R}$$

which are generated independently according to a distribution $\mathcal{P}_r$ defined as follows: for $(\boldsymbol{x}, y) \sim \mathcal{P}_r$, we have

$$\boldsymbol{x}_i \sim \mathcal{N}(0,1) \text{ independently for all } i \in [n]$$
$$\boldsymbol{v} \sim_{\mathsf{Unif}} \mathcal{V} \text{ and } y \mid \boldsymbol{x}, \boldsymbol{v} \sim \mathcal{N}(\langle \boldsymbol{v}, \boldsymbol{x} \rangle, \sigma^2).$$

In other words, each entry of $\boldsymbol{x}$ is sampled independently from $\mathcal{N}(0,1)$ and for a fixed $\boldsymbol{x}$, the conditional distribution of y given $\boldsymbol{x}$ is a Gaussian with mean $\langle \boldsymbol{v}, \boldsymbol{x} \rangle$ and known variance σ^2 where $\boldsymbol{v}$ is uniformly sampled from the multi-set $\mathcal{V}$.

Mixtures of Sparse Linear Classifiers (MLC). Consider m samples

$$(\boldsymbol{x}^{(1)}, y^{(1)}), (\boldsymbol{x}^{(2)}, y^{(2)}), \ldots, (\boldsymbol{x}^{(m)}, y^{(m)}) \in \mathbb{R}^n \times \{-1, +1\}$$

which are generated independently according to a distribution $\mathcal{P}_c$ defined as follows: for $(\boldsymbol{x}, y) \sim \mathcal{P}_c$, we have

$$\begin{aligned} &\boldsymbol{x}_i \sim \mathcal{N}(0,1) \text{ independently for all } i \in [n] \\ &\boldsymbol{v} \sim_{\mathsf{Unif}} \mathcal{V} \text{ and } z \sim \mathcal{N}(0, \sigma^2) \text{ and } y = \mathsf{sign}(\langle \boldsymbol{v}, \boldsymbol{x} \rangle + z). \end{aligned}$$

In other words, each entry of $\boldsymbol{x}$ is sampled independently from $\mathcal{N}(0,1)$ and for a fixed $\boldsymbol{x}$, the conditional distribution of y given $\boldsymbol{x}$ is $+1$ if $\langle \boldsymbol{v}, \boldsymbol{x} \rangle \geq -z$ and -1 otherwise; here, $\boldsymbol{v}$ is uniformly sampled from the multi-set of unknown vectors $\mathcal{V}$ and z denotes zero mean Gaussian noise with variance σ^2.

Our goal in all the three problems described above is to recover the support of unknown vectors $\boldsymbol{v}^{(1)}, \boldsymbol{v}^{(2)}, \ldots, \boldsymbol{v}^{(L)} \in \mathcal{V}$ with minimum number of samples m. More formally, we look at two distinct notions of support recovery:

Definition 5.1 (Exact Support Recovery)**.** *We will say that an algorithm achieves Exact Support Recovery in the MLC/MLR/MD setting if it can recover the support of all the unknown vectors in $\mathcal{V}$ exactly.*

Definition 5.2 (Deduplicated set)**.** *A deduplicated set $\mathcal{V}'$ is a subset of $\mathcal{V}$ such that 1) $\mathsf{supp}(\boldsymbol{v}^{(1)}) \not\subseteq \mathsf{supp}(\boldsymbol{v}^{(2)})$ for any distinct $\boldsymbol{v}^{(1)}, \boldsymbol{v}^{(2)} \in \mathcal{V}'$ and 2) $\boldsymbol{v} \notin \mathcal{V}'$ if there exists $\boldsymbol{v}' \in \mathcal{V}$ satisfying $\mathsf{supp}(\boldsymbol{v}) \subseteq \mathsf{supp}(\boldsymbol{v}')$. Now,*

$$\mathsf{Trimmed}(\mathcal{V}) \triangleq \mathsf{argmax}_{\mathcal{V}' \subseteq \mathcal{V}} |\mathcal{V}'| \tag{5.1}$$

where the maximization is over all deduplicated sets.

We can show that the set $\mathsf{Trimmed}(\mathcal{V})$ is unique (see Lemma C.1 in Appendix C.1).

Definition 5.3 (Deduplicated Support Recovery)**.** *We will say that an algorithm achieves Deduplicated support recovery in the MLR/MLC/MD setting if it can recover the support of all the unknown vectors in* $\mathsf{Trimmed}(\mathcal{V})$ *exactly.*

Note that in Definition 5.3, the objective is to recover supports of the largest set of vectors in $\mathcal{V}$, where no support is included completely in another support. This goal is easier than exact support recovery (Definition 5.1).

Remark 5.1. *If every unknown vector* $\boldsymbol{v} \in \mathcal{V}$ *had a unique non-zero index* $i \in [n]$ *i.e.* $\boldsymbol{v}_i \neq 0$ *and* $\boldsymbol{v}'_i = 0$ *for all* $\boldsymbol{v}' \in \mathcal{V} \setminus \{\boldsymbol{v}\}$*, then Deduplicated support recovery is equivalent to Exact Support Recovery. This condition, also known as the separability condition, has been commonly used in the literature for example in unique non-negative matrix factorization [8, 56, 129] and approximate parameter recovery in MLC in the query-based setting [68].*

5.1.3 Discussion on Our Results and Other Related Works

Mixtures of Distributions: Our technique of learning the supports of the latent parameter vectors in mixture of simple distributions is based on the *method of moments* [82, 72]. This method works in general, as long as moments of the distribution of each coordinate can be described as a polynomial in the component parameters. The authors in [18] showed (see Table 2 in [18]) that most common distributions, including Gaussian, Uniform, Poisson, and Laplace distributions, satisfy this assumption. Our results in this part that include sample complexity guarantees for both exact support recovery (see Theorem 5.1) and Deduplicated support recovery (see Theorem 5.2) are not only applicable to many canonical distributions but also makes progress towards quantifying the sufficient number of moments in the general problem defined in Sec. 5.1.2.

An alternate approach to the support recovery problem is to first recover the *union* of supports of the unknown parameters and then apply known parameter estimation guarantees to identify the support of each of the unknown vectors after reducing the dimension of the problem. Note that this approach crucially requires parameter estimation results for the corresponding family of mixtures which may be unavailable. To the best of our knowledge, most constructive sample complexity guarantees for parameter estimation in mixture models without separability assumptions correspond to mixtures of Gaussians [88, 18, 114, 72, 64, 77, 108, 76]. Moreover, most known results correspond to mixtures of Gaussians with two components. The only known results for parameter estimation in mixtures of Gaussians with more than 2 components is [114] but as we describe later, using the alternate approach with the guarantees in [114] results in a polynomial dependence on the sparsity. On the contrary, our sample complexity guarantees scales logarithmically with the sparsity or dimension (for constant L), see Corollary 5.3, which is a significant improvement over the alternate approach.

For other than Gaussian distributions, [18, 99] studied parameter estimation under the same moment-based assumption that we use. However, [18] uses non-constructive arguments from algebraic geometry because of which, their results did not include bounds on the sufficient number of moments for learning the parameters in a mixture model. In [99], the authors resolve this question to a certain extent for these aforementioned families of mixture models as they quantify the sufficient number of moments for parameter estimation under the restrictive assumption that the latent parameters lie on an integer lattice. Therefore, our results for these distributions form the first guarantees for support recovery.

Mixtures of Linear Regression For the support recovery problem in the sparse mixtures of linear regressions (MLR) setting, we provide a suite of results under different assumptions. In particular, we study the exact support recovery problem when the

unknown sparse parameters are binary (see Theorem 5.4) and the deduplicated support recovery problem when 1) the unknown sparse parameters have non-negative values (see Corollary 5.4), or 2) the unknown sparse parameters are distributed according to a Gaussian (see Corollary 5.5). As in the MD setting, an alternate approach for the support recovery problem is to first find the union of support of the unknown parameters and then apply existing parameter estimation guarantees to recovery the support of each of the unknown linear functions. The state of the art guarantees in MLR for parameter estimation is given by [103] providing a sample complexity guarantee which is linear in the dimension (linear in sparsity when restricted to the union of support). Our results for support recovery are polynomial in sparsity and are therefore worse than the parameter estimation guarantees of [103] applied to our sparse setting (see Theorem 5.5) when the sparsity is large. On the other hand, the sample complexity guarantees of [103] scales exponentially with L^2 and polynomially with the inverse of the failure probability. In contrast, our sample complexity guarantees are polynomial in L and logarithmic in the inverse of the failure probability.

Mixtures of Linear Classifiers Unlike the MLR and MD setting, mixture of linear classifiers (MLC) is far less studied. It is understandably more difficult to analyze than MLR since only the sign of the linear function of the covariates is retained. We study the exact support recovery problem in sparse MLC (see Theorem 5.3) under the setting that all the parameters of the unknown vectors are either nonnegative or they are all nonpositive. Although this assumption might seem restrictive, note that theoretical work in the MLC setting is extremely limited. To the best of our knowledge, there are only two relevant papers [134, 125] that have studied this problem. In [134], the authors do not make any assumptions on sparsity and provide an algorithm for recovering the subspace in which the parameter vectors corresponding to the unknown linear functions lie. In contrast, support recovery is a different objective and hence is incomparable to the subspace recovery guarantees. The second work, [125] uses

tensor decomposition based methods to provide sample complexity guarantees for learning the parameter vectors; but their sample complexity is inversely proportional to the square of the minimum eigenvalue of the matrix comprising the unknown parameter vectors as columns. This is an unwanted dependence as it implies that if the parameter vectors are linearly dependent, then the algorithm will require infinite samples to recover the parameter vectors. On the other hand, our support recovery guarantees do not have any such assumption on the parameters. Moreover, unlike the MD setting, it is not evident in MLC how to recover the union of support of the unknown sparse vectors. Hence the sample complexity obtained by applying the results in [125] directly will lead to a polynomial dependence on the dimension of the latent space which is undesirable (ideally, we require a logarithmic dependence on the latent space dimension). Our results exhibit such dependence on the dimension and also does not assume linear independence of the parameter vectors. We believe this to be an important progress towards further understanding of theoretical properties of mixtures where the response is a mixture of nonlinear functions of the covariates.

Main Technical Contribution beyond Chapter 3. As discussed earlier, our unsupervised setting is different from the query-based setting of Chapter 3, where the focus is also support recovery. However, we crucially use a general technique introduced in Chapter 3 (see Lemma 5.1) for exact support recovery. Namely, support recovery is possible if we can estimate some subset statistics.

But computing estimates of these subset statistics to invoke the guarantees given in Lemma 5.1 is a difficult problem. For the three settings, namely MD/MLR/MLC, we provide distinct and novel techniques to compute these quantities. Our approach to compute the sufficient statistics in MD setting involve a two-step approach with polynomial identities : 1) first, using the method of moments, we compute estimates of the power sum polynomial of degree p in the variables $\{\prod_{i\in\mathcal{C}} \boldsymbol{v}_i^2\}_{\boldsymbol{v}\in\mathcal{V}}$ for all subsets $\mathcal{C} \subset [n]$ up to a certain size; 2) secondly, we use an elegant connection via Newton's

identities to compute estimates on the elementary symmetric polynomial in the variables $\{\prod_{i\in\mathcal{C}} \boldsymbol{v}_i^2\}_{\boldsymbol{v}\in\mathcal{V}}$ which in turn allows us to compute the sufficient statistics. In MLR, for a set $\mathcal{C} \subseteq [n]$, we again analyze an interesting quantity namely $y^{|\mathcal{C}|} \cdot \Big(\prod_{i\in\mathcal{C}} \boldsymbol{x}_i \Big)$ that reveals the sufficient statistic for invoking Lemma 5.1. In MLC, our method is quite different and it involves conditioning on the event that certain coordinates of the covariate have large values. If this event is true, then analyzing the response variables reveals the sufficient statistics for invoking Lemma 5.1.

Organization: The rest of the paper is organized as follows: in Section 5.2, we provide the necessary preliminary lemmas for support recovery. In Section 5.3, we provide our main results and discuss our core approaches in each of the settings namely MD/MLR/MLC at a high level. For example, see Corollary 5.3, Theorem 5.5, and Theorem 5.3 for representative results in the three settings respectively. In Sections 5.4.1, 5.4.2 and 5.4.3, we provide the detailed proofs of all our results in the MD, MLC and MLR setting respectively. In Appendix C.1, we provide the missing proofs of lemmas in Section 5.2. In Appendix C.2, we provide some technical lemmas that are used in the main proofs.

5.2 Preliminaries: Useful Results in Subset Identification

The missing proofs of this section can be found in Appendix C.1.

To derive our support recovery results, we will crucially use the result of Lemma 5.1 below which has been proved in Chapter 3. Recall the definition of $\mathsf{occ}(\mathcal{C}, \boldsymbol{a})$ in Sec. 5.1.1. Lemma 5.1 states that if $\mathsf{occ}(\mathcal{C}, \boldsymbol{a})$ is known for all sets $\mathcal{C} \subseteq [n]$ up to a cardinality of $\log L + 1$, then it is possible to recover the support of all the unknown vectors in $\mathcal{V}$. We restate the result according to our terminology.

Lemma 5.1. *[Corollary 3.1 in Chapter 3] Let $\mathcal{V}$ be a set of L unknown vectors in $\mathbb{R}^n$. Then, if $\mathsf{occ}(\mathcal{C}, \boldsymbol{a})$ is provided as input for all sets $\mathcal{C} \subset [n], |\mathcal{C}| \leq \log L + 1$ and for all*

$\boldsymbol{a} \in \{0,1\}^{|\mathcal{C}|}$, *then there exists an algorithm (see Algorithm 3.13) that can recover the support of the unknown vectors in* $\mathcal{V}$.

Remark 5.2. *Lemma 5.1 provides an unconditional guarantee for recovering the support of the unknown vectors in* $\mathcal{V}$. *In other words, in the worst case, we only need to know* $\mathsf{occ}(\mathcal{C}, \boldsymbol{a})$ *for all sets of size* $|\mathcal{C}| \leq \log L + 1$. *However, in Chapter 3[Theorems 3.9, 3.10 and 3.11], significantly relaxed sufficient conditions for recovering the support of* $\mathcal{V}$ *under different structural assumptions were also provided. As noted in Chapter 3, these additional conditions are mild and in particular, the authors in Chapter 3 conjecture that if* $\mathsf{occ}(\mathcal{C}, \boldsymbol{a})$ *is known for all sets* $\mathcal{C} \subseteq [n]$ *up to a cardinality of* 3, *then it is possible to recover the support of all the unknown vectors in* $\mathcal{V}$.

Next, we describe another result, Lemma 5.2, proved in Chapter 3 that is also going to be useful for us. The main takeaway from Lemma 5.2 is that computing $|\cup_{i \in \mathcal{C}} \mathcal{S}(i)|$ (which represents the number of unknown vectors in $\mathcal{V}$ having non-zero values in at least one entry corresponding to $\mathcal{C}$) for all sets smaller than a fixed size (say t) is sufficient to compute $\mathsf{occ}(\mathcal{C}, \boldsymbol{a})$ for all subsets $\mathcal{C} \subseteq [n], |\mathcal{C}| \leq t$ and all vectors $\boldsymbol{a} \in \{0,1\}^{|\mathcal{C}|}$. However, we provide an additional result in Lemma 5.2 where we show that it is also possible to compute $\mathsf{occ}(\mathcal{C}, \boldsymbol{a})$ if the quantities $|\cap_{i \in \mathcal{C}} \mathcal{S}(i)|$ (which represents the number of unknown vectors in $\mathcal{V}$ having non-zero values in all entries corresponding to $\mathcal{C}$) are provided for all subsets $\mathcal{C} \subseteq [n]$ satisfying $|\mathcal{C}| \leq t$.

Lemma 5.2 (Partially proved in Chapter 3)**.** *Let* $\mathcal{V}$ *be a set of* L *unknown vectors in* $\mathbb{R}^n$. *If* $|\bigcup_{i \in \mathcal{C}} \mathcal{S}(i)|$ *is provided as input for all sets* $\mathcal{C} \subset [n], |\mathcal{C}| \leq t$ *or alternatively* $|\bigcap_{i \in \mathcal{C}} \mathcal{S}(i)|$ *is provided as input for all sets* $\mathcal{C} \subset [n], |\mathcal{C}| \leq t$, *then we can compute* $\mathsf{occ}(\mathcal{C}, \boldsymbol{a})$ *for all sets* $\mathcal{C} \subseteq [n], |\mathcal{C}| \leq t, \boldsymbol{a} \in \{0,1\}^{|\mathcal{C}|}$.

Corollary 5.1. *Let* $\mathcal{V}$ *be a set of* L *unknown* k*-sparse vectors in* $\mathbb{R}^n$. *Suppose, for each* $\mathcal{C} \subseteq [n], |\mathcal{C}| \leq \log L + 1$, *we can compute* $|\cup_{i \in \mathcal{C}} \mathcal{S}(i)|$ *(or alternatively* $|\bigcap_{i \in \mathcal{C}} \mathcal{S}(i)|$*) with probability* $1 - \gamma$ *using* $\mathsf{T} \log \gamma^{-1}$ *samples where* T *is independent of* γ. *Then,*

there exists an algorithm (see Algorithm 5.1) that can achieve Exact Support Recovery with probability at least $1 - \gamma$ *using* $O(\mathsf{T} \log(\gamma^{-1}(n + (Lk)^{\log L+1})))$ *samples.*

Proof. We know that all vectors $\boldsymbol{v} \in \mathcal{V}$ satisfy $\|\boldsymbol{v}\|_0 \leq k$ as they are k-sparse. Therefore, in the first stage, by computing $|\mathcal{S}(i)|$ for all $i \in [n]$, we can recover the union of support of all the unknown vectors $\cup_{\boldsymbol{v} \in \mathcal{V}} \mathsf{supp}(\boldsymbol{v})$ by computing $\mathcal{T} = \{i \in [n] \mid \mathcal{S}(i) > 0\}$. The probability of failure in finding the union of support exactly is at most $n\gamma$. Once we recover $\mathcal{T}$, we compute $|\cup_{i \in \mathcal{C}} \mathcal{S}(i)|$ for all $\mathcal{C} \subseteq \mathcal{T}, |\mathcal{C}| \leq \log L + 1$ (or alternatively $|\cap_{i \in \mathcal{C}} \mathcal{S}(i)|$ for all $\mathcal{C} \subseteq \mathcal{T}, |\mathcal{C}| \leq \log L + 1$). The probability of failure for this this event $(Lk)^{\log L+1}\gamma$. From Lemma 5.1, we know that computing $|\cup_{i \in \mathcal{C}} \mathcal{S}(i)|$ for all $\mathcal{C} \subseteq [n], |\mathcal{C}| \leq \log L + 1$ (or alternatively $|\cap_{i \in \mathcal{C}} \mathcal{S}(i)|$ for all $\mathcal{C} \subseteq \mathcal{T}, |\mathcal{C}| \leq \log L + 1$) exactly will allow us to recover the support of all the unknown vectors in $\mathcal{V}$. However $|\cup_{i \in \mathcal{C}} \mathcal{S}(i)| = 0$ for all $\mathcal{C} \subseteq [n] \setminus \mathcal{T}$ provided $\mathcal{T}$ is computed correctly. Therefore, we can recover the support of all the unknown vectors in $\mathcal{V}$ with $\mathsf{T} \log \gamma^{-1}$ samples with probability at least $1 - ((Lk)^{\log L+1} + n)\gamma$. Rewriting the previous statement so that the failure probability is γ leads to the statement of the lemma. □

Algorithm 5.1 EXACT SUPPORT RECOVERY USING ACCESS TO ESTIMATES OF $|\cap_{i \in \mathcal{C}} \mathcal{S}(i)|$ (OR ALTERNATIVELY $|\cup_{i \in \mathcal{C}} \mathcal{S}(i)|$) THAT ARE CORRECT WITH HIGH PROBABILITY

Require: For $\mathcal{C} \subseteq [n]$, access to estimates of $|\cap_{i \in \mathcal{C}} \mathcal{S}(i)|$ (or alternatively $|\cup_{i \in \mathcal{C}} \mathcal{S}(i)|$) that are correct with high probability.

1: For each $i \in [n]$, compute an estimate of $|\mathcal{S}(i)|$.
2: Compute $\mathcal{T} = \{i \in [n] \mid \text{estimate}(\mathcal{S}(i)) > 0\}$.
3: Compute estimates of $|\cap_{i \in \mathcal{C}} \mathcal{S}(i)|$ (or alternatively $|\cup_{i \in \mathcal{C}} \mathcal{S}(i)|$) for all subsets $\mathcal{C} \subseteq \mathcal{T}, |\mathcal{C}| \leq \log L + 1$.
4: Compute $\mathsf{occ}(\mathcal{C}, \boldsymbol{a})$ for all subsets $\mathcal{C} \subseteq \mathcal{T}, |\mathcal{C}| \leq \log L + 1, \boldsymbol{a} \in \{0, 1\}^{|\mathcal{C}|}$ using the computed estimates of $|\cap_{i \in \mathcal{C}} \mathcal{S}(i)|$ (or alternatively $|\cup_{i \in \mathcal{C}} \mathcal{S}(i)|$).
5: Use Algorithm 3.13 to recover the support of all unknown vectors in $\mathcal{V}$.

In the next few lemmas, we characterize the set $\mathsf{Trimmed}(\mathcal{V})$ and show some useful properties. We start with the following definition:

Definition 5.4 (t-good)**.** *For a binary matrix $\boldsymbol{A} \in \{0,1\}^{n \times L}$ with all distinct columns is called t-good if for every column $\boldsymbol{A}_i$, there exists a set $S \subset [n]$ of at most t-indices such that $\boldsymbol{A}_i|_S = \mathbf{1}_t$, and $\boldsymbol{A}_j|_S \neq \mathbf{1}_t$ for all $j \neq i$.*

Let $\mathcal{V}$ be set of L unknown vectors in $\mathbb{R}^n$, and $\boldsymbol{A} \in \{0,1\}^{n \times L}$ be its support matrix. Let $\boldsymbol{B}$ be the sub-matrix obtained by deleting duplicate columns of $\boldsymbol{A}$. The set $\mathcal{V}$ is called t-good if $\boldsymbol{B}$ is t-good.

Notice that if any set $\mathcal{V}$ is t-good then it must be r-good for all $r \geq t$. In Lemma 5.3, we show that $\mathsf{Trimmed}(\mathcal{V})$ is $(L-1)$-good and in Lemma 5.5, we provide sufficient conditions for deduplicated support recovery of the set of unknown vectors $\mathcal{V}$.

Lemma 5.3. *for all sets of L unknown vectors $\mathcal{V}$, $\mathsf{Trimmed}(\mathcal{V})$ must be $(L-1)$-good.*

Lemma 5.4. *If it is known whether $|\cap_{i \in \mathcal{C}} \mathcal{S}(i)| > 0$ or not for all sets $\mathcal{C} \subseteq [n], |\mathcal{C}| \leq s+1$, then there exists an algorithm that achieves Deduplicated support recovery of the set of unknown vectors $\mathcal{V}$ provided $\mathsf{Trimmed}(\mathcal{V})$ is known to be s-good for $s \leq L-1$ and $|\mathsf{Trimmed}(\mathcal{V})| \geq 2$.*

Algorithm 5.2 PARTIAL SUPPORT RECOVERY USING THE QUANTITIES $\mathbf{1}[|\cap_{i \in \mathcal{C}} \mathcal{S}(i)| > 0]$

Require: For every $\mathcal{C} \subseteq [n], |\mathcal{C}| \leq L$, the quantities $\mathbf{1}[|\cap_{i \in \mathcal{C}} \mathcal{S}(i)| > 0]$ are provided as input

Set $\mathcal{T} = \phi$
while There exists a set $\mathcal{C} \subseteq [n], |\mathcal{C}| \leq L-1$ such that $\boldsymbol{v}_{|\mathcal{C}} \neq \mathbf{1}_{|\mathcal{C}|}$ and $\mathbf{1}[|\cap_{i \in \mathcal{C}} \mathcal{S}(i)| > 0] = 1$. **do**
  Set $\mathcal{U} = \mathcal{C}$.
  for $j \in [n] \setminus \mathcal{C}$ **do**
    if $\mathbf{1}[\left|\cap_{i \in \mathcal{C} \cup \{j\}} \mathcal{S}(i)\right| > 0] = 1$ **then**
      Set $\mathcal{U} \leftarrow \mathcal{U} \cup \{j\}$
    end if
  end for
  Set $\mathcal{T} \leftarrow \mathcal{T} \cup \{\boldsymbol{v}\}$ where $\boldsymbol{v} \in \{0,1\}^n$ and $\mathsf{supp}(\boldsymbol{v}) = \mathcal{U}$.
end while
Return $\mathcal{T}$.

Algorithm 5.3 PARTIAL SUPPORT RECOVERY USING ACCESS TO ESTIMATES OF $\mathbf{1}[|\cap_{i\in\mathcal{C}}\mathcal{S}(i)|>0]$ THAT ARE CORRECT WITH HIGH PROBABILITY

Require: For $\mathcal{C}\subseteq[n]$, access to estimates of $\mathbf{1}[|\cap_{i\in\mathcal{C}}\mathcal{S}(i)|>0]$ that are correct with high probability.

1: For each $i\in[n]$, compute an estimate of $\mathbf{1}[|\cap_{i\in\mathcal{C}}\mathcal{S}(i)|>0]$.
2: Compute $\mathcal{T}=\{i\in[n]\mid \text{estimate}(\mathbf{1}[|\mathcal{S}(i)|>0]))=\text{True}\}$.
3: Compute estimates of $\mathbf{1}[|\cap_{i\in\mathcal{C}}\mathcal{S}(i)|>0]$ for all subsets $\mathcal{C}\subseteq\mathcal{T},|\mathcal{C}|\leq L$.
4: Use Algorithm 5.2 to recover the support of all unknown vectors in $\mathcal{V}$.

Lemma 5.5. *If it is known whether $|\cap_{i\in\mathcal{C}}\mathcal{S}(i)|>0$ or not for all sets $\mathcal{C}\subseteq[n],|\mathcal{C}|=L$, then there exists an algorithm (see Algorithm 5.2) that achieves Deduplicated support recovery of the set of unknown vectors $\mathcal{V}$.*

Corollary 5.2. *Let $\mathcal{V}$ be a set of L unknown k-sparse vectors in $\mathbb{R}^n$. Suppose with probability $1-\gamma$, for each $\mathcal{C}\subseteq[n],|\mathcal{C}|\leq L$, we can compute if $|\cap_{i\in\mathcal{C}}\mathcal{S}(i)|>0$ correctly with $\mathsf{T}\log\gamma^{-1}$ samples where T is independent of γ. Then, there exists an algorithm (see Algorithm 5.3) that can achieve Deduplicated support recovery with probability at least $1-\gamma$ using*

$$O(\mathsf{T}\log(\gamma^{-1}(n+(Lk)^L)))$$

samples.

Proof. Again, we know that all vectors $\boldsymbol{v}\in\mathcal{V}$ satisfy $\|\boldsymbol{v}\|_0\leq k$ as they are k-sparse. Therefore, in the first stage, by computing if $|\mathcal{S}(i)|>0$ for all $i\in[n]$, we can recover the union of support of all the unknown vectors $\cup_{\boldsymbol{v}\in\mathcal{V}}\mathsf{supp}(\boldsymbol{v})$ by computing $\mathcal{T}=\{i\in[n]\mid\mathcal{S}(i)>0\}$. The probability of failure in finding the union of support correctly is at most $n\gamma$. Once we recover $\mathcal{T}$ correctly, we compute $|\cap_{i\in\mathcal{C}}\mathcal{S}(i)|$ for all $\mathcal{C}\subseteq\mathcal{T},|\mathcal{C}|\leq L$. The probability of failure for this event $(Lk)^L\gamma$. From Lemma 5.5, we know that computing $|\cap_{i\in\mathcal{C}}\mathcal{S}(i)|$ for all $\mathcal{C}\subseteq[n],|\mathcal{C}|\leq L$ exactly will allow us to recover the support of all the unknown vectors in $\mathcal{V}$. On the other hand, we will have $|\cap_{i\in\mathcal{C}}\mathcal{S}(i)|=0$ for all $\mathcal{C}\subseteq[n]\setminus\mathcal{T}$ provided $\mathcal{T}$ is computed correctly. Therefore, we can

achieve deduplicated support recovery of all the unknown vectors in $\mathcal{V}$ with $\mathsf{T} \log \gamma^{-1}$ samples with probability at least $1 - ((Lk)^L + n)\gamma$. Rewriting, so that the failure probability is γ leads to the statement of the lemma. □

Remark 5.3. *Corollary 5.2 describes the sample complexity for deduplicated support recovery using Lemma 5.5 which provides the worst-case guarantees as* $\mathsf{Trimmed}(\mathcal{V})$ *is* $(L-1)-$*good for all sets* $\mathcal{V}$*. We can also provide improved guarantees for deduplicated support recovery provided* $\mathsf{Trimmed}(\mathcal{V})$ *is known to be* $s-$*good by using Lemma 5.4. However, for the sake of simplicity of exposition, we have only provided results for deduplicated support recovery in mixture models using Corollary 5.2.*

5.3 Our Results and Techniques

5.3.1 Mixtures of Distributions

In this section, we will present our main results and high level techniques in the MD setting. The detailed proofs of all results in this section can be found in Section 5.4.1. We will start by introducing some additional notations specifically for this setting.

Additional Notations for MD: Recall that $\mathbb{E}_{x \sim \boldsymbol{P}(\theta)} x^t$ can be written as a polynomial in θ of degree t. We will write

$$q_t(\theta) \triangleq \mathbb{E}_{x \sim \boldsymbol{P}(\theta)} x^t = \sum_{i \in [t+1]} \beta_{t,i} \theta^{i-1}$$

to denote this aforementioned polynomial where we use $\{\beta_{t,i}\}_{i \in [t+1]}$ to denote its coefficients. For all sets $\mathcal{A} \subseteq [n]$, we will write $\mathcal{Q}_i(\mathcal{A})$ to denote all subsets of $\mathcal{A}$ of size at most i i.e. $\mathcal{Q}_i(\mathcal{A}) = \{\mathcal{C} \mid \mathcal{C} \subseteq \mathcal{A}, |\mathcal{C}| \leq i\}$. Let us define the function $\pi : \mathcal{Q}([n]) \times [n] \to [n]$ denote a function that takes as input a set $\mathcal{C} \subseteq [n]$, an index $r \in \mathcal{C}$ and returns as output the position of r among all indices in $\mathcal{C}$ sorted in ascending

order. In other words, for a fixed set $\mathcal{C}$ and all $j \in [|\mathcal{C}|]$, $\pi(\mathcal{C}, \cdot)$ maps the j^{th} smallest index in $\mathcal{C}$ to j; for example, if $\mathcal{C} = \{3, 5, 9\}$, then $\pi(\mathcal{C}, 3) = 1, \pi(\mathcal{C}, 5) = 2$ and $\pi(\mathcal{C}, 9) = 3$.

We will write $\mathbb{Z}^+$ to denote the set of non-negative integers and $(\mathbb{Z}^+)^n$ to denote the set of all n-dimensional vectors having entries which are non-negative integers. For two vectors $\boldsymbol{u}, \boldsymbol{t} \in (\mathbb{Z}^+)^n$, we will write $\boldsymbol{u} \leq \boldsymbol{t}$ if $\boldsymbol{u}_i \leq \boldsymbol{t}_i$ for all $i \in [n]$; similarly, we will write $\boldsymbol{u} < \boldsymbol{t}$ if $\boldsymbol{u}_i < \boldsymbol{t}_i$ for some $i \in [n]$. For any fixed subset $\mathcal{C} \subseteq [n]$ and vectors $\boldsymbol{u}, \boldsymbol{t} \in (\mathbb{Z}^+)^{|\mathcal{C}|}$, we will write $\zeta_{\boldsymbol{t},\boldsymbol{u}}$ to denote the quantity

$$\zeta_{\boldsymbol{t},\boldsymbol{u}} \triangleq \prod_{i \in \mathcal{C}} \beta_{\boldsymbol{t}_{\pi(\mathcal{C},i)}, \boldsymbol{u}_{\pi(\mathcal{C},i)}+1}.$$

For any $\boldsymbol{u}, \boldsymbol{z} \in (\mathbb{Z}^+)^{|\mathcal{C}|}$ satisfying $\boldsymbol{u} < \boldsymbol{z}$, we will define a path M to be a sequence of vectors $\boldsymbol{z}_1 > \boldsymbol{z}_2 > \cdots > \boldsymbol{z}_m$ such that $\boldsymbol{z}_1, \boldsymbol{z}_2, \ldots, \boldsymbol{z}_m \in (\mathbb{Z}^+)^n$, $\boldsymbol{z}_1 = \boldsymbol{z}$ and $\boldsymbol{z}_m = \boldsymbol{u}$. Let $\mathcal{M}(\boldsymbol{z}, \boldsymbol{u})$ be the set of all paths starting from $\boldsymbol{z}$ and ending at $\boldsymbol{u}$. We will also write a path $\mathsf{M} \in \mathcal{M}(\boldsymbol{z}, \boldsymbol{u})$ uniquely as a set of $m-1$ ordered tuples $\{(\boldsymbol{z}_1, \boldsymbol{z}_2), (\boldsymbol{z}_2, \boldsymbol{z}_3), \ldots, (\boldsymbol{z}_{m-1}, \boldsymbol{z}_m)\}$ where each tuple consists of adjacent vectors in the path sequence. We will also use $\mathcal{T}(\mathsf{M}) \equiv \{\boldsymbol{z}_1, \boldsymbol{z}_2, \ldots, \boldsymbol{z}_m\}$ to denote the set of elements in the path.

We start with the following assumption which states that every unknown vector is bounded within an euclidean ball and furthermore, the magnitude of every non-zero co-ordinate of all unknown vectors is bounded from below:

Assumption 5.1. *We will assume that all unknown vectors in the set $\mathcal{V}$ are bounded within a ball of known radius R i.e. $\left|\left|\boldsymbol{v}^{(i)}\right|\right|_2 \leq R$ for all $i \in [L]$. Furthermore, the magnitude of all non-zero entries of all unknown vectors in $\mathcal{V}$ is bounded from below by δ i.e. $\min_{\boldsymbol{v} \in \mathcal{V}} \min_{i:\boldsymbol{v}_i \neq 0} |\boldsymbol{v}_i| \geq \delta$.*

Now, we show our main lemma in this setting where we characterize the sufficient number of samples to compute $\left|\bigcap_{i\in\mathcal{C}}\mathcal{S}(i)\right|$ for each set $\mathcal{C}\subseteq[n]$ with high probability in terms of the coefficients of the polynomials $\{q_t(\theta)\}_t$:

Lemma 5.6. *Suppose Assumption 5.1 is true. Let*

$$\Phi \triangleq \frac{\delta^{2L|\mathcal{C}|}}{2\Big(3\max(LR^{2L|\mathcal{C}|}, 2^L R^{L+|\mathcal{C}|})\Big)^{(L-1)} L!} \times \left(\max_{\boldsymbol{z}\leq 2L\mathbf{1}_{|\mathcal{C}|}} \frac{L}{\zeta_{\boldsymbol{z},\boldsymbol{z}}} + \sum_{\boldsymbol{u}<\boldsymbol{z}} \sum_{\mathsf{M}\in\mathcal{M}(\boldsymbol{z},\boldsymbol{u})} \frac{L\prod_{(\boldsymbol{r},\boldsymbol{s})\in\mathsf{M}}\zeta_{\boldsymbol{r},\boldsymbol{s}}}{\prod_{\boldsymbol{r}\in\mathcal{T}(\mathsf{M})}\zeta_{\boldsymbol{r},\boldsymbol{r}}}\right)^{-1}$$

$$g_{L,\mathcal{V}} \triangleq \frac{\max_{\boldsymbol{z}\leq 2L\mathbf{1}_{|\mathcal{C}|}} \mathbb{E}\prod_{i\in\mathcal{C}} \boldsymbol{x}_i^{2\boldsymbol{z}_{\pi(\mathcal{C},i)}}}{\Phi^2}$$

where $g_{L,\mathcal{V}}$ is a constant that is independent of k and n but depends on L. There exists an algorithm (see Algorithm 5.4) that can compute $\left|\bigcap_{i\in\mathcal{C}}\mathcal{S}(i)\right|$ exactly for each set $\mathcal{C}\subseteq[n]$ with probability at least $1-\gamma$ using $O\Big(\log(\gamma^{-1}(2L)^{|\mathcal{C}|})g_{L,\mathcal{V}}\Big)$ samples generated according to $\mathcal{P}_d$.

In order to prove Lemma 5.6, we first show that (see Lemma 5.10) for each fixed ordered set $\mathcal{C}\subseteq[n]$ and each vector $\boldsymbol{t}\in(\mathbb{Z}^+)^{|\mathcal{C}|}$, we must have

$$\mathbb{E}\prod_{i\in\mathcal{C}} \boldsymbol{x}_i^{\boldsymbol{t}_{\pi(\mathcal{C},i)}} = \frac{1}{L}\sum_{\boldsymbol{u}\leq\boldsymbol{t}} \zeta_{\boldsymbol{t},\boldsymbol{u}} \cdot \Big(\sum_{j\in[L]}\prod_{i\in\mathcal{C}} (\boldsymbol{v}_i^{(j)})^{\boldsymbol{u}_{\pi(\mathcal{C},i)}}\Big). \tag{5.2}$$

Note that each summand in equation 5.2 is a product of the powers of the coordinates of the same unknown vector. In Lemma 5.11, we show that for each set $\mathcal{C}\subseteq[n]$ and any vector $\boldsymbol{t}\in(\mathbb{Z}^+)^{|\mathcal{C}|}$, we can compute $\sum_{j\in[L]}\prod_{i\in\mathcal{C}}(\boldsymbol{v}_i^{(j)})^{\boldsymbol{t}_{\pi(\mathcal{C},i)}}$ via a recursive procedure provided for all $\boldsymbol{u}\in(\mathbb{Z}^+)^{|\mathcal{C}|}$ satisfying $\boldsymbol{u}\leq\boldsymbol{t}$, the quantity $\mathbb{E}\prod_{i\in\mathcal{C}}\boldsymbol{x}_i^{\boldsymbol{u}_{\pi(\mathcal{C},i)}}$ is pre-computed. This implies that we can compute $\sum_{j\in[L]}\prod_{i\in\mathcal{C}}(\boldsymbol{v}_i^{(j)})^{2p}$ for all $p\in[L]$ from the quantities $\mathbb{E}\prod_{i\in\mathcal{C}}\boldsymbol{x}_i^{\boldsymbol{u}_{\pi(\mathcal{C},i)}}$ for all $\boldsymbol{u}\leq 2p\mathbf{1}_{|\mathcal{C}|}$. It is easy to recognize $\sum_{\boldsymbol{v}\in\mathcal{V}}\Big(\prod_{i\in\mathcal{C}}\boldsymbol{v}_i^2\Big)^p$ as the power sum polynomial of degree p in the variables

$\{\prod_{i\in\mathcal{C}} \boldsymbol{v}_i^2\}_{\boldsymbol{v}\in\mathcal{V}}$. Now, let us define the quantity $\mathsf{A}_{\mathcal{C},t}$ for a fixed ordered set $\mathcal{C}$ and parameter $t \in [L]$ as follows:

$$\mathsf{A}_{\mathcal{C},t} \triangleq \sum_{\substack{\mathcal{C}'\subseteq[L]\\|\mathcal{C}'|=t}} \prod_{\substack{i\in\mathcal{C}\\ j\in\mathcal{C}'}} (\boldsymbol{v}_i^{(j)})^2$$

Notice that $\mathsf{A}_{\mathcal{C},t} > 0$ if and only if there exists a subset $\mathcal{C}' \subseteq [L], |\mathcal{C}'| = t$ such that $\boldsymbol{v}_i^{(j)} \neq 0$ for all $i \in \mathcal{C}, j \in \mathcal{C}'$. Hence, the maximum value of t such that $\mathsf{A}_{\mathcal{C},t} > 0$ is the number of unknown vectors in $\mathcal{V}$ having non-zero value in all the indices in $\mathcal{C}$. In other words, we have that

$$\left|\bigcap_{i\in\mathcal{C}} \mathcal{S}(i)\right| = \max_{t\in[L]} t \cdot \mathbf{1}[\mathsf{A}_{\mathcal{C},t} > 0].$$

Notice that $\mathsf{A}_{\mathcal{C},t}$ is the elementary symmetric polynomial of degree t in the variables $\{\prod_{i\in\mathcal{C}} \boldsymbol{v}_i^2\}_{\boldsymbol{v}\in\mathcal{V}}$. We can use Newton's identities to state that for all $t \in [L]$,

$$t\mathsf{A}_{\mathcal{C},t} = \sum_{p=1}^{t} (-1)^{p+1} \mathsf{A}_{\mathcal{C},t-p} \Big(\sum_{\boldsymbol{v}\in\mathcal{V}} \Big(\prod_{i\in\mathcal{C}} \boldsymbol{v}_i^2 \Big)^p \Big)$$

using which, we can recursively compute $\mathsf{A}_{\mathcal{C},t}$ for all $t \in [L]$ ($\mathsf{A}_{\mathcal{C},0} = 1$) and hence $\left|\bigcap_{i\in\mathcal{C}} \mathcal{S}(i)\right|$ if we were given $\sum_{\boldsymbol{v}\in\mathcal{V}} \left(\prod_{i\in\mathcal{C}} \boldsymbol{v}_i^2 \right)^p$ as input for all $p \in [L]$ (see Lemma 5.12). Lemma 5.6 follows from making these set of computations robust. We next show Theorem 5.1 which follows from applying Lemma 5.6 and Corollary 5.1.

Theorem 5.1. *Let $\mathcal{V}$ be a set of L unknown vectors in $\mathbb{R}^n$ satisfying Assumption 5.1. Let $\mathcal{F}_m = \mathcal{Q}_1([n]) \cup \mathcal{Q}_m(\cup_{\boldsymbol{v}\in\mathcal{V}} \mathsf{supp}(\boldsymbol{v}))$ and*

$$\Phi_m \triangleq \frac{\delta^{2Lm}}{2\Big(3L\max(R^{2Lm}, 2^L R^{L+m})\Big)^{(L-1)} L!} \left(\max_{\boldsymbol{z}\leq 2L\mathbf{1}_m} \frac{L}{\zeta_{\boldsymbol{z},\boldsymbol{z}}} + \sum_{\boldsymbol{u}<\boldsymbol{z}} \sum_{\mathsf{M}\in\mathcal{M}(\boldsymbol{z},\boldsymbol{u})} \frac{L\prod_{(\boldsymbol{r},\boldsymbol{s})\in\mathsf{M}} \zeta_{\boldsymbol{r},\boldsymbol{s}}}{\prod_{\boldsymbol{r}\in\mathcal{T}(\mathsf{M})} \zeta_{\boldsymbol{r},\boldsymbol{r}}} \right)^{-1}$$

$$f_{L,\mathcal{V}} \triangleq \max_{\substack{\boldsymbol{z} \le 2L\mathbf{1}_{\log L+1} \\ \mathcal{C} \in \mathcal{F}_{\log L+1}}} \frac{\mathbb{E}\prod_{i\in\mathcal{C}} \boldsymbol{x}_i^{2\boldsymbol{z}_{\pi(\mathcal{C},i)}}}{\Phi^2_{\log L+1}}.$$

Here $f_{L,\mathcal{V}}$ is a constant that is independent of k and n but depends on L. Then, there exists an algorithm (see Algorithm 5.4 and 5.1) that achieves Exact Support Recovery with probability at least $1-\gamma$ using $O\Big(\log(\gamma^{-1}(2L)^{\log L+1}(n+(Lk)^{\log L+1}))f_{L,\mathcal{V}}\Big)$ samples generated according to $\mathcal{P}_d$.

Now, we provide a corollary of Theorem 5.1 specifically for mean-estimation in a mixture of distributions with constant number of components i.e. $L = O(1)$. In particular, consider the setting where

$$t \sim_{\mathsf{Unif}} [L] \quad \text{and} \quad \boldsymbol{x}_i \mid t \sim \boldsymbol{P}(\boldsymbol{v}_i^{(t)}) \text{ independently for all } i \in [n]$$
$$\mathbb{E}_{\boldsymbol{x}\sim\mathcal{P}_d}[\boldsymbol{x}_i \mid t = j] = \boldsymbol{v}_i^{(j)}$$

i.e. the mean of the i^{th} co-ordinate of the random vector $\boldsymbol{x}$ distributed according to $\mathcal{P}_d$ is $\boldsymbol{v}_i^{(j)}$.

Corollary 5.3. *Consider the mean estimation problem described above. Let $\mathcal{V}$ be a set of $L = O(1)$ unknown vectors in $\mathbb{R}^n$ satisfying Assumption 5.1 and $f_{L,\mathcal{V}}$ be as defined in Theorem 5.1. Then, there exists an algorithm (see Algorithm 5.4 and 5.1) that with probability at least $1-\gamma$, achieves Exact Support Recovery using $O\Big(\log(n\gamma^{-1})\mathsf{poly}(\delta R^{-1})f_{L,\mathcal{V}}\Big)$ samples generated according to $\mathcal{P}_d$.*

We can compare the sample complexity presented in Corollary 5.3 with the alternate approach for support recovery namely the two stage process of recovering the union of support followed by parameter estimation restricted to the union of support. Most known results (other than [114]) for parameter estimation in Gaussian mixtures without separability assumptions hold for two mixtures and are therefore

not applicable for $L > 2$. For general value of L, the only known sample complexity guarantees for parameter estimation in mixture of Gaussians is provided in [114].

Note that computing the union of support is not difficult in the MD setting. In particular, in Lemma 5.6, the guarantees include the sample complexity of testing whether a particular index belongs to the union of support; this can be used to compute the union of support itself after taking a union bound over all indices leading to a multiplicative $\log n$ factor.

However, for one dimensional Gaussian mixture models (1D GMM), the parameter estimation guarantees in [114] (See Corollary 5) are polynomial in the inverse of the failure probability. Since parameter estimation in 1D GMM is used as a framework for solving the high dimensional problem, it can be extracted that the sample complexity in n dimensions must be polynomial in n with degree at least 1 to achieve a per coordinate error (error in L_∞ norm). If restricted to the union of support of the unknown vectors in $\mathcal{V}$, then using the guarantees in [114] directly will lead to a polynomial dependence on Lk. In essence, the sample complexity of the alternate approach has a logarithmic dependence on the latent space dimension and a polynomial dependence on sparsity k (for constant L). Note that our sample complexity only has a logarithmic dependence on the dimension n (and is independent of k for constant L) and is therefore essentially *dimension-free*.

For other distributions, to the best of our knowledge, the only known parameter estimation results that exist in literature are [18, 99]. In both of these works, the authors use the same assumption that $\mathbb{E}_{x\sim \boldsymbol{P}(\theta)} x^L$ can be written as a polynomial in θ of degree exactly L. While the guarantees in [18] are non-constructive, the results in [99] need the restrictive assumption that the means must be multiple of some $\epsilon > 0$ and moreover, they have an exponential dependence on the noise variance and ϵ^{-1}. Our results do not have these limitations and are therefore widely applicable.

In the next part we provide results on deduplicated support recovery in this setting. Note that from Lemma 5.5, for partial recovery, we only need to estimate correctly if $\left|\bigcap_{i\in\mathcal{C}}\mathcal{S}(i)\right|>0$ for ordered sets $\mathcal{C}\subseteq[n]$. Notice that $\left|\bigcap_{i\in\mathcal{C}}\mathcal{S}(i)\right|>0$ if and only if $\sum_{\boldsymbol{v}\in\mathcal{V}}\prod_{i\in\mathcal{C}}\boldsymbol{v}_i^2>0$. From our previous arguments, $\sum_{\boldsymbol{v}\in\mathcal{V}}\prod_{i\in\mathcal{C}}\boldsymbol{v}_i^2$ can be computed if the quantities $\mathbb{E}\prod_{i\in\mathcal{C}}\boldsymbol{x}_i^{\boldsymbol{u}_{\pi(\mathcal{C},i)}}$ for all $\boldsymbol{u}\leq 2\mathbf{1}_{|\mathcal{C}|}$ are pre-computed. The following lemma stems from making this computation robust to the randomness in the dataset:

Lemma 5.7. *Suppose Assumption 5.1 is true. Let*

$$\Phi\triangleq\max_{\boldsymbol{z}\leq 2\mathbf{1}_{|\mathcal{C}|}}\frac{\delta^{2|\mathcal{C}|}}{2}\Big(\frac{L}{\zeta_{\boldsymbol{z},\boldsymbol{z}}}+\sum_{\boldsymbol{u}<\boldsymbol{z}}\sum_{\mathsf{M}\in\mathcal{M}(\boldsymbol{z},\boldsymbol{u})}\frac{L\prod_{(\boldsymbol{r},\boldsymbol{s})\in\mathsf{M}}\zeta_{\boldsymbol{r},\boldsymbol{s}}}{\prod_{\boldsymbol{r}\in\mathcal{T}(\mathsf{M})}\zeta_{\boldsymbol{r},\boldsymbol{r}}}\Big)^{-1}$$
$$h_{L,\mathcal{V}}\triangleq\frac{\max_{\boldsymbol{z}\leq 2\mathbf{1}_{|\mathcal{C}|}}\mathbb{E}\prod_{i\in\mathcal{C}}\boldsymbol{x}_i^{2\boldsymbol{z}_{\pi(\mathcal{C},i)}}}{\Phi^2}$$

where $h_{L,\mathcal{V}}$ is a constant independent of k and n but depends on L. There exists an algorithm (see Algorithm 5.5) that can compute if $\left|\bigcap_{i\in\mathcal{C}}\mathcal{S}(i)\right|>0$ correctly for each set $\mathcal{C}\subseteq[n]$ with probability at least $1-\gamma$ using $O(h_{L,\mathcal{V}}\log\gamma^{-1})$ samples generated according to $\mathcal{P}_d$.

The subsequent theorem follows from Lemma 5.7 and Corollary 5.2. Note that, compared to exact support recovery (Theorem 5.2) the sample complexity for deduplicated support recovery has significantly improved dependency on δ and furthermore, it is also independent of R.

Theorem 5.2. *Let $\mathcal{V}$ be a set of unknown vectors in $\mathbb{R}^n$ satisfying Assumption 5.1. Let $\mathcal{F}_m=\mathcal{Q}_1([n])\cup\mathcal{Q}_m(\cup_{\boldsymbol{v}\in\mathcal{V}}\mathsf{supp}(\boldsymbol{v}))$ and*

$$\Phi_m=\max_{\boldsymbol{z}\leq 2\mathbf{1}_{|\mathcal{C}|}}\frac{\delta^{2|\mathcal{C}|}}{2}\Big(\frac{L}{\zeta_{\boldsymbol{z},\boldsymbol{z}}}+\sum_{\boldsymbol{u}<\boldsymbol{z}}\sum_{\mathsf{M}\in\mathcal{M}(\boldsymbol{z},\boldsymbol{u})}\frac{L\prod_{(\boldsymbol{r},\boldsymbol{s})\in\mathsf{M}}\zeta_{\boldsymbol{r},\boldsymbol{s}}}{\prod_{\boldsymbol{r}\in\mathcal{T}(\mathsf{M})}\zeta_{\boldsymbol{r},\boldsymbol{r}}}\Big)^{-1}$$
$$h'_{L,\mathcal{V}}\triangleq\max_{\substack{\boldsymbol{z}\leq 2\mathbf{1}_L\\ \mathcal{C}\in\mathcal{F}_L}}\frac{\mathbb{E}\prod_{i\in\mathcal{C}}\boldsymbol{x}_i^{2\boldsymbol{z}_{\pi(\mathcal{C},i)}}}{\Phi_L^2}$$

where $h'_{L,\mathcal{V}}$ is a constant independent of k and n but depends on L. Accordingly, there exists an algorithm (see Algorithm 5.5 and 5.3) that achieves Deduplicated support recovery with probability at least $1-\gamma$ using $O\Big(h'_{L,\mathcal{V}} \log(\gamma^{-1}(n+(Lk)^L))\Big)$ samples generated from $\mathcal{P}_d$.

5.3.2 Mixtures of Linear Classifiers

In this section, we will present our main results and high level techniques in the MLC setting. The detailed proofs of all results in this section can be found in Section 5.4.2. We solve the sparse recovery problem when the observed samples are generated according to $\mathcal{P}_c$ under the following assumption which states that the unknown vectors in $\mathcal{V}$ either all have non-negative entries or they all have non-positive entries.

Assumption 5.2. *The non-zero entries of unknown vectors in $\mathcal{V}$ are all either positive ($\boldsymbol{v}_i \geq 0$ for all $i \in [n], \boldsymbol{v} \in \mathcal{V}$) or they are all negative ($\boldsymbol{v}_i \leq 0$ for all $i \in [n], \boldsymbol{v} \in \mathcal{V}$).*

Next, if Assumption 5.2 is satisfied, we show the sample complexity of computing $\big|\bigcup_{i\in\mathcal{C}} \mathcal{S}(i)\big|$ for each set $\mathcal{C} \subseteq [n]$.

Lemma 5.8. *Suppose Assumptions 5.1 and 5.2 are true. Let $a = \frac{\sqrt{2(R^2+\sigma^2)}}{\delta}\mathsf{erf}^{-1}\Big(1-\frac{1}{2L}\Big)$. There exists an algorithm (see Algorithm 5.6) that can compute $\big|\bigcup_{i\in\mathcal{C}} \mathcal{S}(i)\big|$ for each set $\mathcal{C} \subseteq [n]$ with probability at least $1-\gamma$ using $O\Big((1-\phi(a))^{-|\mathcal{C}|}L^2 \log\gamma^{-1}\Big)$ i.i.d samples from $\mathcal{P}_c$.*

Let us present a high level proof of Lemma 5.8. Without loss of generality, let us assume that all unknown vectors in $\mathcal{V}$ have positive non-zero entries. For a fixed set $\mathcal{C} \subseteq [n]$, suppose we condition on the event $\mathcal{E}_{\mathcal{C}}$ which is true when for all $j \in \mathcal{C}$, $\boldsymbol{x}_j > a$ for some suitably chosen $a > 0$. Furthermore, let $\mathcal{E}_{\boldsymbol{v}}$ be the event that the particular vector $\boldsymbol{v} \in \mathcal{V}$ is used to generate the sample $(\boldsymbol{x}, y)$. Notice that if $\boldsymbol{v}_i = 0$ for all $i \in \mathcal{C}$, then conditioning on the event $\mathcal{E}_{\mathcal{C}}$ does not change the distribution of the response $y \mid \mathcal{E}_{\boldsymbol{v}}$; hence the probability of $y = 1$ is exactly $1/2$ in this case. On

the other hand, if $\boldsymbol{v}_i \neq 0$ for some $i \in \mathcal{C}$, then conditioning on the event $\mathcal{E}_{\mathcal{C}}$ does change the distribution of the response $y \mid \mathcal{E}_{\boldsymbol{v}}$. In particular, if $\boldsymbol{v}_i \neq 0$, note that $\langle \boldsymbol{v}_{|\mathcal{C}}, \boldsymbol{x}_{|\mathcal{C}} \rangle \geq a\delta$ and therefore $\Pr(y = 1 \mid \mathcal{E}_{\mathcal{C}}, \mathcal{E}_{\boldsymbol{v}})$ must be larger than $1/2$ and is an increasing function of a. Of course, if a is chosen to $+\infty$, then $\Pr(y = 1 \mid \mathcal{E}_{\mathcal{C}}, \mathcal{E}_{\boldsymbol{v}}) = 1$ and therefore $2\Pr(y = 1 \mid \mathcal{E}_{\mathcal{C}}) = 1 + L^{-1} |\cup_{i \in \mathcal{C}} \mathcal{S}(i)|$. Thus, if $a = +\infty$, we can use the fact that $|\cup_{i \in \mathcal{C}} \mathcal{S}(i)|$ is integral to compute $|\cup_{i \in \mathcal{C}} \mathcal{S}(i)|$ correctly from an estimate of $\Pr(y = 1 \mid \mathcal{E}_{\mathcal{C}})$ that is within an additive error of $1/4L$. Of course, we cannot choose $a = +\infty$ since no samples will satisfy the event $\mathcal{E}_{\mathcal{C}}$ in that case. However, we can choose a, $(a > 0)$ carefully so that it is small enough to make $\Pr(\mathcal{E}_{\mathcal{C}})$ reasonably large and at the same time, a is large enough to allow us to correctly compute $|\cup_{i \in \mathcal{C}} \mathcal{S}(i)|$ from a reasonably good estimate of $\Pr(y = 1 \mid \mathcal{E}_{\mathcal{C}})$. Next, we can again use Lemma 5.8 and Corollary 5.1 to arrive at the main theorem for Mixtures of Linear Classifiers:

Theorem 5.3. *Let $\mathcal{V}$ be a set of L unknown vectors in $\mathbb{R}^n$ satisfying Assumptions 5.1 and 5.2. Let $a = \frac{\sqrt{2(R^2+\sigma^2)}}{\delta} \mathsf{erf}^{-1}\Big(1 - \frac{1}{2L}\Big)$. Then, there exists an algorithm (see Algorithm 5.6 and 5.1) that achieves Exact Support Recovery with probability at least $1 - \gamma$ using $O\Big((1 - \phi(a))^{-(\log L + 1)} L^2 \log(\gamma^{-1}(n + (Lk)^{\log L + 1}))\Big)$ samples generated according to $\mathcal{P}_c$.*

The only comparable result is provided in [125] who provide parameter estimation guarantees in the MLC setting. However, since it is not evident how to recover the union of support in the sparse MLC setting; directly applying the result in [125] will lead to polynomial dependence on n which is undesirable. Moreover, the guarantees in [125] also require the latent parameter vectors to be linearly independent. In contrast, our sample complexity guarantees for support recovery scale logarithmically with n and also does not need the latent parameter vectors to be linearly independent (in fact they are not even required to be distinct).

5.3.3 Mixtures of Linear Regression

Finally, we move on to the mixtures of linear regression or MLR setting. Note that the sample complexity guarantees for MLC (Theorem 5.3) is also valid in the MLR setting as we can simulate MLR responses by simply taking the sign of the response in the MLR dataset. However, note that the sample complexity presented in Theorem 5.3 has a poor dependence on R, δ and L. Here we solve the support recovery problem provided the unknown vectors in $\mathcal{V}$ are all binary and demonstrate significantly better sample complexity guarantees under this assumption. The detailed proofs of all results in this section can be found in Section 5.4.3. As usual, we start with a lemma where we characterize the sample complexity of estimating $\left|\bigcap_{i\in\mathcal{C}} \mathcal{S}(i)\right|$ correctly:

Lemma 5.9. *If the unknown vectors in the set $\mathcal{V}$ are all binary i.e. $\boldsymbol{v}^{(1)}, \boldsymbol{v}^{(2)}, \ldots, \boldsymbol{v}^{(L)} \in \{0,1\}^n$, then, with probability at least $1-\gamma$, for each set $\mathcal{C} \subseteq [n]$, there exists an algorithm (see Algorithm 5.7) that can compute $\left|\bigcap_{i\in\mathcal{C}} \mathcal{S}(i)\right|$ using*

$$O(L^2(k+\sigma^2)^{|C|/2}(\log n)^{2|C|}\log\gamma^{-1})$$

i.i.d samples from $\mathcal{P}_r$.

We provide a high level proof of Lemma 5.9 here. We consider the random variable $y^{|\mathcal{C}|}\cdot\left(\prod_{i\in\mathcal{C}}\boldsymbol{x}_i\right)$ where $(\boldsymbol{x},y)\sim\mathcal{P}_r$. Clearly, we can write $y=\langle\boldsymbol{v},\boldsymbol{x}\rangle+\zeta$ where $\zeta\sim\mathcal{N}(0,\sigma^2)$ and $\boldsymbol{v}$ is uniformly sampled from the set of unknown vectors $\mathcal{V}$. We can show that

$$\mathbb{E}_{(\boldsymbol{x},y)\sim\mathcal{P}_r} y^{|\mathcal{C}|}\cdot\Big(\prod_{i\in\mathcal{C}}\boldsymbol{x}_i\Big) = \mathbb{E}_{\boldsymbol{x},\zeta}\ell^{-1}\sum_{\boldsymbol{v}\in\mathcal{V}}\Big(\prod_{i\in\mathcal{C}}\boldsymbol{x}_i\Big)\cdot\Big(\langle\boldsymbol{v},\boldsymbol{x}\rangle+\zeta\Big)^{|\mathcal{C}|}$$

$$\mathbb{E}y^{|\mathcal{C}|}\cdot\Big(\prod_{i\in\mathcal{C}}\boldsymbol{x}_i\Big) = \frac{1}{\ell}\sum_{\boldsymbol{v}\in\mathcal{V}}\Big(\prod_{i\in\mathcal{C}}\mathbb{E}\boldsymbol{x}_i^2\cdot\boldsymbol{v}_i\Big) = \frac{\left|\bigcap_{i\in\mathcal{C}}\mathcal{S}(i)\right|}{\ell}.$$

$$\mathbb{E}_{(\boldsymbol{x},y)\sim\mathcal{P}_r} y^{|\mathcal{C}|} \cdot \Big(\prod_{i\in\mathcal{C}} \boldsymbol{x}_i\Big) = \mathbb{E}_{\boldsymbol{x},\zeta} \ell^{-1} \sum_{\boldsymbol{v}\in\mathcal{V}} \Big(\prod_{i\in\mathcal{C}} \boldsymbol{x}_i\Big) \cdot \Big(\langle \boldsymbol{v}, \boldsymbol{x}\rangle + \zeta\Big)^{|\mathcal{C}|}$$
$$\mathbb{E} y^{|\mathcal{C}|} \cdot \Big(\prod_{i\in\mathcal{C}} \boldsymbol{x}_i\Big) = \frac{1}{L} \sum_{\boldsymbol{v}\in\mathcal{V}} \Big(\prod_{i\in\mathcal{C}} \mathbb{E}\boldsymbol{x}_i^2 \cdot \boldsymbol{v}_i\Big) = \frac{\big|\bigcap_{i\in\mathcal{C}} \mathcal{S}(i)\big|}{L}.$$

Hence, by using the fact that $\big|\bigcap_{i\in\mathcal{C}} \mathcal{S}(i)\big|$ is integral, we can estimate the quantity correctly from a reasonably good estimate of $\mathbb{E} y^{|\mathcal{C}|} \cdot \Big(\prod_{i\in\mathcal{C}} \boldsymbol{x}_i\Big)$. Again, by an application of Corollary 5.1, we arrive at the following theorem:

Theorem 5.4. *Let $\mathcal{V}$ be a set of L unknown binary vectors in $\{0,1\}^n$. Then, with probability at least $1-\gamma$, there exists an algorithm (see Algorithm 5.7 and 5.1) that achieves Exact Support Recovery with*

$$O\left(L^2(k+\sigma^2)^{(\log L+1)/2}(\log n)^{2(\log L+1)} \log((n+(Lk)^{\log L+1})\gamma^{-1})\right)$$

samples generated according to $\mathcal{P}_r$.

As in mixtures of distributions, it is possible to recover the union of support of the unknown vectors in $\mathcal{V}$ in the MLR setting with a small number of samples (see Lemma C.5 in Appendix C.2). Therefore an alternate approach that can be used for support recovery is to recover the union of support followed by parameter estimation with the features being restricted to the union of the support. Note that if the set of unknown vectors satisfy Assumption 5.1, then estimating each vector up to an L_2 norm of δ will suffice for support recovery. Hence, by using Lemma C.5 followed by Theorem 1 in [103], we arrive at the following result for support recovery:

Theorem 5.5. *Let $\mathcal{V}$ be a set of L unknown vectors satisfying Assumption 5.1. Further, assume that any two distinct vectors $\boldsymbol{v}, \boldsymbol{v}' \in \mathcal{V}$ satisfies $\|\boldsymbol{v}-\boldsymbol{v}'\|_2 \geq \Delta$. Then,*

with high probability, there exists an algorithm that achieves Exact Support Recovery with

$$O\left(Lk\log\left(\frac{Lk}{\delta}\right)\mathsf{poly}\left(\frac{L\sigma}{\Delta}\right)+\left(\frac{\sigma L}{\Delta}\right)^{O(L^2)}+L^2(R^2+\sigma^2)(\log n)^3/\delta^2\right)$$

samples generated according to $\mathcal{P}_r$.

If the unknown vectors in $\mathcal{V}$ are restricted to being binary, then the sample complexity in Theorem 5.5 has a linear dependence on the sparsity but on the other hand, its dependence on σ, L is very poor; note that Theorem 5.5 uses parameter estimation framework in mixtures of Gaussians ([114]) as a black-box leading to the polynomial in L, σ with a possibly high degree. Moreover, the sample complexity in Theorem 5.5 has an $\exp(L^2)$ dependence on the number of unknown vectors which is undesirable when the number of unknown vectors L is large. In contrast, the sample complexity of Theorem 5.4 has a polynomial dependence on L, k, σ whose degree can be precisely extracted from the expression. In particular, in the regime where σ or L is large, Theorem 5.4 provides significant improvements over the guarantees in Theorem 5.5. Finally, although not mentioned explicitly in Theorem 1 in [103], it can be extracted that the sample complexity is polynomial in γ^{-1} where γ is the failure probability; this leads to a similar dependence on the failure probability in Theorem 5.5. On the other hand, the sample complexity in Theorem 5.4 depends logarithmically on γ^{-1}.

Our final results are for deduplicated support recovery in the MLR setting under different assumptions. Below, we state Assumption 5.3 which is a generic condition and if satisfied by the set of unknown vectors $\mathcal{V}$ allows for deduplicated support recovery of $\mathcal{V}$.

Assumption 5.3. *We assume that there exists positive numbers $\alpha_1, \alpha_2, \ldots, \alpha_L > 0$ such that for all sets $\mathcal{C} \subseteq [n], |C| \leq L$ the following condition is satisfied by the set of L unknown vectors $\boldsymbol{v}^{(1)}, \boldsymbol{v}^{(2)}, \ldots, \boldsymbol{v}^{(L)} \in \mathcal{V}$:*

$$\text{If there exists } \boldsymbol{v} \in \mathcal{V} \text{ such that } \prod_{j \in \mathcal{C}} \boldsymbol{v}_j \neq 0 \text{ then } \left| \sum_{\boldsymbol{v} \in \mathcal{V}} \Big(\prod_{j \in \mathcal{C}} \boldsymbol{v}_j \Big) \right| \geq \alpha_{|\mathcal{C}|}.$$

Theorem 5.6. *Suppose the following conditions are satisfied:*

1. *All unknown vectors in $\mathcal{V}$ are bounded within a ball of radius R i.e. $\left|\left|\boldsymbol{v}^{(i)}\right|\right|_2 \leq R$ for all $i \in [L]$.*
2. *Assumption 5.3 is satisfied by the set of unknown vectors $\mathcal{V}$.*

Accordingly, there exists an algorithm (see Algorithms 5.8 and 5.3) that achieves Deduplicated support recovery with probability at least $1 - \gamma$ using

$$O(L^2(R^2 + \sigma^2)^{L/2}(\log n)^{2L} \log((n + (Lk)^L)\gamma^{-1})/\alpha_L^2)$$

samples from $\mathcal{P}_r$.

Next, using Theorem 5.6, we provide deduplicated support recovery guarantees in two cases: 1) The set of unknown vectors in $\mathcal{V}$ satisfies Assumptions 5.1 and all unknown parameters are non-negative 2) The non-zero entries in the unknown vectors in $\mathcal{V}$ are distributed according to a zero mean Gaussian $\mathcal{N}(0, \nu^2)$.

Corollary 5.4. *Consider a set of L unknown vectors $\mathcal{V}$ that satisfies Assumptions 5.1 and furthermore, every non-zero entry in all the unknown vectors is positive ($\boldsymbol{v}_i \geq 0$ for all $i \in [n], \boldsymbol{v} \in \mathcal{V}$). In that case, Assumption 5.3 is satisfied with $\alpha_{|\mathcal{C}|} \geq \delta^{|\mathcal{C}|}$. Accordingly, there exists an algorithm that achieves Deduplicated support recovery with probability at least $1 - \gamma$ using*

$$O(L^2(R^2 + \sigma^2)^{L/2}(\log n/\delta)^{2L} \log((n + (Lk)^L)\gamma^{-1}))$$

samples from $\mathcal{P}_r$.

Corollary 5.5. *If all non-zero entries in the set of unknown vectors $\mathcal{V}$ are sampled i.i.d according to $\mathcal{N}(0, \nu^2)$, then with probability $1 - \eta$, Assumption 5.3 is satisfied with $\alpha_{|\mathcal{C}|} \geq \delta_{|\mathcal{C}|}^{|\mathcal{C}|}$ where*

$$\delta_{|\mathcal{C}|} = \Big(\sqrt{\frac{\pi}{8}} \frac{\nu\eta}{L|\mathcal{C}|(Lk)^{|\mathcal{C}|}}\Big).$$

Conditioned on this event, there exists an Algorithm that achieves Deduplicated support recovery with probability at least $1 - \gamma$ using

$$O(L^2(R^2 + \sigma^2)^{L/2}(\log n)^{2L} \log((n + (Lk)^L)\gamma^{-1})/\delta_L^2)$$

samples from $\mathcal{P}_r$.

5.4 Detailed Algorithms and Proofs

5.4.1 Mixtures of Distributions

Lemma 5.10. *For each fixed set $\mathcal{C} \subseteq [n]$ and each vector $\boldsymbol{t} \in (\mathbb{Z}^+)^{|\mathcal{C}|}$, we must have*

$$\mathbb{E} \prod_{i \in \mathcal{C}} \boldsymbol{x}_i^{\boldsymbol{t}_{\pi(\mathcal{C},i)}} = \frac{1}{L} \sum_{\boldsymbol{u} \leq \boldsymbol{t}} \zeta_{\boldsymbol{t},\boldsymbol{u}} \cdot \Big(\sum_{j \in [L]} \prod_{i \in \mathcal{C}} (\boldsymbol{v}_i^{(j)})^{\boldsymbol{u}_{\pi(\mathcal{C},i)}} \Big).$$

Proof. We will have

$$\mathbb{E} \prod_{i \in \mathcal{C}} \boldsymbol{x}_i^{\boldsymbol{t}_{\pi(\mathcal{C},i)}} = \frac{1}{L} \sum_{j \in [L]} \Big(\prod_{i \in \mathcal{C}} q_{\boldsymbol{t}_{\pi(\mathcal{C},i)}}(\boldsymbol{v}_i^{(j)}) \Big).$$

From the above equations, note that each summand is a product of polynomials in $\boldsymbol{v}_i^{(j)}$ for a fixed j. Expanding the polynomial and using the fact that $\zeta_{\boldsymbol{t},\boldsymbol{u}} =$

Algorithm 5.4 RECOVER $|\bigcap_{i\in\mathcal{C}}\mathcal{S}(i)|$ IN MD SETTING

Require: Samples $\boldsymbol{x}^{(1)}, \boldsymbol{x}^{(2)}, \ldots, \boldsymbol{x}^{(m)} \sim \mathcal{P}_d$. Set $\mathcal{C} \subseteq [n]$.

1: For every $\boldsymbol{z} \leq 2L\mathbf{1}_{|\mathcal{C}|}$, compute estimate $\widehat{U}^{\boldsymbol{z}}$ of $\mathbb{E}\prod_{i\in\mathcal{C}} \boldsymbol{x}_i^{\boldsymbol{z}_{\pi(\mathcal{C},i)}}$ using Algorithm C.1 on the set of samples $\{(\boldsymbol{x}_i^j)^{\boldsymbol{z}_{\pi(\mathcal{C},i)}}\}_{j=1}^m$.

2: For every $\boldsymbol{z} \leq 2L\mathbf{1}_{|\mathcal{C}|}$, compute an estimate $\widehat{V}^{\boldsymbol{z}}$ of $\sum_{j\in[L]}\prod_{i\in\mathcal{C}}(\boldsymbol{v}_i^{(j)})^{\boldsymbol{z}_{\pi(\mathcal{C},i)}}$ recursively using the following equation:

$$L\widehat{U}^{\boldsymbol{z}} - \sum_{\boldsymbol{u}<\boldsymbol{z}} \zeta_{\boldsymbol{z},\boldsymbol{u}} \cdot \widehat{V}^{\boldsymbol{u}} = \zeta_{\boldsymbol{z},\boldsymbol{z}} \cdot \widehat{V}^{\boldsymbol{z}}.$$

3: For every $t \in [L]$, compute an estimate $\widehat{\mathsf{A}}_{\mathcal{C},t}$ of $\sum_{\substack{\mathcal{C}'\subseteq[L]\\|\mathcal{C}'|=t}}\prod_{\substack{i\in\mathcal{C}\\ j\in\mathcal{C}'}}(\boldsymbol{v}_i^{(j)})^2$ recursively using Newton's identities

$$t\widehat{\mathsf{A}}_{\mathcal{C},t} = \sum_{p=1}^{t}(-1)^{p+1}\widehat{\mathsf{A}}_{\mathcal{C},t-p}\widehat{V}^{2p\mathbf{1}_{|\mathcal{C}|}}.$$

4: Return $\max_{t\in[L]} t.\mathbf{1}[\widehat{\mathsf{A}}_{\mathcal{C},t} > 0]$.

$\prod_{i\in\mathcal{C}}\beta_{\boldsymbol{t}_{\pi(\mathcal{C},i)},\boldsymbol{u}_{\pi(\mathcal{C},i)}+1}$ is the coefficient of the monomial $\prod_{i\in\mathcal{C}}(\boldsymbol{v}_i^{(j)})^{\boldsymbol{u}_{\pi(\mathcal{C},i)}}$ for all $j \in [L]$, we obtain the proof of the lemma.

□

Lemma 5.11. *For each fixed set $\mathcal{C} \subseteq [n]$ and each vector $\boldsymbol{t} \in (\mathbb{Z}^+)^{|\mathcal{C}|}$, we can compute $\sum_{j\in[L]}\prod_{i\in\mathcal{C}}(\boldsymbol{v}_i^{(j)})^{\boldsymbol{t}_{\pi(\mathcal{C},i)}}$ provided for all $\boldsymbol{u} \in (\mathbb{Z}^+)^{|\mathcal{C}|}$ satisfying $\boldsymbol{u} \leq \boldsymbol{t}$, the quantities $\mathbb{E}\prod_{i\in\mathcal{C}}\boldsymbol{x}_i^{\boldsymbol{u}_{\pi(\mathcal{C},i)}}$ are pre-computed.*

Proof. We will prove this lemma by induction. For the base case, we have from Lemma 5.10 that $L\mathbb{E}\boldsymbol{x}_i = \beta_{1,2}\sum_{j\in[L]}\boldsymbol{v}_i^{(j)} + \beta_{1,1}$. Hence $\sum_{j\in[L]}\boldsymbol{v}_i^{(j)}$ can be computed from $\mathbb{E}\boldsymbol{x}_i$ by using the following equation:

$$\sum_{j\in[L]}\boldsymbol{v}_i^{(j)} = \frac{1}{\beta_{1,2}}\Big(L\mathbb{E}\boldsymbol{x}_i - \beta_{1,1}\Big).$$

Now suppose for all vectors $\boldsymbol{u} \in (\mathbb{Z}^+)^{|\mathcal{C}|}$ satisfying $\boldsymbol{u} \leq \boldsymbol{t}$, the lemma statement is true. Consider another vector $\boldsymbol{z} \in (\mathbb{Z}^+)^{|\mathcal{C}|}$ such that there exists an index $j \in |\mathcal{C}|$ for

which $\boldsymbol{z}_j = \boldsymbol{t}_j + 1$ and $\boldsymbol{z}_i = \boldsymbol{t}_i$ for all $i \neq j$. From the statement of Lemma 5.10, we know that

$$\mathbb{E}\prod_{i\in\mathcal{C}} \boldsymbol{x}_i^{\boldsymbol{z}_{\pi(\mathcal{C},i)}} = \frac{1}{L}\sum_{\boldsymbol{u}\leq\boldsymbol{z}} \zeta_{\boldsymbol{z},\boldsymbol{u}} \cdot \Big(\sum_{j\in[L]}\prod_{i\in\mathcal{C}} (\boldsymbol{v}_i^{(j)})^{\boldsymbol{u}_{\pi(\mathcal{C},i)}}\Big)$$

where $\zeta_{\boldsymbol{z},\boldsymbol{u}} = \prod_{i\in\mathcal{C}} \beta_{\boldsymbol{z}_{\pi(\mathcal{C},i)},\boldsymbol{u}_{\pi(\mathcal{C},i)}+1}$. From our induction hypothesis, we have already computed $\sum_{j\in[L]}\prod_{i\in\mathcal{C}} (\boldsymbol{v}_i^{(j)})^{\boldsymbol{u}_{\pi(\mathcal{C},i)}}$ for all $\boldsymbol{u} < \boldsymbol{z}$ (the set $\{\boldsymbol{u} \in (\mathbb{Z}^+)^{|\mathcal{C}|} \mid \boldsymbol{u} < \boldsymbol{z}\}$ is equivalent to the set $\{\boldsymbol{u} \in (\mathbb{Z}^+)^{|\mathcal{C}|} \mid \boldsymbol{u} \leq \boldsymbol{t}\}$). Since $\mathbb{E}\prod_{i\in\mathcal{C}} \boldsymbol{x}_i^{\boldsymbol{z}_{\pi(\mathcal{C},i)}}$ is already pre-computed, we can compute $\sum_{j\in[L]}\prod_{i\in\mathcal{C}} (\boldsymbol{v}_i^{(j)})^{\boldsymbol{z}_{\pi(\mathcal{C},i)}}$ as follows:

$$L\mathbb{E}\prod_{i\in\mathcal{C}} \boldsymbol{x}_i^{\boldsymbol{z}_{\pi(\mathcal{C},i)}} - \sum_{\boldsymbol{u}<\boldsymbol{z}} \zeta_{\boldsymbol{z},\boldsymbol{u}} \cdot \Big(\sum_{j\in[L]}\prod_{i\in\mathcal{C}} (\boldsymbol{v}_i^{(j)})^{\boldsymbol{u}_{\pi(\mathcal{C},i)}}\Big) = \zeta_{\boldsymbol{z},\boldsymbol{z}} \cdot \Big(\sum_{j\in[L]}\prod_{i\in\mathcal{C}} (\boldsymbol{v}_i^{(j)})^{\boldsymbol{z}_{\pi(\mathcal{C},i)}}\Big).$$

This completes the proof of the lemma. □

Lemma 5.12. *For each fixed set $\mathcal{C} \subseteq [n]$, we can compute $\left|\bigcap_{i\in\mathcal{C}} \mathcal{S}(i)\right|$ provided for all $p \in [L]$, the quantity $\sum_{\boldsymbol{v}\in\mathcal{V}} \left(\prod_{i\in\mathcal{C}} \boldsymbol{v}_i^2\right)^p$ is pre-computed.*

Proof. Let us fix a particular subset $\mathcal{C} \subseteq [n]$. Now, let us define the quantity

$$\mathsf{A}_{\mathcal{C},t} = \sum_{\substack{\mathcal{C}'\subseteq[L]\\|\mathcal{C}'|=t}} \prod_{\substack{i\in\mathcal{C}\\ j\in\mathcal{C}'}} (\boldsymbol{v}_i^{(j)})^2$$

Notice that $\mathsf{A}_{\mathcal{C},t} > 0$ if and only if there exists a subset $\mathcal{C}' \subseteq [L], |\mathcal{C}'| = t$ such that $\boldsymbol{v}_i^{(j)} \neq 0$ for all $i \in \mathcal{C}, j \in \mathcal{C}'$. Hence, the maximum value of t such that $\mathsf{A}_{\mathcal{C},t} > 0$ is the number of unknown vectors in $\mathcal{V}$ having non-zero value in all the indices in $\mathcal{C}$. In other words, we have that

$$\left|\bigcap_{i\in\mathcal{C}} \mathcal{S}(i)\right| = \max_{t\in[L]} t \cdot \mathbf{1}[\mathsf{A}_{\mathcal{C},t} > 0].$$

Let $t^\star$ be the maximum value of t for which $\mathsf{A}_{\mathcal{C},t} > 0$. We will have $\mathsf{A}_{\mathcal{C},t^\star} \geq \delta^{2L|\mathcal{C}|}$. It is easy to recognize $\sum_{\boldsymbol{v}\in\mathcal{V}} \Big(\prod_{i\in\mathcal{C}} \boldsymbol{v}_i^2 \Big)^p$ as the power sum polynomial of degree p in the variables $\{\prod_{i\in\mathcal{C}} \boldsymbol{v}_i^2\}_{\boldsymbol{v}\in\mathcal{V}}$. On the other hand, $\mathsf{A}_{\mathcal{C},t}$ is the elementary symmetric polynomial of degree t in the variables $\{\prod_{i\in\mathcal{C}} \boldsymbol{v}_i^2\}_{\boldsymbol{v}\in\mathcal{V}}$. We can use Newton's identities to state that for all $t \in [L]$,

$$t\mathsf{A}_{\mathcal{C},t} = \sum_{p=1}^{t} (-1)^{p+1} \mathsf{A}_{\mathcal{C},t-p} \Big(\sum_{\boldsymbol{v}\in\mathcal{V}} \Big(\prod_{i\in\mathcal{C}} \boldsymbol{v}_i^2 \Big)^p \Big)$$

using which, we can recursively compute $\mathsf{A}_{\mathcal{C},t}$ for all $t \in [L]$ if we were given $\sum_{\boldsymbol{v}\in\mathcal{V}} \Big(\prod_{i\in\mathcal{C}} \boldsymbol{v}_i^2 \Big)^p$ as input for all $p \in [L]$.

□

We are now ready to prove Lemma 5.6.

Lemma (Restatement of Lemma 5.6)**.** *Suppose Assumption 5.1 is true. Let*

$$\begin{aligned}
\Phi \triangleq & \frac{\delta^{2L|\mathcal{C}|}}{2\Big(3\max(LR^{2L|\mathcal{C}|}, 2^L R^{L+|\mathcal{C}|})\Big)^{(L-1)} L!} \\
& \times \left(\max_{\boldsymbol{z}\leq 2L\mathbf{1}_{|\mathcal{C}|}} \frac{L}{\zeta_{\boldsymbol{z},\boldsymbol{z}}} + \sum_{\boldsymbol{u}<\boldsymbol{z}} \sum_{\mathsf{M}\in\mathcal{M}(\boldsymbol{z},\boldsymbol{u})} \frac{L \prod_{(\boldsymbol{r},\boldsymbol{s})\in\mathsf{M}} \zeta_{\boldsymbol{r},\boldsymbol{s}}}{\prod_{\boldsymbol{r}\in\mathcal{T}(\mathsf{M})} \zeta_{\boldsymbol{r},\boldsymbol{r}}} \right)^{-1} \\
g_{L,\mathcal{V}} \triangleq & \frac{\max_{\boldsymbol{z}\leq 2L\mathbf{1}_{|\mathcal{C}|}} \mathbb{E} \prod_{i\in\mathcal{C}} \boldsymbol{x}_i^{2\boldsymbol{z}_{\pi(\mathcal{C},i)}}}{\Phi^2}
\end{aligned}$$

where $g_{L,\mathcal{V}}$ is a constant that is independent of k and n but depends on L. There exists an algorithm (see Algorithm 5.4) that can compute $\Big|\bigcap_{i\in\mathcal{C}} \mathcal{S}(i)\Big|$ exactly for each set $\mathcal{C} \subseteq [n]$ with probability at least $1-\gamma$ using $O\Big(\log(\gamma^{-1}(2L)^{|\mathcal{C}|}) f_{L,\mathcal{V}} \Big)$ samples generated according to $\mathcal{P}_d$.

Proof. Suppose, for every vector $\boldsymbol{z} \in (\mathbb{Z}^+)^{|\mathcal{C}|}$ satisfying $\boldsymbol{z} \leq 2L\mathbf{1}_{|\mathcal{C}|}$, we compute an estimate $\widehat{U}^{\boldsymbol{z}}$ of $\mathbb{E}\prod_{i\in\mathcal{C}} \boldsymbol{x}_i^{\boldsymbol{z}_{\pi(\mathcal{C},i)}}$ such that $\left|\widehat{U}^{\boldsymbol{z}} - \mathbb{E}\prod_{i\in\mathcal{C}} \boldsymbol{x}_i^{\boldsymbol{z}_{\pi(\mathcal{C},i)}}\right| \leq \Phi_{\boldsymbol{z}}$ where $\Phi_{\boldsymbol{z}}$ is going to be determined later. Recall that in Lemma 5.12, we showed

$$L\mathbb{E}\prod_{i\in\mathcal{C}} \boldsymbol{x}_i^{\boldsymbol{z}_{\pi(\mathcal{C},i)}} - \sum_{\boldsymbol{u}<\boldsymbol{z}} \zeta_{\boldsymbol{z},\boldsymbol{u}} \cdot \Big(\sum_{j\in[L]}\prod_{i\in\mathcal{C}} (\boldsymbol{v}_i^{(j)})^{\boldsymbol{u}_{\pi(\mathcal{C},i)}}\Big) = \zeta_{\boldsymbol{z},\boldsymbol{z}} \cdot \Big(\sum_{j\in[L]}\prod_{i\in\mathcal{C}} (\boldsymbol{v}_i^{(j)})^{\boldsymbol{z}_{\pi(\mathcal{C},i)}}\Big). \quad (5.3)$$

Using the computed $\widehat{U}^{\boldsymbol{z}}$'s , we can compute an estimate $\widehat{V}^{\boldsymbol{z}}$ of $\sum_{j\in[L]}\prod_{i\in\mathcal{C}}(\boldsymbol{v}_i^{(j)})^{\boldsymbol{z}_{\pi(\mathcal{C},i)}}$ for all $\boldsymbol{z} \in (\mathbb{Z}^+)^{|\mathcal{C}|}$ satisfying $\boldsymbol{z} \leq 2L\mathbf{1}_{|\mathcal{C}|}$. Let us denote the error in estimation by $\epsilon_{\boldsymbol{z}}$ i.e. we have $\left|\widehat{V}^{\boldsymbol{z}} - \sum_{j\in[L]}\prod_{i\in\mathcal{C}}(\boldsymbol{v}_i^{(j)})^{\boldsymbol{z}_{\pi(\mathcal{C},i)}}\right| \leq \epsilon_{\boldsymbol{z}}$. Now, we prove the following claim.

Claim 5.1. *We must have*

$$\epsilon_{\boldsymbol{z}} \leq \frac{L\Phi_{\boldsymbol{z}}}{\zeta_{\boldsymbol{z},\boldsymbol{z}}} + \sum_{\boldsymbol{u}<\boldsymbol{z}}\sum_{\mathsf{M}\in\mathcal{M}(\boldsymbol{z},\boldsymbol{u})} \frac{L\Phi_{\boldsymbol{u}}\prod_{(\boldsymbol{r},\boldsymbol{s})\in\mathsf{M}}\zeta_{\boldsymbol{r},\boldsymbol{s}}}{\prod_{\boldsymbol{r}\in\mathcal{T}(\mathsf{M})}\zeta_{\boldsymbol{r},\boldsymbol{r}}}$$

Proof. We will prove this lemma by induction. Let $\boldsymbol{e}_i$ be the standard basis vector having a non-zero entry at the i^{th} index and is zero everywhere else. For the base case, we have from Lemma 5.10 that $L\mathbb{E}\boldsymbol{x}_i = \beta_{1,2}\sum_{j\in[L]}\boldsymbol{v}_i^j + \beta_{1,1}$. Therefore, we must have

$$L\mathbb{E}\boldsymbol{x}_i - L\widehat{U}^{\boldsymbol{e}_i} = \beta_{1,2}(\sum_{j\in[L]}\boldsymbol{v}_i^j - \widehat{U}^{\boldsymbol{e}_i})$$

$$\implies L\Phi_{\boldsymbol{e}_i} = \beta_{1,2}\epsilon_{\boldsymbol{e}_i}.$$

From definition, (recall that $\zeta_{\boldsymbol{z},\boldsymbol{u}} = \prod_{i\in\mathcal{C}}\beta_{\boldsymbol{z}_{\pi(i)},\boldsymbol{u}_{\pi(i)}+1}$), we have $\zeta_{\boldsymbol{e}_i,\boldsymbol{e}_i} = \beta_{1,2}$ which completes the proof of the base case. Now suppose for all vectors $\boldsymbol{u} \in (\mathbb{Z}^+)^{|\mathcal{C}|}$ satisfying $\boldsymbol{u} \leq \boldsymbol{t}$, the lemma statement is true. Consider another vector $\boldsymbol{z} \in (\mathbb{Z}^+)^{|\mathcal{C}|}$ such that there exists an index $j \in |\mathcal{C}|$ for which $\boldsymbol{z}_j = \boldsymbol{t}_j + 1$ and $\boldsymbol{z}_i = \boldsymbol{t}_i$ for all $i \neq j$. From the statement of Lemma 5.10, we know that

$$L\mathbb{E}\prod_{i\in\mathcal{C}}\boldsymbol{x}_i^{\boldsymbol{z}_{\pi(i)}}-\sum_{\boldsymbol{u}<\boldsymbol{z}}\zeta_{\boldsymbol{z},\boldsymbol{u}}\cdot\Big(\sum_{j\in[L]}\prod_{i\in\mathcal{C}}(\boldsymbol{v}_i^j)^{\boldsymbol{u}_{\pi(i)}}\Big)=\zeta_{\boldsymbol{z},\boldsymbol{z}}\cdot\Big(\sum_{j\in[L]}\prod_{i\in\mathcal{C}}(\boldsymbol{v}_i^j)^{\boldsymbol{z}_{\pi(i)}}\Big).$$

Hence, we must have

$$\begin{aligned}&\Big(L\mathbb{E}\prod_{i\in\mathcal{C}}\boldsymbol{x}_i^{\boldsymbol{z}_{\pi(i)}}-L\widehat{U}^{\boldsymbol{z}}\Big)-\Big(\sum_{\boldsymbol{u}<\boldsymbol{z}}\zeta_{\boldsymbol{z},\boldsymbol{u}}\cdot\Big(\sum_{j\in[L]}\prod_{i\in\mathcal{C}}(\boldsymbol{v}_i^j)^{\boldsymbol{u}_{\pi(i)}}-\widehat{V}^{\boldsymbol{u}}\Big)\Big)\\&=\zeta_{\boldsymbol{z},\boldsymbol{z}}\cdot\Big(\sum_{j\in[L]}\prod_{i\in\mathcal{C}}(\boldsymbol{v}_i^j)^{\boldsymbol{z}_{\pi(i)}}-\widehat{V}^{\boldsymbol{z}}\Big)\\&\implies\zeta_{\boldsymbol{z},\boldsymbol{z}}\epsilon_{\boldsymbol{z}}\le L\Phi_{\boldsymbol{z}}+\sum_{\boldsymbol{u}<\boldsymbol{z}}\zeta_{\boldsymbol{z},\boldsymbol{u}}\epsilon_{\boldsymbol{u}}.\end{aligned}$$

Now, by using our induction hypothesis, we must have

$$\begin{aligned}\zeta_{\boldsymbol{z},\boldsymbol{z}}\epsilon_{\boldsymbol{z}}&\le L\Phi_{\boldsymbol{z}}+\sum_{\boldsymbol{u}<\boldsymbol{z}}\zeta_{\boldsymbol{z},\boldsymbol{u}}\left(\frac{L\Phi_{\boldsymbol{u}}}{\zeta_{\boldsymbol{u},\boldsymbol{u}}}+\sum_{\boldsymbol{v}<\boldsymbol{u}}\sum_{\mathsf{M}\in\mathcal{M}(\boldsymbol{u},\boldsymbol{v})}\frac{L\Phi_{\boldsymbol{v}}\prod_{(\boldsymbol{r},\boldsymbol{s})\in\mathsf{M}}\zeta_{\boldsymbol{r},\boldsymbol{s}}}{\prod_{\boldsymbol{r}\in\mathcal{T}(\mathsf{M})}\zeta_{\boldsymbol{r},\boldsymbol{r}}}\right)\\&\implies\epsilon_{\boldsymbol{z}}\le\frac{L\Phi_{\boldsymbol{z}}}{\zeta_{\boldsymbol{z},\boldsymbol{z}}}+\sum_{\boldsymbol{u}<\boldsymbol{z}}\zeta_{\boldsymbol{z},\boldsymbol{u}}\left(\frac{L\Phi_{\boldsymbol{u}}}{\zeta_{\boldsymbol{z},\boldsymbol{z}}\zeta_{\boldsymbol{u},\boldsymbol{u}}}+\sum_{\boldsymbol{v}<\boldsymbol{u}}\sum_{\mathsf{M}\in\mathcal{M}(\boldsymbol{u},\boldsymbol{v})}\frac{L\Phi_{\boldsymbol{v}}\prod_{(\boldsymbol{r},\boldsymbol{s})\in\mathsf{M}}\zeta_{\boldsymbol{r},\boldsymbol{s}}}{\zeta_{\boldsymbol{z},\boldsymbol{z}}\prod_{\boldsymbol{r}\in\mathcal{T}(\mathsf{M})}\zeta_{\boldsymbol{r},\boldsymbol{r}}}\right)\\&\implies\epsilon_{\boldsymbol{z}}\le\frac{L\Phi_{\boldsymbol{z}}}{\zeta_{\boldsymbol{z},\boldsymbol{z}}}+\sum_{\boldsymbol{u}<\boldsymbol{z}}\sum_{\mathsf{M}\in\mathcal{M}(\boldsymbol{z},\boldsymbol{u})}\frac{L\Phi_{\boldsymbol{u}}\prod_{(\boldsymbol{r},\boldsymbol{s})\in\mathsf{M}}\zeta_{\boldsymbol{r},\boldsymbol{s}}}{\prod_{\boldsymbol{r}\in\mathcal{T}(\mathsf{M})}\zeta_{\boldsymbol{r},\boldsymbol{r}}}.\end{aligned}$$

This completes the proof of the claim. □

Hence, for fixed $\Phi_{\boldsymbol{z}}=\Phi$ for all $\boldsymbol{z}\le 2L\mathbf{1}_{|\mathcal{C}|}$, we get

$$\epsilon_{\boldsymbol{z}}\le\Phi\Big(\frac{L}{\zeta_{\boldsymbol{z},\boldsymbol{z}}}+\sum_{\boldsymbol{u}<\boldsymbol{z}}\sum_{\mathsf{M}\in\mathcal{M}(\boldsymbol{z},\boldsymbol{u})}\frac{L\prod_{(\boldsymbol{r},\boldsymbol{s})\in\mathsf{M}}\zeta_{\boldsymbol{r},\boldsymbol{s}}}{\prod_{\boldsymbol{r}\in\mathcal{T}(\mathsf{M})}\zeta_{\boldsymbol{r},\boldsymbol{r}}}\Big).$$

For a fixed Φ, let us write ϵ to denote the following quantity:

$$\epsilon\triangleq\max_{\boldsymbol{z}\le 2L\mathbf{1}_{|\mathcal{C}|}}\Phi\Big(\frac{L}{\zeta_{\boldsymbol{z},\boldsymbol{z}}}+\sum_{\boldsymbol{u}<\boldsymbol{z}}\sum_{Q\in\mathcal{Q}(\boldsymbol{z},\boldsymbol{u})}\frac{L\prod_{(\boldsymbol{r},\boldsymbol{s})\in Q}\zeta_{\boldsymbol{r},\boldsymbol{s}}}{\prod_{\boldsymbol{r}\in\mathcal{T}(Q)}\zeta_{\boldsymbol{r},\boldsymbol{r}}}\Big)$$

Consider a fixed subset of indices $\mathcal{C} \subseteq [n]$ and a fixed vector $\boldsymbol{t} \in (\mathbb{Z}^+)^{|\mathcal{C}|}$. Using the fact $\max_{\boldsymbol{v}\in\mathcal{V}, i\in[n]} \boldsymbol{v}_i^2 \leq R^2$, we have that

$$\frac{1}{L}\sum_{\boldsymbol{v}\in\mathcal{V}}\Big(\prod_{i\in\mathcal{C}}\boldsymbol{v}_i^2\Big)^p \leq R^{2p|\mathcal{C}|} \quad \text{and} \quad \mathsf{A}_{\mathcal{C},t} = \sum_{\substack{\mathcal{C}'\subseteq[L]\\|\mathcal{C}'|=t}}\prod_{\substack{i\in\mathcal{C}\\ j\in\mathcal{C}'}}(\boldsymbol{v}_i^{(j)})^2 \leq \binom{L}{t}R^{2(t+|\mathcal{C}|)} \leq 2^L R^{2(t+|\mathcal{C}|)}.$$

We can compute an estimate $\widehat{\mathsf{A}}_{\mathcal{C},t}$ of $\mathsf{A}_{\mathcal{C},t}$ by using $\widehat{V}^{2p\mathbf{1}_{|\mathcal{C}|}}$ in the following set of recursive equations

$$t\widehat{\mathsf{A}}_{\mathcal{C},t} = \sum_{p=1}^{t}(-1)^{p+1}\widehat{\mathsf{A}}_{\mathcal{C},t-p}\widehat{V}^{2p\mathbf{1}_{|\mathcal{C}|}}.$$

Claim 5.2.

$$\left|\widehat{\mathsf{A}}_{\mathcal{C},t} - \mathsf{A}_{\mathcal{C},t}\right| \leq \epsilon\Big(3\max(LR^{2L|\mathcal{C}|}, 2^L R^{L+|\mathcal{C}|})\Big)^{(t-1)} t! \text{ for all } t \in [L].$$

Proof. We will prove this claim by induction. For the base case i.e. $t = 1$, notice that

$$\left|\widehat{\mathsf{A}}_{\mathcal{C},1} - \mathsf{A}_{\mathcal{C},1}\right| \leq \left|\widehat{V}^{2\mathbf{1}_{|\mathcal{C}|}} - \sum_{\boldsymbol{v}\in\mathcal{V}}\prod_{i\in\mathcal{C}}\boldsymbol{v}_i^2\right| \leq \epsilon.$$

Now, suppose for all $t \leq k$, the following holds true:

$$\left|\widehat{\mathsf{A}}_{\mathcal{C},t} - \mathsf{A}_{\mathcal{C},t}\right| \leq \epsilon\Big(3\max(LR^{2L|\mathcal{C}|}, 2^L R^{L+|\mathcal{C}|})\Big)^{t-1} t!.$$

For ease of notation, let us denote $a = 3\max(LR^{2L|\mathcal{C}|}, 2^L R^{L+|\mathcal{C}|})$. In that case, for $t = k+1$, we must have

$$t\left|\widehat{\mathsf{A}}_{\mathcal{C},t} - \mathsf{A}_{\mathcal{C},t}\right| \leq \sum_{p\leq t}\left|\widehat{\mathsf{A}}_{\mathcal{C},t-p}\widehat{V}^{2p\mathbf{1}_{|\mathcal{C}|}} - \mathsf{A}_{\mathcal{C},t-p}\cdot\sum_{\boldsymbol{v}\in\mathcal{V}}\Big(\prod_{i\in\mathcal{C}}\boldsymbol{v}_i^2\Big)^p\right|$$

$$
\begin{aligned}
&\leq \left| \widehat{V}^{2\mathbf{1}_{|\mathcal{C}|}} - \sum_{\boldsymbol{v}\in\mathcal{V}} \Big(\prod_{i\in\mathcal{C}} \boldsymbol{v}_i^2 \Big)^{(k+1)} \right| \\
&+ \sum_{p\leq t-1} \left| \epsilon a^{t-2}(t-1)! \cdot \sum_{\boldsymbol{v}\in\mathcal{V}} \Big(\prod_{i\in\mathcal{C}} \boldsymbol{v}_i^2 \Big)^{p} + \epsilon \cdot \mathsf{A}_{\mathcal{C},t-p} + \epsilon^2 a^{t-2}(t-1)! \right| \\
&\leq \epsilon + \sum_{p\leq t-1} \left| \epsilon a^{t-2}(t-1)! L R^{2L|\mathcal{C}|} + \epsilon \cdot 2^L R^{2(L+|\mathcal{C}|)} + \epsilon^2 a^{t-2}(t-1)! \right| \\
&\leq \epsilon + \sum_{p\leq t-1} \epsilon a^{t-1}(t-1)! \leq \epsilon a^{(t-1)} t!.
\end{aligned}
$$

Hence, $\left| \widehat{\mathsf{A}}_{\mathcal{C},t} - \mathsf{A}_{\mathcal{C},t} \right| \leq \epsilon a^{t-1} t!$ thus proving our claim. □

Hence, to identify $t^\star$ correctly, we must have

$$
\begin{aligned}
&\epsilon \Big(3 \max(LR^{2L|\mathcal{C}|}, 2^L R^{L+|\mathcal{C}|}) \Big)^{(L-1)} L! \leq \frac{\delta^{2L|\mathcal{C}|}}{2} \\
&\implies \Phi \leq \frac{\delta^{2L|\mathcal{C}|}}{2 \Big(3 \max(LR^{2L|\mathcal{C}|}, 2^L R^{L+|\mathcal{C}|}) \Big)^{(L-1)} L!} \\
&\times \left(\max_{\boldsymbol{z}\leq 2p\mathbf{1}_{|\mathcal{C}|}} \frac{1}{\zeta_{\boldsymbol{z},\boldsymbol{z}}} + \sum_{\boldsymbol{u}<\boldsymbol{z}} \sum_{\mathsf{M}\in\mathcal{M}(\boldsymbol{z},\boldsymbol{u})} \frac{\prod_{(\boldsymbol{r},\boldsymbol{s})\in\mathsf{M}} \zeta_{\boldsymbol{r},\boldsymbol{s}}}{\prod_{\boldsymbol{r}\in\mathcal{T}(\mathsf{M})} \zeta_{\boldsymbol{r},\boldsymbol{r}}} \right)^{-1}
\end{aligned}
$$

where we inserted the definition of Φ. Therefore, for every vector $\boldsymbol{z} \in (\mathbb{Z}^+)^{|\mathcal{C}|}$ satisfying $\boldsymbol{z} \leq 2L\mathbf{1}_{|\mathcal{C}|}$, in order to compute $\widehat{U}^{\boldsymbol{z}}$ of $\mathbb{E}\prod_{i\in\mathcal{C}} \boldsymbol{x}_i^{\boldsymbol{z}_{\pi(\mathcal{C},i)}}$ such that $\left| \widehat{U}^{\boldsymbol{z}} - \mathbb{E}\prod_{i\in\mathcal{C}} \boldsymbol{x}_i^{\boldsymbol{z}_{\pi(\mathcal{C},i)}} \right| \leq \Phi$, the number of samples that is sufficient with probability $1-\gamma$ is going to be

$$
O\Big(\log(\gamma^{-1}(2L)^{|\mathcal{C}|}) \frac{\max_{\boldsymbol{z}\leq 2L\mathbf{1}_{|\mathcal{C}|}} \mathbb{E}\prod_{i\in\mathcal{C}} \boldsymbol{x}_i^{2\boldsymbol{z}_{\pi(\mathcal{C},i)}}}{\Phi^2} \Big).
$$

□

Theorem (Restatement of Theorem 5.1)**.** *Let $\mathcal{V}$ be a set of L unknown vectors in $\mathbb{R}^n$ satisfying Assumption 5.1. Let $\mathcal{F}_m = \mathcal{Q}_1([n]) \cup \mathcal{Q}_m(\cup_{\boldsymbol{v}\in\mathcal{V}} \mathsf{supp}(\boldsymbol{v}))$ and*

$$
\Phi_m = \frac{\delta^{2Lm}}{2 \Big(3L \max(R^{2Lm}, 2^L R^{L+m}) \Big)^{(L-1)} L!}
$$

$$\times \left(\max_{\boldsymbol{z} \le 2L\mathbf{1}_m} \frac{L}{\zeta_{\boldsymbol{z},\boldsymbol{z}}} + \sum_{\boldsymbol{u}<\boldsymbol{z}} \sum_{\mathsf{M} \in \mathcal{M}(\boldsymbol{z},\boldsymbol{u})} \frac{L \prod_{(\boldsymbol{r},\boldsymbol{s}) \in \mathsf{M}} \zeta_{\boldsymbol{r},\boldsymbol{s}}}{\prod_{\boldsymbol{r} \in \mathcal{T}(\mathsf{M})} \zeta_{\boldsymbol{r},\boldsymbol{r}}} \right)^{-1}$$

$$f_{L,\mathcal{V}} = \max_{\substack{\boldsymbol{z} \le 2L\mathbf{1}_{\log L+1} \\ \mathcal{C} \in \mathcal{F}_{\log L+1}}} \frac{\mathbb{E} \prod_{i \in \mathcal{C}} \boldsymbol{x}_i^{2\boldsymbol{z}_{\pi(\mathcal{C},i)}}}{\Phi^2_{\log L+1}}$$

where $f_{L,\mathcal{V}}$ is a constant that is independent of k and n but depends on L. Then, there exists an algorithm (see Algorithm 5.4 and 5.1) that achieves Exact Support Recovery with probability at least $1-\gamma$ using $O\Big(\log(\gamma^{-1}(2L)^{\log L+1}(n+(Lk)^{\log L+1}))f_{L,\mathcal{V}} \Big)$ samples generated according to $\mathcal{P}_d$.

Proof. The proof follows directly from Corollary 5.1 and Lemma 5.6. □

Corollary (Restatement of Corollary 5.3)**.** *Consider the mean estimation problem where $\mathbb{E}_{\boldsymbol{x} \sim \mathcal{P}_d}[\boldsymbol{x}_i \mid t = j] = \boldsymbol{v}_i^{(j)}$. Let $\mathcal{V}$ be a set of $L = O(1)$ unknown vectors in $\mathbb{R}^n$ satisfying Assumption 5.1 and $f_{L,\mathcal{V}}$ be as defined in Theorem 5.2. Then, there exists an algorithm (see Algorithm 5.4 and 5.1) that with probability at least $1-\gamma$, achieves Exact Support Recovery using $O\Big(\log(n\gamma^{-1})\mathsf{poly}(\delta R^{-1})f_{L,\mathcal{V}} \Big)$ samples generated according to $\mathcal{P}_d$.*

Proof. We can re-scale the samples (dividing them by R) so that Assumption 5.1 will be satisfied with $\delta' = \delta/R$ and $R' \le 1$. Since L is a constant, $\Phi_{\log L} = O(\mathsf{poly}(\delta R^{-1}))$. Therefore, the corollary follows from Theorem 5.1. □

Lemma (Restatement of Lemma 5.7)**.** *Suppose Assumption 5.1 is true. Let*

$$\Phi \triangleq \max_{\boldsymbol{z} \le 2\mathbf{1}_{|\mathcal{C}|}} \frac{\delta^{2|\mathcal{C}|}}{2} \Big(\frac{L}{\zeta_{\boldsymbol{z},\boldsymbol{z}}} + \sum_{\boldsymbol{u}<\boldsymbol{z}} \sum_{\mathsf{M} \in \mathcal{M}(\boldsymbol{z},\boldsymbol{u})} \frac{L \prod_{(\boldsymbol{r},\boldsymbol{s}) \in \mathsf{M}} \zeta_{\boldsymbol{r},\boldsymbol{s}}}{\prod_{\boldsymbol{r} \in \mathcal{T}(\mathsf{M})} \zeta_{\boldsymbol{r},\boldsymbol{r}}} \Big)^{-1}$$

$$h_{L,\mathcal{V}} \triangleq \frac{\max_{\boldsymbol{z} \le 2\mathbf{1}_{|\mathcal{C}|}} \mathbb{E} \prod_{i \in \mathcal{C}} \boldsymbol{x}_i^{2\boldsymbol{z}_{\pi(\mathcal{C},i)}}}{\Phi^2}$$

where $h_{L,\mathcal{V}}$ is a constant independent of k and n but depends on L. There exists an algorithm (see Algorithm 5.5) that can compute if $\left| \bigcap_{i \in \mathcal{C}} \mathcal{S}(i) \right| > 0$ correctly for each

Algorithm 5.5 ESTIMATE IF $|\bigcap_{i\in\mathcal{C}}\mathcal{S}(i)|>0$ IN MD SETTING

Require: Samples $\boldsymbol{x}^{(1)},\boldsymbol{x}^{(2)},\ldots,\boldsymbol{x}^{(m)}\sim\mathcal{P}_d$. Set $\mathcal{C}\subseteq[n]$.

1: For every $\boldsymbol{z}\leq 2\mathbf{1}_{|\mathcal{C}|}$, compute estimate $\widehat{U}^{\boldsymbol{z}}$ of $\mathbb{E}\prod_{i\in\mathcal{C}}\boldsymbol{x}_i^{\boldsymbol{z}_{\pi(\mathcal{C},i)}}$ using Algorithm C.1 on the set of samples $\{(\boldsymbol{x}_i^j)^{\boldsymbol{z}_{\pi(\mathcal{C},i)}}\}_{j=1}^m$.

2: For every $\boldsymbol{z}\leq 2\mathbf{1}_{|\mathcal{C}|}$, compute an estimate $\widehat{V}^{\boldsymbol{z}}$ of $\sum_{j\in[L]}\prod_{i\in\mathcal{C}}(\boldsymbol{v}_i^{(j)})^{\boldsymbol{z}_{\pi\mathcal{C},i}}$ recursively using the following equation:

$$L\widehat{U}^{\boldsymbol{z}}-\sum_{\boldsymbol{u}<\boldsymbol{z}}\zeta_{\boldsymbol{z},\boldsymbol{u}}\cdot\widehat{V}^{\boldsymbol{u}}=\zeta_{\boldsymbol{z},\boldsymbol{z}}\cdot\widehat{V}^{\boldsymbol{z}}.$$

3: If $\widehat{V}^{2\mathbf{1}_{|\mathcal{C}|}}\geq\delta^{2|\mathcal{C}|}/2$, return True and otherwise return False.

set $\mathcal{C}\subseteq[n]$ with probability at least $1-\gamma$ using $O(h_{L,\mathcal{V}}\log\gamma^{-1})$ samples generated according to $\mathcal{P}_d$.

Proof. For a fixed ordered set $\mathcal{C}\subseteq[n]$, consider the statistic $\sum_{\boldsymbol{v}\in\mathcal{V}}\prod_{i\in\mathcal{C}}\boldsymbol{v}_i^2$. If $\sum_{\boldsymbol{v}\in\mathcal{V}}\prod_{i\in\mathcal{C}}\boldsymbol{v}_i^2>0$, then $|\cap_{i\in\mathcal{C}}\mathcal{S}(i)|>0$ and otherwise, if $\sum_{\boldsymbol{v}\in\mathcal{V}}\prod_{i\in\mathcal{C}}\boldsymbol{v}_i^2=0$, then $|\cap_{i\in\mathcal{C}}\mathcal{S}(i)|=0$. Hence it suffices to estimate correctly if $\sum_{\boldsymbol{v}\in\mathcal{V}}\prod_{i\in\mathcal{C}}\boldsymbol{v}_i^2>0$ or not. From Lemma 5.11, we know that for each set $\mathcal{C}\subseteq[n]$, we can compute $\sum_{j\in[L]}\prod_{i\in\mathcal{C}}(\boldsymbol{v}_i^{(j)})^2$ provided for all $\boldsymbol{u}\in(\mathbb{Z}^+)^{|\mathcal{C}|}$ satisfying $\boldsymbol{u}\leq 2\mathbf{1}_{|\mathcal{C}|}$, the quantity $\mathbb{E}\prod_{i\in\mathcal{C}}\boldsymbol{x}_i^{\boldsymbol{u}_{\pi(\mathcal{C},i)}}$ is pre-computed.

Suppose, for every vector $\boldsymbol{z}\in(\mathbb{Z}^+)^{|\mathcal{C}|}$ satisfying $\boldsymbol{z}\leq 2\mathbf{1}_{|\mathcal{C}|}$, we compute an estimate $\widehat{U}^{\boldsymbol{z}}$ of $\mathbb{E}\prod_{i\in\mathcal{C}}\boldsymbol{x}_i^{\boldsymbol{z}_{\pi(\mathcal{C},i)}}$ such that $\left|\widehat{U}^{\boldsymbol{z}}-\mathbb{E}\prod_{i\in\mathcal{C}}\boldsymbol{x}_i^{\boldsymbol{z}_{\pi(\mathcal{C},i)}}\right|\leq\Phi$ where Φ is going to be determined later. Using the computed $\widehat{U}^{\boldsymbol{z}}$'s , we can compute an estimate $\widehat{V}^{\boldsymbol{z}}$ of $\sum_{j\in[L]}\prod_{i\in\mathcal{C}}(\boldsymbol{v}_i^{(j)})^{\boldsymbol{z}_{\pi(\mathcal{C},i)}}$ for all $\boldsymbol{z}\in(\mathbb{Z}^+)^{|\mathcal{C}|}$ satisfying $\boldsymbol{z}\leq 2\mathbf{1}_{|\mathcal{C}|}$. As before, let us denote the error in estimation by $\epsilon_{\boldsymbol{z}}$ i.e. we have $\left|\widehat{V}^{\boldsymbol{z}}-\sum_{j\in[L]}\prod_{i\in\mathcal{C}}(\boldsymbol{v}_i^{(j)})^{\boldsymbol{z}_{\pi(\mathcal{C},i)}}\right|\leq\epsilon_{\boldsymbol{z}}$. Note that we showed in Lemma 5.12 that for fixed Φ, we get for all $\boldsymbol{z}\leq 2\mathbf{1}_{|\mathcal{C}|}$,

$$\epsilon_{\boldsymbol{z}}\leq\Phi\Big(\frac{L}{\zeta_{\boldsymbol{z},\boldsymbol{z}}}+\sum_{\boldsymbol{u}<\boldsymbol{z}}\sum_{\mathsf{M}\in\mathcal{M}(\boldsymbol{z},\boldsymbol{u})}\frac{L\prod_{(\boldsymbol{r},\boldsymbol{s})\in\mathsf{M}}\zeta_{\boldsymbol{r},\boldsymbol{s}}}{\prod_{\boldsymbol{r}\in\mathcal{T}(\mathsf{M})}\zeta_{\boldsymbol{r},\boldsymbol{r}}}\Big).$$

Note that the minimum value of $\sum_{\boldsymbol{v}\in\mathcal{V}}\prod_{i\in\mathcal{C}}\boldsymbol{v}_i^2$ is at least $\delta^{2|\mathcal{C}|}$ and therefore, it suffices $\epsilon_{\boldsymbol{z}}$ to be less than $\delta^{2|\mathcal{C}|}/2$. Hence, it is sufficient if

$$\Phi \leq \max_{\boldsymbol{z} \leq 2\mathbf{1}_{|\mathcal{C}|}} \frac{\delta^{2|\mathcal{C}|}}{2}\Big(\frac{L}{\zeta_{\boldsymbol{z},\boldsymbol{z}}} + \sum_{\boldsymbol{u}<\boldsymbol{z}} \sum_{\mathsf{M}\in\mathcal{M}(\boldsymbol{z},\boldsymbol{u})} \frac{L\prod_{(\boldsymbol{r},\boldsymbol{s})\in\mathsf{M}} \zeta_{\boldsymbol{r},\boldsymbol{s}}}{\prod_{\boldsymbol{r}\in\mathcal{T}(\mathsf{M})} \zeta_{\boldsymbol{r},\boldsymbol{r}}}\Big)^{-1}.$$

Now, we use Lemma C.6 to complete the proof of the lemma (similar to Lemma 5.12)

□

Theorem (Restatement of Theorem 5.2)**.** *Let $\mathcal{V}$ be a set of unknown vectors in $\mathbb{R}^n$ satisfying Assumption 5.1. Let $\mathcal{F}_m = \mathcal{Q}_1([n]) \cup \mathcal{Q}_m(\cup_{\boldsymbol{v}\in\mathcal{V}}\mathsf{supp}(\boldsymbol{v}))$ and*

$$\Phi_m = \max_{\boldsymbol{z} \leq 2\mathbf{1}_{|\mathcal{C}|}} \frac{\delta^{2|\mathcal{C}|}}{2}\Big(\frac{L}{\zeta_{\boldsymbol{z},\boldsymbol{z}}} + \sum_{\boldsymbol{u}<\boldsymbol{z}} \sum_{\mathsf{M}\in\mathcal{M}(\boldsymbol{z},\boldsymbol{u})} \frac{L\prod_{(\boldsymbol{r},\boldsymbol{s})\in\mathsf{M}} \zeta_{\boldsymbol{r},\boldsymbol{s}}}{\prod_{\boldsymbol{r}\in\mathcal{T}(\mathsf{M})} \zeta_{\boldsymbol{r},\boldsymbol{r}}}\Big)^{-1}$$

$$h'_{L,\mathcal{V}} \triangleq \max_{\substack{\boldsymbol{z}\leq 2\mathbf{1}_L \\ \mathcal{C}\in\mathcal{F}_L}} \frac{\mathbb{E}\prod_{i\in\mathcal{C}} \boldsymbol{x}_i^{2\boldsymbol{z}_{\pi(\mathcal{C},i)}}}{\Phi_L^2}$$

where $h'_{L,\mathcal{V}}$ is a constant independent of k and n but depends on L. Accordingly, there exists an algorithm (see Algorithm 5.5 and 5.3) that achieves Deduplicated support recovery with probability at least $1-\gamma$ using $O\Big(h'_{L,\mathcal{V}} \log(\gamma^{-1}(n+(Lk)^L))\Big)$ samples generated from $\mathcal{P}_d$.

Proof. The proof follows from Lemma 5.7 and Corollary 5.2. □

5.4.2 Mixtures of Linear Classifiers

Recall that in this section, we solve the sparse recovery problem when the observed samples are generated according to $\mathcal{P}_c$ under Assumption 5.2.

Lemma (Restatement of Lemma 5.8)**.** *Suppose Assumptions 5.1 and 5.2 are true. Let $a = \frac{\sqrt{2(R^2+\sigma^2)}}{\delta}\mathsf{erf}^{-1}\Big(1-\frac{1}{2L}\Big)$. There exists an algorithm (see Algorithm 5.6) that can compute $\Big|\bigcup_{i\in\mathcal{C}} \mathcal{S}(i)\Big|$ for each set $\mathcal{C}\subseteq[n]$ with probability at least $1-\gamma$ using $O\Big((1-\phi(a))^{-|\mathcal{C}|}L^2\log\gamma^{-1}\Big)$ i.i.d samples from $\mathcal{P}_c$.*

Proof. Without loss of generality, let us assume that all unknown vectors in $\mathcal{V}$ have positive non-zero entries. for each fixed set $\mathcal{C}\subseteq[n]$, we will condition on event $\mathcal{E}_{\mathcal{C}}$

Algorithm 5.6 RECOVER $|\bigcup_{i\in\mathcal{C}} \mathcal{S}(i)|$ IN MLC SETTING

Require: Samples $(\boldsymbol{x}^{(1)}, y^{(1)}), (\boldsymbol{x}^{(2)}, y^{(2)}), \ldots, (\boldsymbol{x}^{(m)}, y^{(m)}) \sim \mathcal{P}_c$. Set $\mathcal{C} \subseteq [n]$. Parameter $a > 0$.

1: Find the subset of samples $\mathcal{T} = \{(\boldsymbol{x}^{(i)}, y^{(i)}) \mid \boldsymbol{x}_j^{(i)} > a \text{ for all } i \in [m]\}$.

2: Compute an estimate $\widehat{P}$ of $\Pr(y = 1 \mid \mathcal{E}_{\mathcal{C}})$ as

$$\widehat{P} = \frac{1}{|\mathcal{T}|} \sum_{(\boldsymbol{x},y)\in\mathcal{T}} \mathbf{1}\left[y = 1\right]$$

3: Find $t \in [L]$ such that

$$\frac{1}{2}\Big(1 + \frac{t}{L}\Big) - \frac{t}{4L^2} \leq \widehat{P} \leq \frac{1}{2}\Big(1 + \frac{t}{L}\Big)$$

4: Return t

defined as follows: for all $j \in \mathcal{C}$, the data-point $\boldsymbol{x}$ satisfies $\boldsymbol{x}_j > a$ for some suitably chosen $a > 0$. Recall that the minimum magnitude of any non-zero entry in an unknown vector in $\mathcal{V}$ is at least δ. Further condition on the event $\mathcal{E}_{\boldsymbol{v}}$ which is true when a particular unknown vector $\boldsymbol{v}$ is being sampled from $\mathcal{V}$. In that case, we show the following claim:

Claim 5.3.

$$\Pr(y = 1 \mid \mathcal{E}_{\boldsymbol{v}}, \mathcal{E}_{\mathcal{C}}) = \frac{1}{2} \text{ if } \boldsymbol{v}_{|\mathcal{C}} = \mathbf{0}$$
$$1 \geq \Pr(y = 1 \mid \mathcal{E}_{\boldsymbol{v}}, \mathcal{E}_{\mathcal{C}}) \geq \frac{1}{2} + \frac{1}{2} \cdot \mathsf{erf}\Big(\frac{a\delta}{\sqrt{2(R^2 + \sigma^2)}}\Big) \text{ if } \boldsymbol{v}_{|\mathcal{C}} \neq \mathbf{0}.$$

Proof. In order to see the above equation, note that if $\boldsymbol{v}_{|\mathcal{C}} = \mathbf{0}$, then $\langle \boldsymbol{v}, \boldsymbol{x}\rangle + z \sim \mathcal{N}(0, ||\boldsymbol{v}||_2^2 + \sigma^2)$ or in other words, conditioning on the event $\mathcal{E}_{\mathcal{C}}$ has no effect on the distribution of y. On the other hand, if $\boldsymbol{v}_{|\mathcal{C}} \neq 0$, conditioning on the event $\mathcal{E}_{\mathcal{C}}$ modifies the distribution of y. Consider an index $j \in \mathsf{supp}(\boldsymbol{v}) \cap \mathcal{C}$. Since $\boldsymbol{v}_j \boldsymbol{x}_j \geq a\delta$, we must have $\langle \boldsymbol{v}_{|\mathcal{C}}, \boldsymbol{x}_{|\mathcal{C}}\rangle \geq a\delta$ using Assumption 5.2. Therefore, the probability that $y = 1$ must be at least $\Pr(\langle \boldsymbol{v}_{|[n]\setminus\mathcal{C}}, \boldsymbol{x}_{|[n]\setminus\mathcal{C}}\rangle + z \geq -a\delta)$. Using the fact that

$\langle \boldsymbol{v}_{|[n]\setminus\mathcal{C}}, \boldsymbol{x}_{|[n]\setminus\mathcal{C}}\rangle + z \sim \mathcal{N}(0, \nu^2+\sigma^2)$ (where $\nu \le R$) and the property of error function ($\Pr_{u\sim\mathcal{N}(0,\sigma^2)}(|u| \le a) = \mathsf{erf}(a/\sqrt{2}\sigma)$), we prove the claim. □

Hence we must have

$$\frac{1}{2} + \frac{|\cup_{i\in\mathcal{C}}\mathcal{S}(i)|}{2L} \ge \Pr(y=1 \mid \mathcal{E}_{\mathcal{C}}) \ge \frac{1}{2} + \frac{|\cup_{i\in\mathcal{C}}\mathcal{S}(i)|}{2L}\mathsf{erf}\Big(\frac{a\delta}{\sqrt{2(R^2+\sigma^2)}}\Big)$$

We choose a such that $\mathsf{erf}\Big(\frac{a\delta}{\sqrt{2(R^2+\sigma^2)}}\Big) \ge 1 - \frac{1}{2L}$ in which case, we must have

$$\frac{1}{2}\Big(1+\frac{1}{L}\Big|\bigcup_{i\in\mathcal{C}}\mathcal{S}(i)\Big|\Big) - \frac{1}{4L^2}\cdot\Big|\bigcup_{i\in\mathcal{C}}\mathcal{S}(i)\Big| \le \Pr(y=1 \mid \mathcal{E}_{\mathcal{C}}) \le \frac{1}{2}\Big(1+\frac{1}{L}\Big|\bigcup_{i\in\mathcal{C}}\mathcal{S}(i)\Big|\Big)$$

Clearly, for each value of $\big|\bigcup_{i\in\mathcal{C}}\mathcal{S}(i)\big| \in \{0,1,\ldots,L\}$, the interval in which $\Pr(y = 1 \mid \mathcal{E}_{\mathcal{C}})$ lies are disjoint and each interval is separated by at least $1/4L$. Hence, if we are able to estimate $\Pr(y=1 \mid \mathcal{E}_{\mathcal{C}})$ up to an additive factor of $1/8L$, then we can uniquely (and correctly) decode the value of $\big|\bigcup_{i\in\mathcal{C}}\mathcal{S}(i)\big|$. By using Chernoff bound, with $O(L^2\log\gamma^{-1})$ samples satisfying the event $\mathcal{E}_{\mathcal{C}}$, we can estimate $\Pr(y=1 \mid \mathcal{E}_{\mathcal{C}})$ (See Step 2 in Algorithm 5.6 for the estimator) with probability at least $1-\gamma/2$. From our previous analysis, we chose $a = \frac{\sqrt{2(R^2+\sigma^2)}}{\delta}\mathsf{erf}^{-1}\Big(1-\frac{1}{2L}\Big)$. The probability that for a sample $(\boldsymbol{x}, y) \sim \mathcal{P}_c$, the event $\mathcal{E}_{\mathcal{C}}$ is true is exactly $O\Big((1-\phi(a))^{|\mathcal{C}|}\Big)$. Therefore, with $(1-\phi(a))^{-|\mathcal{C}|}L^2\log\gamma^{-1}\Big)$ samples, we will have $O(L^2\log\gamma^{-1})$ samples satisfying the event $\mathcal{E}_{\mathcal{C}}$ with probability at least $1-\gamma/2$. Hence, this allows us to recover $\big|\bigcup_{i\in\mathcal{C}}\mathcal{S}(i)\big|$ with probability at least $1-\gamma$. □

Theorem (Restatement of Theorem 5.3)**.** *Let $\mathcal{V}$ be a set of L unknown vectors in $\mathbb{R}^n$ satisfying Assumptions 5.1 and 5.2. Let $a = \frac{\sqrt{2(R^2+\sigma^2)}}{\delta}\mathsf{erf}^{-1}\Big(1-\frac{1}{2L}\Big)$. Then, there exists an algorithm (see Algorithm 5.6 and 5.1) that achieves Exact Support Recovery*

with probability at least $1-\gamma$ *using* $O\Big((1-\phi(a))^{-(\log L+1)}L^2\log(\gamma^{-1}(n+(Lk)^{\log L+1}))\Big)$ *samples generated according to* $\mathcal{P}_c$.

Proof. The proof follows directly from Lemma 5.8 and Corollary 5.1. □

5.4.3 Mixtures of Linear Regression

Algorithm 5.7 RECOVER $|\bigcap_{i\in\mathcal{C}}\mathcal{S}(i)|$ IN MLR SETTING

Require: Samples $(\boldsymbol{x}^{(1)},y^{(1)}),(\boldsymbol{x}^{(2)},y^{(2)}),\ldots,(\boldsymbol{x}^{(m)},y^{(m)})\sim\mathcal{P}_r$. Set $\mathcal{C}\subseteq[n]$.

1: Return $\mathsf{round}\Big(\frac{L}{m}\cdot\sum_{j=1}^{m}\Big(y^{(j)}\Big)^{|\mathcal{C}|}\Big(\prod_{i\in\mathcal{C}}\boldsymbol{x}_i^{(j)}\Big)\Big)$

Unknown binary Vectors

Lemma (Restatement of Lemma 5.9)**.** *If the unknown vectors in the set* $\mathcal{V}$ *are all binary i.e.* $\boldsymbol{v}^{(1)},\boldsymbol{v}^{(2)},\ldots,\boldsymbol{v}^{(L)}\in\{0,1\}^n$*, then, with probability at least* $1-\gamma$*, for each set* $\mathcal{C}\subseteq[n]$*, there exists an algorithm (see Algorithm 5.7) that can compute* $\big|\bigcap_{i\in\mathcal{C}}\mathcal{S}(i)\big|$ *using* $O(L^2(k+\sigma^2)^{|C|/2}(\log n)^{2|C|}\log\gamma^{-1})$ *i.i.d samples from* $\mathcal{P}_r$.

Proof. Consider the random variable $y^{|\mathcal{C}|}\cdot\Big(\prod_{i\in\mathcal{C}}\boldsymbol{x}_i\Big)$ where $(\boldsymbol{x},y)\sim\mathcal{P}_r$. Clearly, we can write $y=\langle\boldsymbol{v},\boldsymbol{x}\rangle+\zeta$ where $\zeta\sim\mathcal{N}(0,\sigma^2)$ and $\boldsymbol{v}$ is uniformly sampled from the set of unknown vectors $\mathcal{V}$. Therefore, we must have

$$\mathbb{E}_{(\boldsymbol{x},y)\sim\mathcal{P}_r}y^{|\mathcal{C}|}\cdot\Big(\prod_{i\in\mathcal{C}}\boldsymbol{x}_i\Big)=\mathbb{E}_{\boldsymbol{x},\zeta}\ell^{-1}\sum_{\boldsymbol{v}\in\mathcal{V}}\Big(\prod_{i\in\mathcal{C}}\boldsymbol{x}_i\Big)\cdot\Big(\langle\boldsymbol{v},\boldsymbol{x}\rangle+\zeta\Big)^{|\mathcal{C}|}$$

$$\mathbb{E}y^{|\mathcal{C}|}\cdot\Big(\prod_{i\in\mathcal{C}}\boldsymbol{x}_i\Big)=\frac{1}{L}\sum_{\boldsymbol{v}\in\mathcal{V}}\Big(\prod_{i\in\mathcal{C}}\mathbb{E}\boldsymbol{x}_i^2\cdot\boldsymbol{v}_i\Big)=\frac{\big|\bigcap_{i\in\mathcal{C}}\mathcal{S}(i)\big|}{L}.$$

This is because in the expansion of $(\langle\boldsymbol{v},\boldsymbol{x}\rangle+\zeta)^{|\mathcal{C}|}$, the only monomial containing $\boldsymbol{x}_i$ for all $i\in\mathcal{C}$ is $\prod_{i\in\mathcal{C}}\boldsymbol{v}_i\boldsymbol{x}_i$. For any other monomial, the product with $\prod_{i\in\mathcal{C}}\boldsymbol{x}_i$ will contain some $\boldsymbol{x}_j, j\in\mathcal{C}$ such that the degree of $\boldsymbol{x}_j$ in the monomial is 1; the expectation of this monomial goes to zero as all the $\boldsymbol{x}_i$'s are independent. Since $\mathbb{E}\boldsymbol{x}_i^2=1$ for all

$i \in [n]$ and $\prod_{i\in\mathcal{C}} \boldsymbol{v}_i$ is $\mathtt{1}$ iff $\boldsymbol{v}_i = 1$ for all $i \in \mathcal{C}$ (and $\mathtt{0}$ otherwise), we obtain the desired equations. We estimate $\left|\bigcap_{i\in\mathcal{C}} \mathcal{S}(i)\right|$ by computing the following sample average

$$\frac{L}{m} \cdot \sum_{j=1}^{m} \left(y^{(j)}\right)^{|\mathcal{C}|} \Big(\prod_{i\in\mathcal{C}} \boldsymbol{x}_i^{(j)}\Big).$$

From definition for $(\boldsymbol{x}, y) \sim \mathcal{P}_r$, we must have $y \sim L^{-1}\sum_{\boldsymbol{v}\in\mathcal{V}} \mathcal{N}(0, ||\boldsymbol{v}||_0^2 + \sigma^2)$. Therefore, we must have $\mathbb{E}y^2 \leq k + \sigma^2$ since $\boldsymbol{v} \in \{0,1\}^n, ||\boldsymbol{v}||_0 \leq k$ for all $\boldsymbol{v} \in \mathcal{V}$. By using Gaussian concentration inequalities, we must have $\Pr(|y| > t) \leq \exp(-t^2/2(k+\sigma^2))$. Therefore, with probability $1 - n^{-10}$, we have $|y| < 20\sqrt{k+\sigma^2}\log n$. Similarly, with probability $1 - n^{-10}$, $|\boldsymbol{x}_i|$ is bounded from above by $20\log n$. We take a union bound over all $|C|+1$ random variables and all m samples to infer that $\left(y^{(j)}\right)^{|\mathcal{C}|}\Big(\prod_{i\in\mathcal{C}} \boldsymbol{x}_i^{(j)}\Big)$ is bounded within a ball of radius $O((k+\sigma^2)^{|C|/2}(\log n)^{2|C|})$ with probability at least $1 - O(m|C|n^{-10})$. Subsequently, we use Hoeffding's inequality (see Lemma C.2) to say that

$$\begin{aligned}&\Pr\Big(\left|\frac{1}{m}\cdot\sum_{j=1}^{m}\left(y^{(j)}\right)^{|\mathcal{C}|}\Big(\prod_{i\in\mathcal{C}}\boldsymbol{x}_i^{(j)}\Big) - \frac{\left|\bigcap_{i\in\mathcal{C}}\mathcal{S}(i)\right|}{L}\right| \geq \frac{1}{2L}\Big)\\ &\leq \exp\Big(-\Omega\Big(\frac{m}{L^2(k+\sigma^2)^{|C|/2}(\log n)^{2|C|}}\Big)\Big).\end{aligned}$$

Hence, with $m = O(L^2(k+\sigma^2)^{|C|/2}(\log n)^{2|C|}\log\gamma^{-1})$ samples, we can estimate $\left|\bigcap_{i\in\mathcal{C}}\mathcal{S}(i)\right|$ exactly with probability at least $1-\gamma$. □

We can now show the following result:

Theorem (Restatement of Theorem 5.4)**.** *Let $\mathcal{V}$ be a set of L unknown binary vectors in $\{0,1\}^n$. Then, with probability at least $1-\gamma$, there exists an algorithm (see Algorithms 5.7 and 5.3) that achieves Exact Support Recovery with*

$$O\left(L^2(k+\sigma^2)^{(\log L+1)/2}(\log n)^{2(\log L+1)}\log((n+(Lk)^{\log L})\gamma^{-1})\right)$$

samples generated according to $\mathcal{P}_r$.

Proof. The proof follows directly from Lemma 5.9 and Corollary 5.1. □

Separability Assumption on Parameters Below, we show that if Assumption 5.3 is satisfied, then we can recover the support of the unknown vectors. We start with the following theorem:

Algorithm 5.8 ESTIMATE IF $|\bigcap_{i\in\mathcal{C}} \mathcal{S}(i) > 0|$ IN MLR SETTING

Require: Samples $(\boldsymbol{x}^{(1)}, y^{(1)}), (\boldsymbol{x}^{(2)}, y^{(2)}), \ldots, (\boldsymbol{x}^{(m)}, y^{(m)}) \sim \mathcal{P}_r$. Set $\mathcal{C} \subseteq [n]$.

1: If $\frac{2L}{m} \cdot \sum_{j=1}^{m} \left(y^{(j)}\right)^{|\mathcal{C}|} \left(\prod_{i\in\mathcal{C}} \boldsymbol{x}_i^{(j)}\right) \geq \alpha_{|\mathcal{C}|}$, return True else return False.

Theorem (Restatement of Theorem 5.6)**.** *Suppose the following conditions are satisfied:*

1. *All unknown vectors in $\mathcal{V}$ are bounded within a ball of radius R i.e. $\left|\left|\boldsymbol{v}^{(i)}\right|\right|_2 \leq R$ for all $i \in [L]$.*
2. *Assumption 5.3 is satisfied by the set of unknown vectors $\mathcal{V}$.*

Accordingly, with probability at least $1-\gamma$, there exists an algorithm (see Algorithms 5.8 and 5.3) that achieves Deduplicated support recovery using

$$O(L^2(R^2+\sigma^2)^{L/2}(\log n)^{2L}\log((n+(Lk)^L)\gamma^{-1})/\alpha_L^2)$$

samples from $\mathcal{P}_r$.

Proof. Again, we look at the random variable $y^{|\mathcal{C}|} \cdot \left(\prod_{i\in\mathcal{C}} \boldsymbol{x}_i\right)$ where $(\boldsymbol{x}, y) \sim \mathcal{P}_r$ and therefore, we must have

$$\mathbb{E}_{(\boldsymbol{x},y)\sim\mathcal{P}_r} y^{|\mathcal{C}|} \cdot \left(\prod_{i\in\mathcal{C}} \boldsymbol{x}_i\right) = \mathbb{E}_{\boldsymbol{x},\zeta} \ell^{-1} \sum_{\boldsymbol{v}\in\mathcal{V}} \left(\prod_{i\in\mathcal{C}} \boldsymbol{x}_i\right) \cdot \left(\langle \boldsymbol{v}, \boldsymbol{x}\rangle + \zeta\right)^{|\mathcal{C}|}$$

$$\mathbb{E}y^{|\mathcal{C}|} \cdot \Big(\prod_{i \in \mathcal{C}} \boldsymbol{x}_i \Big) = \frac{1}{L} \sum_{\boldsymbol{v} \in \mathcal{V}} \Big(\prod_{i \in \mathcal{C}} \mathbb{E}\boldsymbol{x}_i^2 \cdot \boldsymbol{v}_i \Big) = \frac{1}{L} \sum_{\boldsymbol{v} \in \mathcal{V}} \Big(\prod_{j \in \mathcal{C}} \boldsymbol{v}_j \Big).$$

Notice that $\mathbb{E}y^{|\mathcal{C}|} \cdot \Big(\prod_{i \in \mathcal{C}} \boldsymbol{x}_i \Big) = 0$ if $\big| \bigcap_{i \in \mathcal{C}} \mathcal{S}(i) \big| = 0$ and $\Big| \mathbb{E}y^{|\mathcal{C}|} \cdot \Big(\prod_{i \in \mathcal{C}} \boldsymbol{x}_i \Big) \Big| \geq \alpha_{|\mathcal{C}|}/L$ otherwise (by using Assumption 5.3). We estimate $\mathbb{E}y^{|\mathcal{C}|} \cdot \Big(\prod_{i \in \mathcal{C}} \boldsymbol{x}_i \Big)$ by computing the following sample average

$$\frac{L}{m} \cdot \sum_{j=1}^{m} \Big(y^{(j)} \Big)^{|\mathcal{C}|} \Big(\prod_{i \in \mathcal{C}} \boldsymbol{x}_i^{(j)} \Big).$$

From the definition of $\mathcal{P}_r$, we must have $y \sim L^{-1} \sum_{\boldsymbol{v} \in \mathcal{V}} \mathcal{N}(0, ||\boldsymbol{v}||_2^2 + \sigma^2)$. Therefore, we have that $\mathbb{E}y^2 \leq R^2 + \sigma^2$ since $||\boldsymbol{v}||_2 \leq R$ for all $\boldsymbol{v} \in \mathcal{V}$ from the statement of the Theorem. By using Gaussian concentration inequalities, we must have $\Pr(|y| > t) \leq \exp(-t^2/2(R^2 + \sigma^2))$. Therefore, with probability $1 - n^{-10}$, we have $|y| < 20\sqrt{R^2 + \sigma^2} \log n$. Similarly, with probability $1 - n^{-10}$, $|\boldsymbol{x}_i|$ is bounded from above by $20 \log n$. We take a union bound over all $|C| + 1$ random variables and all m samples to infer that $\Big(y^{(j)} \Big)^{|\mathcal{C}|} \Big(\prod_{i \in \mathcal{C}} \boldsymbol{x}_i^{(j)} \Big)$ is bounded within a ball of radius $O((R^2 + \sigma^2)^{|C|/2} (\log n)^{2|C|})$ with probability at least $1 - O(m|C|n^{-10})$. Subsequently, we use Hoeffding's inequality (see Lemma C.2) to say that

$$\begin{aligned} &\Pr \Big(\Big| \frac{1}{m} \cdot \sum_{j=1}^{m} y^{(j)\,|\mathcal{C}|} \Big(\prod_{i \in \mathcal{C}} \boldsymbol{x}_i^{(j)} \Big) - \frac{1}{L} \sum_{\boldsymbol{v} \in \mathcal{V}} \Big(\prod_{j \in \mathcal{C}} \boldsymbol{v}_j \Big) \Big| \geq \frac{\alpha_{|\mathcal{C}|}}{2L} \Big) \\ &\leq \exp \Big(- \Omega \Big(\frac{m \alpha_{|\mathcal{C}|}^2}{L^2 (R^2 + \sigma^2)^{|C|/2} (\log n)^{2|C|}} \Big) \Big). \end{aligned}$$

Hence, with $m = O(L^2 (R^2 + \sigma^2)^{|C|/2} (\log n)^{2|C|} \log \gamma^{-1} / \alpha_{|\mathcal{C}|}^2)$ samples, we can estimate if $\big| \bigcap_{i \in \mathcal{C}} \mathcal{S}(i) \big| > 0$ or not correctly with probability at least $1 - \gamma$. The proof now follows directly from using Corollary 5.2. □

Corollary (Restatement of Corollary 5.4)**.** *Consider a set of L unknown vectors $\mathcal{V}$ that satisfies Assumptions 5.1 and furthermore, every non-zero entry in all the*

unknown vectors is positive ($\boldsymbol{v}_i \geq 0$ for all $i \in [n], \boldsymbol{v} \in \mathcal{V}$). In that case, Assumption 5.3 is satisfied with $\alpha_{|\mathcal{C}|} \geq \delta^{|\mathcal{C}|}$. Accordingly, there exists an algorithm that achieves Deduplicated support recovery with probability at least $1 - \gamma$ using

$$O(L^2(R^2 + \sigma^2)^{L/2}(\log n/\delta)^{2L} \log((n + (Lk)^L)\gamma^{-1}))$$

samples from $\mathcal{P}_r$.

Proof. Note that when all the unknown vectors in set $\mathcal{V}$ are non-negative, it must happen that for each set $\mathcal{C} \subseteq [n]$, $\left|\sum_{\boldsymbol{v} \in \mathcal{V}} \left(\prod_{j \in \mathcal{C}} \boldsymbol{v}_j\right)\right| \geq \alpha_{|\mathcal{C}|}$ is a sum of positive terms (provided is it non-zero) each of which is at least $\delta^{|\mathcal{C}|}$. Therefore, it must happen that $\alpha_{|\mathcal{C}|} \geq \delta^{|\mathcal{C}|}$. The above argument also holds true when all the unknown vectors in set $\mathcal{V}$ are non-positive. We can directly use Theorem 5.6 to arrive at the statement of the corollary. □

Corollary (Restatement of Corollary 5.5)**.** *If all non-zero entries in the set of unknown vectors $\mathcal{V}$ are sampled i.i.d according to $\mathcal{N}(0, \nu^2)$, then with probability $1 - \eta$, Assumption 5.3 is satisfied with $\alpha_{|\mathcal{C}|} \geq \delta_{|\mathcal{C}|}^{|\mathcal{C}|}$ where*

$$\delta_{|\mathcal{C}|} = \left(\sqrt{\frac{\pi}{8}} \frac{\nu\eta}{L|\mathcal{C}|(Lk)^{|\mathcal{C}|}}\right).$$

Conditioned on this event, there exists an Algorithm that achieves Deduplicated support recovery with probability at least $1 - \gamma$ using

$$O(L^2(R^2 + \sigma^2)^{L/2}(\log n)^{2L} \log((n + (Lk)^L)\gamma^{-1})/\delta_L^2)$$

samples from $\mathcal{P}_r$.

Proof. For a fixed set $\mathcal{C} \subseteq [n]$, consider the random variable $\sum_{\boldsymbol{v} \in \mathcal{V}} \left(\prod_{j \in \mathcal{C}} \boldsymbol{v}_j\right)$. For each vector $\boldsymbol{v} \in \mathcal{V}$ such that $\prod_{j \in \mathcal{C}} \boldsymbol{v}_j \neq 0$, we denote the minimum index $i \in \mathcal{C}$ such

that $\boldsymbol{v}_i \neq 0$ by $i^\star$ and therefore $\boldsymbol{v}_{i^\star} \sim \mathcal{N}(0, \nu^2)$. Now, for each $\boldsymbol{v} \in \mathcal{V}$, let us condition on a fixed realization of non-zero indices of $\boldsymbol{v}$ in $\mathcal{C}$ other than $i^\star$. Let $\mathcal{V}_{\mathcal{C}} \subseteq \mathcal{V}$ be the set of vectors such that $\prod_{j \in \mathcal{C}} \boldsymbol{v}_j \neq 0$. Therefore, we must have

$$\sum_{\boldsymbol{v} \in \mathcal{V}} \Big(\prod_{j \in \mathcal{C}} \boldsymbol{v}_j \Big) \mid \boldsymbol{v}_j \text{ for all } j \in \mathcal{C} \setminus i^\star, \boldsymbol{v} \in \mathcal{V}_{\mathcal{C}} \sim \mathcal{N}\Big(0, \nu^2 \sum_{\boldsymbol{v} \in \mathcal{V}_{\mathcal{C}}} \prod_{j \in \mathcal{C} \setminus i^\star} \boldsymbol{v}_j^2\Big). \tag{5.4}$$

Therefore, conditioned on $\boldsymbol{v}_j$ for all $j \in \mathcal{C} \setminus i^\star, \boldsymbol{v} \in \mathcal{V}_{\mathcal{C}}$, by standard Gaussian anti-concentration inequality (see Lemma C.4), we must have with probability $1 - \rho$,

$$\left| \sum_{\boldsymbol{v} \in \mathcal{V}} \Big(\prod_{j \in \mathcal{C}} \boldsymbol{v}_j \Big) \right| \geq \sqrt{\frac{\pi}{8}} \rho \nu \sqrt{\sum_{\boldsymbol{v} \in \mathcal{V}_{\mathcal{C}}} \prod_{j \in \mathcal{C} \setminus i^\star} \boldsymbol{v}_j^2}. \tag{5.5}$$

for each vector $\boldsymbol{v} \in \mathcal{V}_{\mathcal{C}}$, we must have with probability at least $1 - (|\mathcal{C}| - 1)\rho$ that

$$\left| \prod_{j \in \mathcal{C} \setminus i^\star} \boldsymbol{v}_j \right| \geq \Big(\sqrt{\frac{\pi}{8}} \rho \nu \Big)^{(|\mathcal{C}|-1)}. \tag{5.6}$$

By taking a union bound, we can conclude that with probability at least $1 - L\rho$, we must have

$$\left| \sum_{\boldsymbol{v} \in \mathcal{V}} \Big(\prod_{j \in \mathcal{C}} \boldsymbol{v}_j \Big) \right| \geq \Big(\sqrt{\frac{\pi}{8}} \rho \nu \Big)^{|\mathcal{C}|}$$

since there exists at least one vector $\boldsymbol{v} \in \mathcal{V}_{\mathcal{C}}$ such that equation 5.6 holds true for $\boldsymbol{v}$. Next, after taking another union bound over all subsets of size $|\mathcal{C}|$ restricted to the union of support (at most $(Lk)^{|\mathcal{C}|}$ of them), we have that with probability $1 - |\mathcal{C}|(Lk)^{|\mathcal{C}|}\rho$,

$$\left| \sum_{\boldsymbol{v} \in \mathcal{V}} \Big(\prod_{j \in \mathcal{C}} \boldsymbol{v}_j \Big) \right| \geq \Big(\sqrt{\frac{\pi}{8}} \rho \nu \Big)^{|\mathcal{C}|}.$$

Subsequently, we have with probability at least $1 - \eta/L$

$$\left| \sum_{\boldsymbol{v} \in \mathcal{V}} \Big(\prod_{j \in \mathcal{C}} \boldsymbol{v}_j \Big) \right| \geq \Big(\sqrt{\frac{\pi}{8}} \frac{\nu\eta}{L|\mathcal{C}|(Lk)^{|\mathcal{C}|}} \Big)^{|\mathcal{C}|}.$$

After taking a final union bound over $|\mathcal{C}| \leq L$ and subsequently using Theorem 5.6, we complete the proof of the corollary. $\square$

CHAPTER 6

CONCLUSION

In this thesis, we studied a number of problems under the big umbrella of *mixture models* or *latent variable models* and experimentally validated some of our algorithms on real world data-sets. We provided a variety of technical tools to study latent variable models for both parameter estimation in different mixture models and for support recovery in mixture models with sparse parameters. In particular, we studied three different mixture models namely 1) Mixtures of Linear Regression 2) Mixtures of Linear Classifiers and 3) Mixtures of Distributions. We studied two different settings for these aforementioned mixture models namely 1) the experimental design setting or the query based setting where we can design the covariates and query the corresponding response from an oracle 2) unsupervised setting where the data is sampled from a distribution.

We provide two general frameworks for theoretically studying mixtures that are quite powerful and can be extended to other statistical reconstruction problems as well. First, we demonstrated the use of complex analytic tools to prove lower bounds on the total variation distance between any two mixtures corresponding to many different families of distributions. As an example, for a pair of Gaussian mixtures with shared component variance, we provide guarantees on the total variation distance as a function of the largest gap (among the two mixtures) between the component means. Although intuitive, such a characterization was missing despite a vast literature on the total variation distance between mixtures of Gaussians. The complex analytic tools in this work are quite elementary, and there is significant room for development.

These tools may be helpful in proving bounds on statistical distance between more diverse distributions. We utilized these lower bounds on the total variation distance to convert them into sample complexity guarantees via the minimum distance estimator or the Scheffe estimator. We also used this framework for the Trace Reconstruction problem and for parameter estimation in Mixtures of Linear Regression. It would also be useful and interesting to provide matching upper bounds on the total variation distance between mixtures of different families of distributions to complement our lower bounds. Extending our results to more general mixtures with minimal restrictive assumptions will be of significant interest to both the statistics and machine learning communities.

Secondly, we described several novel techniques for support recovery including low rank tensor decomposition that can be used for other applications as well with significant advantages. Note that low rank tensor decomposition [6] is already a very well known and widely used technique for parameter estimation in latent variable models (that include mixture models as a special case). However the techniques in [6] are only able to design and use tensors of order 3 because, for higher order tensors there does not exist any algorithm that can recover the low rank decomposition uniquely even if it is known to exist. For the problem of support recovery, we design a completely new tensor that has many advantages compared to the techniques of [6]:

1. The tensors that we design are integral (i.e. all its entries are integers) and therefore, it is possible to recover the tensor exactly by correcting a small amount of estimation error stemming from sample estimates.

2. For tensors of order $w > 3$, there is no known efficient algorithm that can recover the correct solution even if its existence and uniqueness is known. On the other hand, since the tensor is recovered exactly, we can exhaustively search over all possibilities (i.e. do a brute force algorithm) to recover the unknown parameters.

In conjunction with these techniques for support recovery, we also use a plethora of different techniques in the many settings that we study to compute the sufficient statistics for support recovery efficiently. Some of these include polynomial identities and combinatorial designs. We believe that these new ideas for support recovery will be applicable for support recovery in other sparse latent generative models as well.

While there is still much to be explored regarding mixture models in machine learning both theoretically and practically, we hope the results in this thesis have made a compelling case for this paradigm. We look forward to future advances in this direction and a deeper understanding of this framework.

APPENDIX A

MISSING PROOFS IN CHAPTER 2

A.1 Proof of Theorem 2.7

Let $\mathbf{a}$ be the characteristic vector of a subset $S \subset \mathcal{U}$. Let $s_\ell = m_\ell(S)$ on this set and let $\mathbf{s} = (s_0, s_1, \ldots, s_{k-1})$. We need to prove $\mathbf{a}$ is uniquely determined by $\mathbf{s}$.

Let us define

$$n_{i,p}(\mathbf{a}) := \sum_{r \equiv_p i} a_r \pmod{p} .$$

Claim A.1. *For a prime number p and $i \not\equiv_p 0$, we have*

$$n_{i,p}(\mathbf{a}) \equiv_p s_0 - \sum_j \binom{p-1}{j} s_j (-i)^{p-1-j}.$$

Proof.

$$n_{i,p}(\mathbf{a}) = \sum_{r \equiv_p i} a_r \pmod{p}$$

Recall that Fermat's theorem ([73]) says that for any prime p and any number $\alpha \not\equiv_p 0$, we must have that $\alpha^{p-1} \equiv_p 1$. Hence, for a prime number p and some number $i \not\equiv_p 0$, we have

$$s_0(\mathbf{a}) - \sum_j \binom{p-1}{j} s_j (-i)^{p-1-j} \equiv_p \sum_r a_r - \sum_j \binom{p-1}{j} \sum_r a_r r^j (-i)^{p-1-j}$$

$$\equiv_p \sum_r a_r - \sum_r a_r \sum_j \binom{p-1}{j} r^j (-i)^{p-1-j}$$
$$\equiv_p \sum_r a_r - \sum_r a_r (r-i)^{p-1}$$
$$\equiv_p \sum_{r \equiv_p i} a_r \equiv_p n_{i,p}(\mathbf{a}) \ .$$

□

Since the value of $n_{i,p}$ is at most $\lceil qn/p \rceil$, we can obtain the value of $n_{i,p}$ exactly if p is chosen to be greater than $\sqrt{qn}$. Now, let us denote the vector $\mathbf{v}_{i,p} \in \mathbb{F}_q^n$ where the ℓth entry is

$$\boldsymbol{v}_{i,p}[\ell] = \begin{cases} 1 & \text{if } \ell \equiv_p i \\ 0 & \text{otherwise} \end{cases} \ .$$

Therefore, consider two different subsets $S, S' \subset \mathcal{U}$ and assume that their characteristic vectors are $\mathbf{a}$ and $\mathbf{b}$ respectively. Therefore, if $\mathbf{a}$ and $\mathbf{b}$ both give rise to the same value of $\mathbf{s}$, then $\mathbf{a}.\mathbf{v}_{i,p} = \mathbf{b}.\mathbf{v}_{i,p}$. Hence, if the set of vectors

$$\mathcal{S} = \{\boldsymbol{v}_{i,p} \mid \sqrt{qn} \leq p \leq k, 0 \leq i \leq p-1, p \text{ prime}\}$$

spans $\mathbb{F}_q^n$, then it must imply that $\mathbf{a} = \mathbf{b}$ and our proof will be complete. Consider a subset $\mathcal{T} \subset \mathcal{S}$ defined by

$$\mathcal{T} = \{\boldsymbol{v}_{i,p} \mid \sqrt{qn} \leq p \leq k, 1 \leq i \leq p-1, p \text{ prime}\}$$

Now, there are two possible cases. First, let us assume that the vectors in $\mathcal{T}$ are not all linearly independent in $\mathbb{F}_q$. In that case, we must have a set of tuples $(i_1, p_1), (i_2, p_2), \ldots, (i_m, p_m)$ such that

$$\sum_{j=1}^{m} \alpha_j \mathbf{v}_{(i_j,p_j)} \equiv_q 0 \tag{A.1}$$

where $0 \neq \alpha_j \in \mathbb{F}_q$ for all j. Now, by the Chinese Remainder Theorem, we can find an integer r such that $r \equiv_{p_1} i_1$ and $r \equiv_{p_j} 0$ for all $p_j \neq p_1$. Define an infinite dimensional vector $\tilde{\mathbf{v}}$ where the ℓth entry is

$$\tilde{\mathbf{v}}[\ell] = \sum_{j=1}^{m} \alpha_j \mathbb{1}\Big[\ell \equiv_{p_j} i_j\Big]$$

Since, $i_j \not\equiv_{p_j} 0$, it is evident that $\tilde{\mathbf{v}}[r] \not\equiv_q 0$ Now, let s be the smallest number such that $\tilde{\mathbf{v}}[s] \neq 0$ and $s > n$ because of our assumption in Eq. A.1. Now consider the vector $\mathbf{v}_t$ where

$$\mathbf{v}_t = \sum_{j=1}^{m} \alpha_j \mathbf{v}_{i_j - s + t, p_j}$$

Now, $\mathbf{v}_t^i = 0$ for all $i < t$ and $\mathbf{v}_t^t \neq 0$. Hence, the set $\{\mathbf{v}_t\}_{t=1}^n$ are in the span of $\mathcal{S}$ and also span $\mathbb{F}_q^n$.

For the second case, let us assume that the vectors in $\mathcal{T}$ are linearly independent. We require the size of $\mathcal{T} > n$ so that the vectors in $\mathcal{T}$ span $\mathbb{F}_q^n$. From the prime number theorem we know that

$$\sum_{p \text{ prime}: p<x} p \sim \frac{x^2}{2 \log x}$$

and hence we simply need that

$$\frac{k^2}{2 \log k} - \frac{qn}{\log n} > n \ .$$

Therefore, $k > (1 + o(1))\sqrt{qn \log qn}$ is sufficient.

A.2 Proof of Lemma 2.9

We use case analysis on different orderings of the means and their separations.

Claim A.2. *For any $t > 0$ such that $t(\mu_1 - \mu_0), t(\mu_1' - \mu_0), t(\mu_1' - \mu_0) \in [0, \frac{\pi}{4}]$, when $\mu_1' > \mu_1$,*

$$\left| e^{it\mu_0} + e^{it\mu_1} - e^{it\mu_0'} - e^{it\mu_1'} \right| \geq \frac{t\delta_2}{2\sqrt{2}}.$$

Proof. Assume that $\mu_1' - \mu_1 \geq \mu_0' - \mu_0$, and recall that $\delta_2 = \max(|\mu_0' - \mu_0|, |\mu_1 - \mu_1'|)$. First, we factor out the lowest common exponent to see that

$$\left| e^{it\mu_0} + e^{it\mu_1} - e^{it\mu_0'} - e^{it\mu_1'} \right| = \left| 1 + e^{it(\mu_1 - \mu_0)} - e^{it(\mu_0' - \mu_0)} - e^{it(\mu_1' - \mu_0)} \right|.$$

Let us denote $\phi_1 = \mu_1 - \mu_0$, $\phi_0' = \mu_0' - \mu_0$ and $\phi_1' = \mu_1' - \mu_0$. The following inequalities hold:

$$\begin{aligned}
\left| 1 + e^{it\phi_1} - e^{it\phi_0'} - e^{it\phi_1'} \right| &\geq |\sin(t\phi_1) - \sin(t\phi_0') - \sin(t\phi_1')| && [|z| \geq |\mathrm{Im}(z)|] \\
&\geq -\sin(t\phi_1) + \sin(t\phi_0') + \sin(t\phi_1') && [\text{Remove } |\cdot|] \\
&\geq -\sin(t\phi_1) + \sin(t(\phi_1 + (\phi_1' - \phi_1))) && [\sin(t\phi_0') \geq 0] \\
&= 2\sin\left(t\frac{\phi_1' - \phi_1}{2}\right)\cos\left(t\left(\phi_1 + \frac{\phi_1' - \phi_1}{2}\right)\right) \\
&\geq \frac{1}{2\sqrt{2}} t\delta_2.
\end{aligned}$$

In the last line, we use that $\cos\left(t\left(\phi_1 + \frac{\phi_1' - \phi_1}{2}\right)\right) \geq 1/\sqrt{2}$ and $\sin\left(t\frac{\phi_1' - \phi_1}{2}\right) \geq \frac{t\delta_2}{4}$, where the former follows from the fact that

$$0 \leq t\left(\phi_1 + \frac{\phi_1' - \phi_1}{2}\right) = \frac{t(\phi_1 + \phi_1')}{2} = \frac{1}{2}\Big(t(\mu_1 - \mu_0) + t(\mu_1' - \mu_0)\Big) \leq \frac{\pi}{4}$$

and the latter follows from $\sin(x) \geq x/2$ for $x \in \mathbb{R}$.

If $\mu_0' - \mu_0 > \mu_1' - \mu_1$, then we can use a similar string of inequalities by using the fact that

$$\left|e^{it\mu_0} + e^{it\mu_1} - e^{it\mu_0'} - e^{it\mu_1'}\right| = \left|1 + e^{it(\mu_0-\mu_1)} - e^{it(\mu_0'-\mu_1)} - e^{it(\mu_1'-\mu_1)}\right|.$$

We denote $\phi_0 = \mu_0 - \mu_1$, $\phi_0' = \mu_0' - \mu_1$ and $\phi_1' = \mu_1' - \mu_1$. Note that all the ϕ are negative and $\phi_0 > \phi_0' > \phi_1'$. The following holds:

$$\begin{aligned}
&\left|e^{it\phi_0} + 1 - e^{it\phi_0'} - e^{it\phi_1'}\right| \\
&\geq |\sin(t\phi_0) - \sin(t\phi_0') - \sin(t\phi_1')| && [|z| \geq |\mathrm{Re}(z)|] \\
&= |-\sin(t|\phi_0|) + \sin(t|\phi_0'|) + \sin(t|\phi_1'|)| && [\sin(\cdot) \text{ odd}] \\
&= -\sin(t|\phi_0|) + \sin(t|\phi_0'|) + \sin(t|\phi_1'|) && [\text{Remove } |\cdot|] \\
&\geq -\sin(t|\phi_0|) + \sin(t(|\phi_0| + (|\phi_0'| - |\phi_0|))) && [\sin(t|\phi_0'|) \geq 0] \\
&= 2\sin\left(\frac{t(|\phi_0'| - |\phi_0|)}{2}\right)\cos\left(t\left(|\phi_0| + \frac{|\phi_0'| - |\phi_0|}{2}\right)\right) \\
&\geq \frac{1}{2\sqrt{2}} t\delta_2.
\end{aligned}$$

In the last line, we use that $\sin\left(\frac{t(|\phi_0'|-|\phi_0|)}{2}\right) \geq \frac{t\delta_2}{4}$ and $\cos\left(t\left(|\phi_0| + \frac{|\phi_0'|-|\phi_0|}{2}\right)\right) \geq \frac{\sqrt{2}}{2}$. □

Claim A.3. *For $t > 0$ such that $t(\mu_1 - \mu_0), t(\mu_1' - \mu_0), t(\mu_1' - \mu_0) \in [0, \frac{\pi}{4}]$, if both $\mu_0', \mu_1' \in [\mu_0, \mu_1]$, then*

$$\left|e^{it\mu_0} + e^{it\mu_1} - e^{it\mu_0'} - e^{it\mu_1'}\right| \geq \max\left(\frac{t^2(\delta_1 - \delta_4)\delta_4}{2}, \frac{t\delta_3}{4\sqrt{2}}\right).$$

Proof. First, we show the left hand side of the inequality in the claim statement is at least $t\delta_3/(4\sqrt{2})$.

Assume that $\mu_0' - \mu_0 = \delta_2$, recalling that $\delta_2 = \max(|\mu_0' - \mu_0|, |\mu_1 - \mu_1'|)$. We factor out the lowest common exponent to see that

$$\left|e^{it\mu_0} + e^{it\mu_1} - e^{it\mu_0'} - e^{it\mu_1'}\right| = \left|1 + e^{it(\mu_1-\mu_0)} - e^{it(\mu_0'-\mu_0)} - e^{it(\mu_1'-\mu_0)}\right|.$$

Let us denote $\phi_1 = \mu_1 - \mu_0$, $\phi_0' = \mu_0' - \mu_0$ and $\phi_1' = \mu_1' - \mu_0$. To prove following inequalities, we need two facts. We use **Fact I** that $\frac{\partial}{\partial x}(\sin(x-y) - \sin(x)) = \cos(x-y) - \cos(x) \geq 0$ for $\frac{\pi}{4} \geq x \geq y \geq 0$. In particular, taking $x = \phi_1$ and $y = \mu_1 - \mu_1'$, the inequality is increasing with respect to ϕ_1, so so we can lower bound the function at $\phi_1 = \mu_1 - \mu_1' + \phi_0'$. Additionally, we use **Fact II** that $-\sin(x+y) + 2\sin(x) \geq \sin((x-y)/2)\cos(y/2)$ for $0 \leq y \leq x \leq \pi/4$, for the choice of $x = \phi_0'$ and $y = \mu_1 - \mu_1'$. Then, we have that

$$\begin{aligned}
&\left|1 + e^{it\phi_1} - e^{it\phi_0'} - e^{it\phi_1'}\right| \\
&\geq |\sin(t\phi_1) - \sin(t\phi_0') - \sin(t\phi_1')| && [|z| \geq |\mathrm{Im}(z)|] \\
&\geq -\sin(t\phi_1) + \sin(t\phi_0') + \sin(t\phi_1') && [\text{Remove } |\cdot|] \\
&\geq -\sin(t\phi_1) + \sin(t\phi_0') + \sin(t(\phi_1 - (\mu_1 - \mu_1'))) \\
&\geq -\sin(t(\mu_1 - \mu_1' + \phi_0')) + 2\sin(t\phi_0') && [\textbf{Fact I} \text{ above}] \\
&\geq \sin\left(\frac{t(-\mu_1 + \mu_1' + \mu_0' - \mu_0)}{2}\right)\cos\left(\frac{t(\mu_1 - \mu_1')}{2}\right) && [\textbf{Fact II} \text{ above}] \\
&\geq \frac{t\delta_3}{4\sqrt{2}} && [\sin(x) \geq x/2; \\
& && \cos(\cdot) \geq \sqrt{2}/2]
\end{aligned}$$

Now, we assume that $\mu_0' - \mu_0 < \mu_1 - \mu_1'$ and factor out $e^{it\mu_1}$:

$$\left|e^{it\mu_0} + e^{it\mu_1} - e^{it\mu_0'} - e^{it\mu_1'}\right| = \left|e^{it(\mu_0-\mu_1)} + 1 - e^{it(\mu_0'-\mu_1)} - e^{it(\mu_1'-\mu_1)}\right|.$$

As in the proofs of other claims, we let $\phi_0 = \mu_0 - \mu_1$, $\phi_0' = \mu_0' - \mu_1$ and $\phi_1' = \mu_1' - \mu_1$. Then, we have

$$
\begin{aligned}
&\left|e^{it\phi_0} + 1 - e^{it\phi_0'} - e^{it\phi_1'}\right| \\
&\geq |\sin(t\phi_0) - \sin(t\phi_0') - \sin(t\phi_1')| && [|z| \geq |\mathrm{Im}(z)|] \\
&\geq |-\sin(t|\phi_0|) + \sin(t|\phi_0'|) + \sin(t|\phi_1'|)| && [\sin(\cdot) \text{ odd}] \\
&\geq -\sin(t|\phi_0|) + \sin(t|\phi_0'|) + \sin(t|\phi_1'|) && [\text{Remove } |\cdot|] \\
&\geq -\sin(t|\phi_0|) + \sin(t|\phi_0| - |\mu_0 - \mu_0'|) + \sin(t|\phi_1'|) \\
&\geq -\sin(t(|\phi_1'| + |\mu_0 - \mu_0'|)) + 2\sin(t|\phi_1'|) && [\textbf{Fact I} \text{ above}] \\
&\geq \sin\left(\frac{t(|\phi_1'| - |\mu_0 - \mu_0'|)}{2}\right)\cos\left(\frac{t|\mu_0 - \mu_0'|}{2}\right) && [\textbf{Fact II} \text{ above}] \\
&\geq \frac{t}{4\sqrt{2}} \cdot (|\phi_1'| - |\mu_0 - \mu_0'|) = \frac{t}{4\sqrt{2}} \cdot \delta_3 && [\sin(x) \geq x/2; \\
& && \cos(\cdot) \geq \sqrt{2}/2]
\end{aligned}
$$

In the application of **Fact I**, we let $|\phi_0|$ be as small as possible, choosing $|\phi_0| = |\phi_1'| + |\mu_0 - \mu_0'|$.

Next, we show the left hand side of the inequality in the claim statement is at least $t^2(\delta_1 - \delta_4)\delta_4/2$. Assume that $\mu_0' - \mu_0 \leq \mu_1 - \mu_1'$. First, we factor out the lowest common exponent to see

$$
\left|e^{it\mu_0} + e^{it\mu_1} - e^{it\mu_0'} - e^{it\mu_1'}\right| = \left|1 + e^{it(\mu_1 - \mu_0)} - e^{it(\mu_0' - \mu_0)} - e^{it(\mu_1' - \mu_0)}\right|.
$$

Again, we denote $\phi_1 = \mu_1 - \mu_0$, $\phi_0' = \mu_0' - \mu_0$ and $\phi_1' = \mu_1' - \mu_0$. The following holds:

$$
\begin{aligned}
&\left|1 + e^{it\phi_1} - e^{it\phi_0'} - e^{it\phi_1'}\right| \\
&\geq |\mathrm{Re}(1 + e^{it\phi_1} - e^{it\phi_0'} - e^{it\phi_1'})| \\
&= |1 + \cos(t\phi_1) - \cos(t\phi_0') - \cos(t\phi_1')| && [|z| \geq |\mathrm{Re}(z)|]
\end{aligned}
$$

$$
\begin{aligned}
&\geq -1 - \cos(t\phi_1) + \cos(t\phi_0') \\
&\quad + \cos(t(\phi_1 - (\mu_1 - \mu_1'))) && \text{[Remove } |\cdot|\text{]} \\
&\geq -1 - \cos(t\phi_1) + \cos(t\phi_0') + \cos(t(\phi_1 - \phi_0')) && [\mu_1 - \mu_1' \geq \phi_0'] \\
&\geq \frac{t^2(\phi_1 - \phi_0')\phi_0'}{2} \geq \frac{t^2(\delta_1 - \delta_4)\delta_4}{2} && \text{[\textbf{Fact III} below]}
\end{aligned}
$$

Recall that $\delta_1 = \max(|\mu_0 - \mu_1|, |\mu_0' - \mu_1'|)$,and $\delta_4 = \min(|\mu_0' - \mu_0|, |\mu_1' - \mu_1|)$, so in the above, $\delta_4 = \mu_0' - \mu_0 = \phi_0'$. The last line uses **Fact III** that $-1 - \cos(x) + \cos(y) + \cos(x - y) \geq (x - y)y/2$ for $0 \leq y \leq x \leq \pi/4$.

When $\mu_0' - \mu_0 > \mu_1 - \mu_1'$ we use the same trick as in the previous claims and factor out $e^{it\mu_1}$ instead of $e^{it\mu_0}$. In particular the following holds:

$$
\left| e^{it\mu_0} + e^{it\mu_1} - e^{it\mu_0'} - e^{it\mu_1'} \right| = \left| e^{it\mu_0 - \mu_1} + 1 - e^{it(\mu_0' - \mu_1)} - e^{it(\mu_1' - \mu_1)} \right|.
$$

Again, we denote $\phi_0 = \mu_0 - \mu_1$, $\phi_0' = \mu_0' - \mu_1$ and $\phi_1' = \mu_1' - \mu_1$. Note that all the ϕ are negative and $\phi_0 > \phi_0' > \phi_1'$. The following holds:

$$
\begin{aligned}
&\left| e^{it\phi_0} + 1 - e^{it\phi_0'} - e^{it\phi_1'} \right| \\
&\geq |\mathrm{Re}(e^{it\phi_0} + 1 - e^{it\phi_0'} - e^{it\phi_1'})| \\
&= |\cos(t\phi_0) + 1 - \cos(t\phi_0') - \cos(t\phi_1')| && [|z| \geq |\mathrm{Re}(z)|] \\
&= |\cos(t|\phi_0|) + 1 - \cos(t|\phi_0'|) - \cos(t|\phi_1'|)| && [\cos(\cdot) \text{ even}] \\
&\geq -\cos(t|\phi_0|) - 1 + \cos(t(|\phi_0| - |\mu_0' - \mu_0|)) && \text{[Remove } |\cdot|\text{]} \\
&\quad + \cos(t(|\phi_1'|)) \\
&\geq -\cos(t|\phi_0|) - 1 + \cos(t(|\phi_0| - |\phi_1'|)) \\
&\quad + \cos(t|\phi_1'|) && [\mu_0' - \mu_0 \geq |\phi_1'|]
\end{aligned}
$$

$$\geq \frac{t^2(|\phi_0| - |\phi_1'|)|\phi_1'|}{2} \geq \frac{t^2(\delta_1 - \delta_4)\delta_4}{2}. \qquad \text{[\textbf{Fact III} above]}$$

□

Proof of Lemma 2.11

Proof of Lemma 2.11. Define α and β such that $\mu_0' - \mu_0 = \alpha\sigma$ and $|\mu_1 - \mu_1'| = \beta\sigma$; note that by assumption $\alpha, \beta \leq 2$. For $x \in \mathbb{R}$, we use the notation $\widetilde{x}$ to denote the unique value such that $x = 2\pi kc\sigma + \widetilde{x}\sigma$, where $k \in \mathbb{Z}$ is a integer and $0 \leq \widetilde{x} < 2\pi c$. We prove this lemma with two cases, when $\mu_1' > \mu_1$ and when $\mu_1' \leq \mu_1$. Without loss of generality, we assume that $|\mu_0 - \mu_1| \geq 100\sigma$. Also, recall our assumption on the ordering of the unknown parameters that $\mu_0 \leq \min(\mu_1, \mu_0', \mu_1')$ and $\mu_0' \leq \mu_1'$.

Case 1 ($\mu_1' > \mu_1$): Here, we will choose $t = c\sigma$, where

$$c = \frac{\mu_1 - \mu_0}{2\pi\sigma\lfloor \frac{\mu_1 - \mu_0}{80\sigma/\pi} \rfloor}.$$

Then substituting in $t = 1/c\sigma$, we see that

$$e^{itx} = e^{it2\pi kc\sigma} e^{it\widetilde{x}\sigma} = e^{i2\pi k} e^{i\widetilde{x}/c} = e^{i\widetilde{x}/c}.$$

From the choice of c and the fact that $\lfloor x \rfloor \leq x$ and $x/2 \leq \lfloor x \rfloor$ for $x \geq 1$, we see $40/\pi^2 \leq c \leq 80/\pi^2$.

As before, let $\phi_1 = \mu_1 - \mu_0, \phi_0' = \mu_0' - \mu_0$ and $\phi_1' = \mu_1' - \mu_0$. We prove that the following hold:

$$\widetilde{\phi}_1 = 0$$
$$\frac{\pi^2\alpha}{80} \leq \frac{\widetilde{\phi}_0'}{c} = \frac{\alpha}{c} \leq \frac{\pi^2\alpha}{40} \leq \frac{\pi^2}{20}$$
$$\frac{\pi^2\beta}{80} \leq \frac{\widetilde{\phi}_1'}{c} = \frac{\beta}{c} \leq \frac{\pi^2\beta}{40} \leq \frac{\pi^2}{20}.$$

We prove these statements in order. To see that $\widetilde{\phi}_1 = 0$, the definitions of c and $\widetilde{\phi}_1$ imply that $\phi_1 = 2\pi\sigma k \frac{\phi_1}{2\pi\sigma\lfloor\phi_1\pi/(80\sigma)\rfloor} + \widetilde{\phi}_1\sigma$, for $k = \lfloor\phi_1\pi/(80\sigma)\rfloor$ and $\widetilde{\phi}_1 = 0$.

Next, since $\alpha\sigma = \phi'_0$, we can write $\alpha/c = 2\pi k + \widetilde{\phi}'_0/c$, and it would follow that $\alpha/c = \widetilde{\phi}'_0/c$ if $\phi'_0 < 2\pi\sigma c = \phi_1/\lfloor\phi_1\pi/(80\sigma)\rfloor$. Indeed this is the case, since

$$\phi_1/\lfloor\phi_1\pi/(80\sigma)\rfloor \geq 80\sigma/\pi > 2\sigma > \phi'_0.$$

Using the fact that $\widetilde{\phi}_1 = 0$, we will show $\widetilde{\phi}'_1/c = \beta/c$. We break up ϕ'_1 into $\phi_1 + \mu'_1 - \mu_1$, writing

$$\phi'_1 = \beta\sigma + \phi_1 = \beta\sigma + k\frac{\phi_1}{\lfloor\phi_1\pi/(80\sigma)\rfloor} + \widetilde{\phi}_1\sigma = \beta\sigma + k\frac{\phi_1}{\lfloor\phi_1\pi/(80\sigma)\rfloor},$$

for $k = \lfloor\phi_1\pi/(80\sigma)\rfloor$. If $\beta < 2\pi c$, then this choice of k is correct for the definition of $\widetilde{\phi}'_1$, and it follows that $\widetilde{\phi}'_1 = \beta$. This is indeed the case as $2\pi c = \frac{\phi_1}{\sigma\lfloor\phi_1\pi/(80\sigma)\rfloor} \geq 80/\pi > \beta$.

We use our lower bounds on $\widetilde{\phi}'_0/c$ and $\widetilde{\phi}'_1/c$ and the fact that $\widetilde{\phi}_1 = 0$ in the following:

$$\begin{aligned}
e^{-\frac{\sigma^2t^2}{2}}\left|e^{it\mu_0} + e^{it\mu_1} - e^{it\mu'_0} - e^{it\mu'_1}\right| &= e^{-\frac{\sigma^2t^2}{2}}\left|1 + e^{it\phi_1} - e^{it\phi'_0} - e^{it\phi'_1}\right| \\
&\geq e^{-\frac{1}{2c^2}}\left|\mathrm{Im}(1 + e^{i\widetilde{\phi}_1/c} - e^{i\widetilde{\phi}'_0/c} - e^{i\widetilde{\phi}'_1/c})\right| \\
&= e^{-\frac{1}{2c^2}}\left|\sin(\widetilde{\phi}_1/c) - \sin(\widetilde{\phi}'_0/c) - \sin(\widetilde{\phi}'_1/c)\right| \\
&= e^{-\frac{1}{2c^2}}\left(\sin(\widetilde{\phi}'_0/c) + \sin(\widetilde{\phi}'_1/c)\right) \\
&\geq e^{-1}\max(\sin(\widetilde{\phi}'_0/c), \sin(\widetilde{\phi}'_1/c)) \\
&\geq e^{-1}\max(\sin(\pi^2\alpha/80), \sin(\pi^2\beta/80)) \\
&\geq \frac{\pi^2\delta_2}{160e}.
\end{aligned}$$

Case 2 ($\mu'_1 \leq \mu_1$): First we consider the case when $\beta \leq \alpha$. Since $\mu_1 - \mu_0 \geq 100\sigma$, we must have $\mu_1 - \mu_0 \geq \mu_1 - \mu_0 - (\mu_1 - \mu'_1) \geq 100\sigma - 2\sigma = 98\sigma$. We choose $t = c\sigma$ for

$$c = \frac{\mu'_1 - \mu_0}{3\pi\sigma/2 + 2\pi\sigma\lfloor\frac{\mu'_1 - \mu_0}{80\sigma/\pi}\rfloor}.$$

As before, for any $x \in \mathbb{R}$ we write $x = 2\pi kc\sigma + \widetilde{x}\sigma$ where $k \in \mathbb{Z}$ is a positive integer and $0 < \widetilde{x} < 2\pi c$. From the choice of c and the fact that $\mu_1' - \mu_0 \geq 98\sigma$, $\frac{25}{\pi^2} \leq c \leq \frac{80}{\pi^2}$. Here we denote $\phi_1 = \mu_1 - \mu_1', \phi_0' = \mu_0' - \mu_0$ and $\phi_1' = \mu_1' - \mu_0$; note this is different than our previous ϕ definitions. We will show the following set of inequalities and equalities:

$$\begin{aligned} \frac{\widetilde{\phi_1'}}{c} &= \frac{3\pi}{2} \\ \frac{\pi^2\alpha}{80} \leq \frac{\widetilde{\phi_0'}}{c} &= \frac{\alpha}{c} \leq \frac{\pi^2\alpha}{25} \leq \frac{\pi^2}{12} \\ \frac{\pi^2\beta}{80} \leq \frac{\widetilde{\phi_1}}{c} &= \frac{\beta}{c} \leq \frac{\pi^2\beta}{25} \leq \frac{\pi^2}{12}. \end{aligned}$$

To see that $\widetilde{\phi_1'} = 3\pi/2$, observe first that $\phi_1'/(c\sigma) = 2\pi k + \widetilde{\phi_1'}/c$; then we can simplify $\phi_1'/(c\sigma)$ and write $\phi_1'/(c\sigma) = 3\pi/2 + 2\pi \lfloor \phi_1'\pi/(80\sigma) \rfloor$. Together these imply that $3\pi/2 + 2\pi \lfloor \phi_1'\pi/(80\sigma) \rfloor = 2\pi k + \widetilde{\phi_1'}/c$. Taking $k = \lfloor \phi_1'\pi/(80\sigma) \rfloor$, it follows that $\widetilde{\phi_1'} = 3\pi/2$.

Additionally, since $\alpha\sigma = \phi_0'$, we can write $\alpha/c = 2\pi k + \tilde{\phi_0'}/c$. It follows that $\alpha/c = \tilde{\phi_0'}/c$ if

$$\phi_0' < 2\pi\sigma c = 2\pi\sigma \frac{\phi_1'}{3\pi\sigma/2 + 2\pi\sigma\lfloor \phi_1'\pi/(80\sigma) \rfloor} = \frac{\phi_1'}{3/4 + \lfloor \phi_1'\pi/(80\sigma) \rfloor}.$$

Indeed this is the case, since if $\lfloor \phi_1'\pi/(80\sigma) \rfloor < 1/4$,

$$\frac{\phi_1'}{3/4 + \lfloor \phi_1'\pi/(80\sigma) \rfloor} > \phi_1' > \phi_0'$$

and if $\lfloor \phi_1'\pi/(80\sigma) \rfloor \geq 1/4$,

$$\frac{\phi_1'}{3/4 + \lfloor \phi_1'\pi/(80\sigma) \rfloor} > \frac{\phi_1'}{4\lfloor \phi_1'\pi/(80\sigma) \rfloor} > 80\sigma/(4\pi) > 6\sigma > \phi_0'.$$

A similar line of reasoning shows that $\widetilde{\phi}_1/c = \beta/c$. Here $\beta/c = 2\pi k + \widetilde{\phi}_1/c$, so it remains to show

$$\phi_1 < 2\pi\sigma c = \frac{\phi_1'}{3/4 + \lfloor \phi_1'\pi/(80\sigma)\rfloor}.$$

Indeed this is the case, since if $\lfloor \phi_1'\pi/(80\sigma)\rfloor < 1/4$,

$$\frac{\phi_1'}{3/4 + \lfloor \phi_1'\pi/(80\sigma)\rfloor} > \phi_1' > 98\sigma > 2\sigma > \mu_1 - \mu_1' = \phi_1,$$

and if $\lfloor \phi_1'\pi/(80\sigma)\rfloor \geq 1/4$, then

$$\frac{\phi_1'}{3/4 + \lfloor \phi_1'\pi/(80\sigma)\rfloor} > \frac{\phi_1'}{4\lfloor \phi_1'\pi/(80\sigma)\rfloor} > 80\sigma/(4\pi) > 6\sigma > \phi_1.$$

Setting $t = 1/c\sigma$, the following calculation holds if $\beta \leq \alpha$:

$$\begin{aligned}
e^{-\frac{\sigma^2 t^2}{2}} \left| e^{it\mu_0} + e^{it\mu_1} - e^{it\mu_0'} - e^{it\mu_1'} \right| &= e^{-\frac{\sigma^2 t^2}{2}} \left| 1 + e^{it(\phi_1+\phi_1')} - e^{it\phi_0'} - e^{it\phi_1'} \right| \\
&= e^{-\frac{1}{2c^2}} \left| 1 + e^{i\frac{\tilde{\phi}_1}{c}} e^{i\frac{\tilde{\phi}_1'}{c}} - e^{i\frac{\tilde{\phi}_0'}{c}} - e^{i\frac{\tilde{\phi}_1'}{c}} \right| \\
&\geq e^{-\frac{1}{2c^2}} \left| \mathrm{Im}(1 + e^{i\frac{\tilde{\phi}_1}{c}} e^{i\frac{\tilde{\phi}_1'}{c}} - e^{i\frac{\tilde{\phi}_0'}{c}} - e^{i\frac{\tilde{\phi}_1'}{c}}) \right| \\
&\geq e^{-\frac{1}{c^2}} (-1 + \cos\frac{\tilde{\phi}_1}{c} + \sin\frac{\tilde{\phi}_0'}{c}) \\
&= e^{-\frac{1}{2c^2}} (-1 + \cos\frac{\beta}{c} + \sin\frac{\alpha}{c}) \\
&\geq e^{-\frac{1}{2c^2}} (\alpha/c - (\alpha/c)^3/6 - (\beta/c)^2/2) \\
&\geq e^{-\frac{1}{2c^2}} (\alpha/c - (\alpha/c)^3/6 - (\alpha/c)^2/2) \\
&\geq e^{-\frac{1}{2c^2}} \frac{\alpha}{3c} \geq \frac{\pi^2\delta_2}{240e}
\end{aligned}$$

The fourth to last inequality follows because $\sin x \geq x - \frac{x^3}{6}$ and $\cos x \geq 1 - \frac{x^2}{2}$. The third to last inequality follows because $\beta \leq \alpha$ and $\alpha/c < 1$. In the final step, we re-used the fact that $\frac{25}{\pi^2} \leq c \leq \frac{80}{\pi^2}$.

If $\alpha < \beta$, then we can do a very similar proof by choosing $t = c\sigma$ for

$$c = \frac{\mu_1 - \mu_0'}{3\pi\sigma/2 + 2\pi\sigma\left\lfloor \frac{\mu_1 - \mu_0'}{80\sigma/\pi} \right\rfloor}.$$

From the choice of c and the fact that $\mu_1 - \mu_0' \geq 98\sigma$, $\frac{25}{\pi^2} \leq c \leq \frac{80}{\pi^2}$. Here we denote $\phi_1' = \mu_1' - \mu_1, \phi_0' = \mu_0' - \mu_1$ and $\phi_0 = \mu_0 - \mu_0'$. From the same explanations as in the case when $\beta \leq \alpha$, we see that

$$\begin{aligned}
\frac{|\widetilde{\phi_0'}|}{c} &= \frac{3\pi}{2} \\
\frac{\pi^2\beta}{80} \leq \frac{|\widetilde{\phi_1'}|}{c} &= \frac{\beta}{c} \leq \frac{\pi^2\beta}{25} \leq \frac{\pi^2}{12} \\
\frac{\pi^2\alpha}{80} \leq \frac{|\widetilde{\phi_0}|}{c} &= \frac{\alpha}{c} \leq \frac{\pi^2\alpha}{25} \leq \frac{\pi^2}{12}.
\end{aligned}$$

We obtain the same bound as in the case of $\beta \leq \alpha$ by factoring out $e^{it\mu_1}$ and using a similar calculation:

$$\begin{aligned}
e^{-\frac{\sigma^2 t^2}{2}} \left| e^{it\mu_0} + e^{it\mu_1} - e^{it\mu_0'} - e^{it\mu_1'} \right| &= e^{-\frac{\sigma^2 t^2}{2}} \left| e^{it(\mu_0 - \mu_1)} + 1 - e^{it(\mu_0' - \mu_1)} - e^{it(\mu_1' - \mu_1)} \right| \\
&= e^{-\frac{\sigma^2 t^2}{2}} \left| e^{it(\phi_0 + \phi_0')} + 1 - e^{it\phi_0'} - e^{it\phi_1'} \right| \\
&= e^{-\frac{1}{2c^2}} \left| 1 + e^{i\frac{\tilde{\phi}_0}{c}} e^{i\frac{\tilde{\phi}_0'}{c}} - e^{i\frac{\tilde{\phi}_0'}{c}} - e^{i\frac{\tilde{\phi}_1'}{c}} \right| \\
&\geq e^{-\frac{1}{2c^2}} \left| \mathrm{Im}\left(1 + e^{i\frac{-|\tilde{\phi}_0|}{c}} e^{i\frac{-|\tilde{\phi}_0'|}{c}} - e^{i\frac{-|\tilde{\phi}_0'|}{c}} - e^{i\frac{-|\tilde{\phi}_1'|}{c}} \right) \right| \\
&\geq e^{-\frac{1}{c^2}} \left(-1 + \cos\frac{|\tilde{\phi}_0|}{c} + \sin\frac{|\tilde{\phi}_1'|}{c} \right) \\
&= e^{-\frac{1}{c^2}} \left(-1 + \cos\frac{\alpha}{c} + \sin\frac{\beta}{c} \right),
\end{aligned}$$

and the rest of the proof follows as before, just swapping α and β. □

APPENDIX B

MISSING PROOFS IN CHAPTER 3

B.1 Description of Algorithm 3.2 and Proof of Theorem 3.4

Algorithm 3.2 (Design of queries and denoising): Let m be the total number of queries that we will make. In the first step of the algorithm, for a particular query vector $\mathbf{v} \in \mathbb{R}^n$, our objective is to recover $\langle \mathbf{v}, \boldsymbol{\beta}^1 \rangle$ and $\langle \mathbf{v}, \boldsymbol{\beta}^2 \rangle$ which we will denote as the *denoised query responses* corresponding to the vector $\mathbf{v}$. It is intuitive, that in order to do this, we need to use the same query vector $\mathbf{v}$ repeatedly a number of times and aggregate the noisy query responses to recover the denoised counterparts.

Therefore, at every iteration in Step 1 of Algorithm 3.2, we sample a vector $\mathbf{v}$ uniformly at random from $\{+1, -1\}^n$. Once the vector $\mathbf{v}$ is sampled, we use $\mathbf{v}$ as query vector repeatedly for T times. We will say that the query responses to the same vector as query to be a *batch* of size T. It can be seen that since $\mathbf{v}$ is fixed, the query responses in a batch is sampled from a Gaussian mixture distribution $\mathcal{M}$ with means $\langle \mathbf{v}, \boldsymbol{\beta}^1 \rangle$ and $\langle \mathbf{v}, \boldsymbol{\beta}^2 \rangle$ and variance σ^2, in short,

$$\mathcal{M} = \frac{1}{2}\mathcal{N}(\langle \mathbf{v}, \boldsymbol{\beta}^1 \rangle, \sigma^2) + \frac{1}{2}\mathcal{N}(\langle \mathbf{v}, \boldsymbol{\beta}^2 \rangle, \sigma^2).$$

Therefore the problem reduces to recovering the mean parameters from a mixture of Gaussian distribution with at most two mixture constituents (since the means can be same) and having the same variance. We will use the following important lemma for this problem.

Lemma B.1 (Lemma 3.2: Learning Gaussian mixtures)**.** *Let* $\mathcal{M} = \frac{1}{L}\sum_{i=1}^{L}\mathcal{N}(\mu_i, \sigma^2)$ *be a uniform mixture of* L *univariate Gaussians, with known shared variance* σ^2 *and with means* $\mu_i \in \epsilon\mathbb{Z}$*. Then, for some constant* $c > 0$ *and some* $t = \omega(L)$*, there exists an algorithm that requires* $ctL^2\exp((\sigma/\epsilon)^{2/3})$ *samples from* $\mathcal{M}$ *and exactly identifies the parameters* $\{\mu_i\}_{i=1}^{L}$ *with probability at least* $1 - 2e^{-2t}$*.*

The proof of this lemma can be found in Chapter 2. We now have the following lemma to characterize the size of each batch T.

Lemma B.2. *For any query vector* $\mathbf{v} \in \{+1, 0, -1\}^n$*, a batchsize of*

$$T = c_1 \log n \exp((\sigma/\epsilon)^{2/3})$$

, for a constant $c_1 > 0$*, is sufficient to recover the denoised query responses* $\langle \mathbf{v}, \boldsymbol{\beta}^1\rangle$ *and* $\langle \mathbf{v}, \boldsymbol{\beta}^2\rangle$ *with probability at least* $1 - 1/\text{poly}(n)$*.*

Proof. Since $\mathbf{v} \in \{+1, 0, -1\}^n$, $\langle \mathbf{v}, \boldsymbol{\beta}^1\rangle, \langle \mathbf{v}, \boldsymbol{\beta}^2\rangle \in \epsilon\mathbb{Z}$. Using Lemma 3.2, the claim follows. □

Corollary B.1. *For any* $O\Big(k \log n \log(n/k)\Big)$ *query vectors sampled uniformly at random from* $\{+1, -1\}^n$*, a batch size of* $T > c_2 \log n \exp((\frac{\sigma}{\epsilon})^{2/3})$*, for some constant* $c_2 > 0$*, is sufficient to recover the denoised query responses corresponding to every query vector with probability at least* $1 - 1/\text{poly}(n)$*.*

Proof. This statement is proved by taking a union bound over $O\Big(k \log n \log(n/k)\Big)$ batches corresponding to that many query vectors. □

Algorithm 3.2 (Alignment step): Notice from the previous discussion, for each batch corresponding to a query vector $\mathbf{v}$, we obtain the pair of values $(\langle \mathbf{v}, \boldsymbol{\beta}^1\rangle, \langle \mathbf{v}, \boldsymbol{\beta}^2\rangle)$. However, we still need to cluster these values (by taking one value from each pair and assigning it to one of the clusters) into two clusters corresponding to β_1 and β_2. We

will first explain the clustering process for two particular query vectors $\mathbf{v}^1$ and $\mathbf{v}^2$ for which we have already obtained the pairs $(\langle \mathbf{v}^1, \boldsymbol{\beta}^1 \rangle, \langle \mathbf{v}^1, \boldsymbol{\beta}^2 \rangle)$ and $(\langle \mathbf{v}^2, \boldsymbol{\beta}^1 \rangle, \langle \mathbf{v}^2, \boldsymbol{\beta}^2 \rangle)$. The objective is to cluster the four samples into two groups of two samples each so that the samples in each cluster correspond to the same unknown sensed vector. Now, we have two cases to consider:

Case 1: $(\langle \mathbf{v}^1, \boldsymbol{\beta}^1 \rangle = \langle \mathbf{v}^1, \boldsymbol{\beta}^2 \rangle$ or $\langle \mathbf{v}^2, \boldsymbol{\beta}^1 \rangle = \langle \mathbf{v}^2, \boldsymbol{\beta}^2 \rangle)$ In this scenario, the values in at least one of the pairs are same and any grouping works.

Case 2: $(\langle \mathbf{v}^1, \boldsymbol{\beta}^1 \rangle \neq \langle \mathbf{v}^1 \boldsymbol{\beta}^2 \rangle$ and $\langle \mathbf{v}^2, \boldsymbol{\beta}^1 \rangle \neq \langle \mathbf{v}^2, \boldsymbol{\beta}^2 \rangle)$. We use two more batches corresponding to the vectors $\frac{\mathbf{v}^1+\mathbf{v}^2}{2}$ and $\frac{\mathbf{v}^1-\mathbf{v}^2}{2}$ which belong to $\{-1, 0, +1\}^n$. We will call the vector $\frac{\mathbf{v}^1+\mathbf{v}^2}{2}$ the *sum query* and the vector $\frac{\mathbf{v}^1-\mathbf{v}^2}{2}$ the *difference query* corresponding to $\mathbf{v}^1, \mathbf{v}^2$ respectively. Hence using Lemma B.2 again, we will be able to obtain the pairs $(\langle \frac{\mathbf{v}^1+\mathbf{v}^2}{2}, \boldsymbol{\beta}^1 \rangle, \langle \frac{\mathbf{v}^1+\mathbf{v}^2}{2}, \boldsymbol{\beta}^2 \rangle)$ and $(\langle \frac{\mathbf{v}^1-\mathbf{v}^2}{2}, \boldsymbol{\beta}^1 \rangle, \langle \frac{\mathbf{v}^1-\mathbf{v}^2}{2}, \boldsymbol{\beta}^2 \rangle)$. Now, we will choose two elements from the pairs $(\langle \mathbf{v}^1, \boldsymbol{\beta}^1 \rangle, \langle \mathbf{v}^1 \boldsymbol{\beta}^2 \rangle)$ and $(\langle \mathbf{v}^2, \boldsymbol{\beta}^1 \rangle, \langle \mathbf{v}^2 \boldsymbol{\beta}^2 \rangle)$ (one element from each pair) such that their sum belongs to the pair $2\langle \frac{\mathbf{v}^1+\mathbf{v}^2}{2}, \boldsymbol{\beta}^1 \rangle, 2\langle \frac{\mathbf{v}^1+\mathbf{v}^2}{2}, \boldsymbol{\beta}^2 \rangle$ and their difference belongs to the pair $2\langle \frac{\mathbf{v}^1-\mathbf{v}^2}{2}, \boldsymbol{\beta}^1 \rangle, 2\langle \frac{\mathbf{v}^1-\mathbf{v}^2}{2}, \boldsymbol{\beta}^2 \rangle$. In our algorithm, we will put these two elements into one cluster and the other two elements into the other cluster. From construction, we must put $(\langle \mathbf{v}^1, \boldsymbol{\beta}^1 \rangle, \langle \mathbf{v}^2, \boldsymbol{\beta}^1 \rangle)$ in one cluster and $(\langle \mathbf{v}^1, \boldsymbol{\beta}^2 \rangle, \langle \mathbf{v}^2, \boldsymbol{\beta}^2 \rangle)$ in other.

Putting it all together, in Algorithm 3.2, we uniformly and randomly choose $c_s k \log \frac{n}{k}$ query vectors from $\{+1, -1\}^n$ and for each of them, we use it repeatedly for $c_2 \log n \exp\left(\frac{\sigma}{\epsilon}\right)^{2/3}$ times. From each batch, we recover the denoised query responses for the query vector associated with that batch. For a particular query vector $\mathbf{v}$, we call the query vector *good* if $\langle \mathbf{v}, \boldsymbol{\beta}^1 \rangle \neq \langle \mathbf{v}, \boldsymbol{\beta}^2 \rangle$. For a $\mathbf{v}$ chosen uniformly at randomly from $\{+1, -1\}^n$, the probability that $\langle \mathbf{v}, \boldsymbol{\beta}^1 - \boldsymbol{\beta}^2 \rangle = 0$ is at most $\frac{1}{2}$. Therefore, if one chooses $\log n$ query vectors uniformly and independently at random from $\{+1, -1\}^n$, at least one is good with probability $1 - \frac{1}{n}$. We are now ready to prove the main theorem.

Proof of Theorem 3.4. For each vector $\mathbf{v}$ belonging to the set of first $\log n$ query vectors and for each query vector $\mathbf{b}$ ($\mathbf{b}$ is among the initial $c_s k \log \frac{n}{k}$ query vectors) different from $\mathbf{v}$, we make two additional batches of queries corresponding to query vectors $\frac{\mathbf{v}+\mathbf{b}}{2}$ and $\frac{\mathbf{v}-\mathbf{b}}{2}$. Consider the first $\log n$ query vectors. We know that one of them, say $\mathbf{g}$, is a good query vector. Let us denote the denoised means obtained from the batch of queries corresponding to $\mathbf{g}$ to be (x, y). We can think of x and y as labels for the clustering of the denoised means from the other query vectors. Now, from the alignment step, we know that for every query vector $\mathbf{b}$ different from $\mathbf{g}$ and the denoised query responses (p, q) corresponding to $\mathbf{b}$, by using the additional sum and difference queries, we can label one of the element in (p, q) as x and the other one as y. Since the vector $\mathbf{g}$ is good, therefore $x \neq y$ and hence we will be able to aggregate the denoised query responses corresponding to $\boldsymbol{\beta}^1$ and the denoised query responses corresponding to $\boldsymbol{\beta}^2$ separately. Since we have $c_s k \log n$ query responses for each of $\boldsymbol{\beta}^1$ and $\boldsymbol{\beta}^2$, we can scale the query responses by a factor of $1/\sqrt{c_s k \log n}$ and subsequently, we can run basis pursuit [29] to recover the best k-sparse approximations of both $\boldsymbol{\beta}^1$ and $\boldsymbol{\beta}^2$. Notice that the total number of queries in this scheme is $O(k \log^2 n)$ and since the size of each batch corresponding to each query is $O(\log n \exp((\frac{\sigma}{\epsilon})^{2/3}))$, the total sample complexity required is $O\Big(k(\log n)^3 \exp\left(\frac{\sigma}{\epsilon}\right)^{2/3}\Big)$. □

B.2 Analysis of Algorithm 3.3 for General L and Proof of Theorem 3.3 and Theorem 3.5

Algorithm 3.3 (Design of queries): In every iteration in Step 1 of Algorithm 3.3, we will sample a vector $\mathbf{v}$ uniformly at random from $\{+1, -1\}^n$, another vector $\mathbf{r}$ uniformly at random from $\mathcal{G} \equiv \{-2z^\star, -2z^\star + 1, \ldots, 2z^\star - 1, 2z^\star\}^n$ and a number q uniformly at random from $\{1, 2, \ldots, 4z^\star + 1\}$. Now, we will use a batch of queries corresponding to the vectors $\mathbf{v} + \mathbf{r}$, $(q-1)\mathbf{r}$ and $\mathbf{v} + q\mathbf{r}$. We have the following lemmas describing several necessary properties of such queries.

We will define a triplet of query vectors $(\mathbf{v}^1, \mathbf{v}^2, \mathbf{v}^3)$ to be *good* if for all triplets of indices $i, j, k \in [L]$ such that i, j, k are not identical, it must happen that

$$\langle \mathbf{v}^1, \boldsymbol{\beta}^i \rangle + \langle \mathbf{v}^2, \boldsymbol{\beta}^j \rangle \neq \langle \mathbf{v}^3, \boldsymbol{\beta}^k \rangle$$

Lemma B.3. *The query vector triplet* $(\mathbf{v} + \mathbf{r}, (q-1)\mathbf{r}, \mathbf{v} + q\mathbf{r})$ *is good with probability at least* $\frac{1}{\sqrt{\alpha^\star}}$.

Proof. Notice that for a fixed triplet $i, j, k \in [L]$ such that i, j, k are not identical, we must have

$$\begin{aligned}
&\Pr(\langle \mathbf{v} + \mathbf{r}, \boldsymbol{\beta}^i \rangle + \langle (q-1)\mathbf{r}, \boldsymbol{\beta}^j \rangle = \langle \mathbf{v} + q\mathbf{r}, \boldsymbol{\beta}^k \rangle) \\
&= \Pr(\langle \mathbf{r}, \boldsymbol{\beta}^i + (q-1)\boldsymbol{\beta}^j - q\boldsymbol{\beta}^k \rangle = \langle \mathbf{v}, \boldsymbol{\beta}^k - \boldsymbol{\beta}^i \rangle) \\
&\leq \Pr(\boldsymbol{\beta}^i + (q-1)\boldsymbol{\beta}^j - q\boldsymbol{\beta}^k = 0) + \Pr(\boldsymbol{\beta}^i + (q-1)\boldsymbol{\beta}^j - q\boldsymbol{\beta}^k \neq 0) \\
&\cdot \Pr(\langle \mathbf{r}, \boldsymbol{\beta}^i + (q-1)\boldsymbol{\beta}^j - q\boldsymbol{\beta}^k \rangle = \langle \mathbf{v}, \boldsymbol{\beta}^k - \boldsymbol{\beta}^i \rangle \mid \boldsymbol{\beta}^i + (q-1)\boldsymbol{\beta}^j - q\boldsymbol{\beta}^k \neq 0) \\
&\leq \Big(1 - \frac{1}{4z^\star + 1}\Big)\frac{1}{4z^\star + 1} + \frac{1}{4z^\star + 1} = \frac{2}{4z^\star + 1} - \frac{1}{(4z^\star + 1)^2}.
\end{aligned}$$

Notice that $\boldsymbol{\beta}^i + (q-1)\boldsymbol{\beta}^j - q\boldsymbol{\beta}^k = 0$ cannot hold for two values of $q : q_1$ and q_2. We will show this fact by contradiction. Suppose it happens that $\boldsymbol{\beta}^i + (q_1 - 1)\boldsymbol{\beta}^j - q_1\boldsymbol{\beta}^k = 0$ and $\boldsymbol{\beta}^i + (q_2 - 1)\boldsymbol{\beta}^j - q_2\boldsymbol{\beta}^k = 0$ in which case we must have $\boldsymbol{\beta}^j = \boldsymbol{\beta}^k$ which is a contradiction to the fact that all the unknown vectors are distinct. We can take a union over all possible triplets (at most L^3 of them) and therefore we must have that

$$\begin{aligned}
&\Pr(\text{The vector triplet } (\mathbf{v} + \mathbf{r}, (q-1)\mathbf{r}, \mathbf{v} + q\mathbf{r}) \text{ is good}) \\
&\geq 1 - L^3\Big(\frac{2}{4z^\star + 1} - \frac{1}{(4z^\star + 1)^2}\Big) \\
&\geq \frac{1}{\sqrt{\alpha^\star}}.
\end{aligned}$$

□

We will now generalize Lemma B.2 in order to characterize the batch size required to recover the denoised query responses when there are L unknown vectors that the oracle can sample from.

Lemma B.4 (Generalization of Lemma B.2)**.** *For a particular query vector* $\mathbf{v}$ *such that each entry of* $\mathbf{v}$ *is integral, a batch size of* $T > c_3 \log n \exp((\sigma/\epsilon)^{2/3})$*, for some constant* $c_3 > 0$*, is sufficient to recover the denoised query responses* $\langle \mathbf{v}, \boldsymbol{\beta}^1 \rangle, \langle \mathbf{v}, \boldsymbol{\beta}^2 \rangle, \ldots, \langle \mathbf{v}, \boldsymbol{\beta}^L \rangle$ *with probability at least* $1 - 1/\text{poly}(n)$*.*

Proof. The proof follows in exactly the same manner as the proof in Lemma B.2 but in this case, we invoke Lemma 3.2 with any general value of L. Since we have assumed that L is a constant, the term L^2 is subsumed within the constant c_3. □

Corollary B.2. *For* $O\Big(k \log^2 n\Big)$ *query vectors such that every entry of every query vector is integral, a batch size of* $T > c_4 \log n \exp((\sigma/\epsilon)^{2/3})$*, for some constant* $c_4 > 0$*, is sufficient to recover the denoised query responses corresponding to every query vector with probability at least* $1 - 1/\text{poly}(n)$*.*

Proof. Again, we can take a union bound over all $O\Big(k \log^2 n\Big)$ query vectors to obtain the result. □

Lemma B.5. *If we draw* $\sqrt{\alpha^\star} \log n$ *triplets of query vectors* $(\mathbf{v} + \mathbf{r}, (q-1)\mathbf{r}, \mathbf{v} + q\mathbf{r})$ *randomly as described, then at least one of the triplets is good with probability at least* $1 - 1/n$*.*

Proof. Now, the probability of a triplet of vectors $(\mathbf{v} + \mathbf{r}, (q-1)\mathbf{r}, \mathbf{r})$ being not good is less than $1 - \frac{1}{\sqrt{\alpha^\star}}$ and therefore the probability of all the $\sqrt{\alpha^\star} \log n$ triplets being not good is less than

$$\Big(1 - \frac{1}{\sqrt{\alpha^\star}}\Big)^{\log n \sqrt{\alpha^\star}} \leq e^{-\log n} \leq n^{-1}$$

which proves the statement of the lemma. □

Lemma B.6. *For a good triplet of vectors* $(\mathbf{v}+\mathbf{r},(q-1)\mathbf{r},\mathbf{v}+q\mathbf{r})$*, we can obtain* $\langle \mathbf{v},\boldsymbol{\beta}^i\rangle$ *for all* $i\in[L]$*.*

Proof. Recall that since we queried the vector $\mathbf{v}+q\mathbf{r}$, we can simply check which element (say x) from the set $\{\langle \mathbf{v}+\mathbf{r},\boldsymbol{\beta}^i\rangle\}_{i=1}^L$ and which element (say y) from the set $\{\langle (q-1)\mathbf{r},\boldsymbol{\beta}^i\rangle\}_{i=1}^L$ adds up to an element in $\{\langle \mathbf{v}+q\mathbf{r},\boldsymbol{\beta}^i\rangle\}_{i=1}^L$. It must happen that the elements x and y must correspond to the same unknown vector $\boldsymbol{\beta}^i$ for some $i\in[L]$ because the triplet of vectors $(\mathbf{v}+\mathbf{r},(q-1)\mathbf{r},q\mathbf{r})$ is good. Hence computing $x-(y/(q-1))$ allows us to obtain $\langle \mathbf{v},\boldsymbol{\beta}^i\rangle$ and this step can be done for all $i\in[L]$. □

Algorithm 3.3 (Alignment step): Let a particular good query vector triplet be $(\mathbf{v}^\star+\mathbf{r}^\star,(q^\star-1)\mathbf{r}^\star,\mathbf{v}^\star+q^\star\mathbf{r}^\star)$. From now, we will consider the L elements $\{\langle \mathbf{r}^\star,\boldsymbol{\beta}^i\rangle\}_{i=1}^L$ (necessarily distinct) to be labels and for a vector $\mathbf{u}$, we will associate a label with every element in $\{\langle \mathbf{u},\beta^i\rangle\}_{i=1}^L$. The labelling is correct if, for all $i\in[L]$, the element labelled as $\langle \mathbf{r}^\star,\boldsymbol{\beta}^i\rangle$ also corresponds to the same unknown vector $\boldsymbol{\beta}^i$. Notice that we can label the elements $\{\langle \mathbf{v}^\star,\beta^i\rangle\}_{i=1}^L$ correctly because the triplet $(\mathbf{v}^\star+\mathbf{r}^\star,(q^\star-1)r^\star,\mathbf{v}^\star+q^\star\mathbf{r}^\star)$ is good and by applying the reasoning in Lemma B.6. Consider another good query vector triplet $(\mathbf{v}'+\mathbf{r}',(q'-1)\mathbf{r}',\mathbf{v}'+q'\mathbf{r}')$ which we will call *matching good* with respect to $(\mathbf{v}^\star+\mathbf{r}^\star,(q^\star-1)r^\star,\mathbf{v}^\star+q^\star\mathbf{r}^\star)$ if it is good and additionally, the vector triplet $(\mathbf{r}',\mathbf{r}^\star,\mathbf{r}'+\mathbf{r}^\star)$ is also good.

Lemma B.7. *For a fixed known good query vector triplet* $(\mathbf{v}^\star+\mathbf{r}^\star,(q^\star-1)r^\star,\mathbf{v}^\star+q^\star\mathbf{r}^\star)$*, the probability that any randomly drawn query vector triplet* $(\mathbf{v}'+\mathbf{r}',(q-1)\mathbf{r}',\mathbf{v}'+q'\mathbf{r}')$ *is matching good with respect to* $(\mathbf{v}^\star+\mathbf{r}^\star,(q^\star-1)r^\star,\mathbf{v}^\star+q^\star\mathbf{r}^\star)$ *is at least* $\frac{1}{\sqrt{\alpha^\star}}$*.*

Proof. From Lemma B.3, we know that the probability that a randomly drawn query vector triplet $(\mathbf{v}'+\mathbf{r}',(q-1)\mathbf{r}',\mathbf{v}'+q'\mathbf{r}')$ is not good is at most $L^3\left(\frac{2}{4z^\star+1}-\frac{1}{(4z^\star+1)^2}\right)$. Again, we must have for a fixed triplet of indices $i,j,k\in[L]$ such that they are not identical

$$\Pr(\langle \mathbf{r}',\boldsymbol{\beta}^i\rangle+\langle \mathbf{r}^\star,\boldsymbol{\beta}^j\rangle=\langle \mathbf{r}'+\mathbf{r}^\star,\boldsymbol{\beta}^k\rangle)$$

$$= \Pr(\langle \mathbf{r}', \boldsymbol{\beta}^i - \boldsymbol{\beta}^k \rangle = \langle \mathbf{r}^\star, \boldsymbol{\beta}^k - \boldsymbol{\beta}^j \rangle) \leq \frac{1}{4z^\star + 1}$$

Taking a union bound over all non-identical triplets (at most L^3 of them), we get that

$$\Pr((\mathbf{r}', \mathbf{r}^\star, \mathbf{r}' + \mathbf{r}^\star \text{ is not good}) \leq \frac{L^3}{4z^\star + 1}$$

Taking a union bound over both the failure events, we get that

$$\begin{aligned}
&\Pr((\mathbf{v}' + \mathbf{r}', (q-1)\mathbf{r}', \mathbf{v}' + q'\mathbf{r}') \text{ is not matching good}) \\
&\leq L^3 \Big(\frac{3}{4z^\star + 1} - \frac{1}{(4z^\star + 1)^2} \Big) \\
&\leq 1 - \frac{1}{\sqrt{\alpha^\star}}
\end{aligned}$$

which proves the lemma. □

Lemma B.8. *For a matching good query vector triplet* $(\mathbf{v}' + \mathbf{r}', (q-1)\mathbf{r}', \mathbf{v}' + q\mathbf{r}')$*, we can label the elements in* $\{\langle \mathbf{v}', \beta^i \rangle\}_{i=1}^L$ *correctly by querying the vector* $\mathbf{r}' + \mathbf{r}^\star$*.*

Proof. Since $(\mathbf{v}' + \mathbf{r}', (q-1)\mathbf{r}', \mathbf{v}' + q\mathbf{r}')$ is good and we have also queried $\mathbf{v}' + q\mathbf{r}'$, we can partition the set of elements $\{\langle \mathbf{v}' + \mathbf{r}', \beta^i \rangle\}_{i=1}^L \cup \{\langle (q-1)\mathbf{r}', \beta^i \rangle\}_{i=1}^L$ into groups of two elements each such that the elements in each group correspond to the same unknown vector $\boldsymbol{\beta}^i$ as in the reasoning presented in proof of Lemma B.6. Again, since $(\mathbf{r}', \mathbf{r}^\star, \mathbf{r}' + \mathbf{r}^\star)$ is good and we have queried $\mathbf{r}' + \mathbf{r}^\star$, we can create a similar partition of the set of elements $\{\langle \mathbf{r}', \beta^i \rangle\}_{i=1}^L \cup \{\langle \mathbf{r}^\star, \beta^i \rangle\}_{i=1}^L$ and multiply every element by a factor of $q - 1$. For each of the two partitions described above we can align two groups together (one from each partition) if both groups contain $\langle (q-1)\mathbf{r}', \beta^i \rangle$ for the same $i \in L$ (the values $\langle \mathbf{r}', \beta^i \rangle$ are necessarily distinct and therefore this is possible). Hence, for every $i \in [L]$, we can compute $\langle \mathbf{v}', \beta^i \rangle$ correctly and also label it correctly because of the alignment. □

Algorithm 3.3 (Putting it all together) First, we condition on the event that for all batches of queries (number of batches will be polynomial in k and $\log n$) we make, the denoised means are extracted correctly which happens with probability at least $1 - \frac{1}{n}$ by Corollary B.2. As described in Algorithm 3.3, in the first step we sample a pair of vectors $(\mathbf{v}, \mathbf{r})$ such that $\mathbf{v}$ is uniformly drawn from $\{-1, +1\}^n$ and $\mathbf{r}$ is uniformly drawn from $\{-2z^\star, -2z^\star + 1, \ldots, 2z^\star - 1, 2z^\star\}^n$. We also sample a random number q uniformly and independently from the set $\{1, 2, \ldots, 4z^\star + 1\}$ and subsequently, we use batches of queries of size $c_4 L^2 \log n \exp((\sigma/\epsilon)^{2/3})$ corresponding to the three vectors $\mathbf{v} + \mathbf{r}, (q-1)\mathbf{r}$ and $\mathbf{v} + q\mathbf{r}$ respectively. We will repeat this step for $\sqrt{\alpha^\star} \log n + c'\alpha^\star k \log(n/k)$ iterations. Additionally, for each query vector pair $((\mathbf{v_1}, \mathbf{r_1})$ among the first $\sqrt{\alpha^\star} \log n$ iterations and for each vector pair $((\mathbf{v_2}, \mathbf{r_2})$ among the latter $c'\alpha^\star k \log(n/k)$ iterations, we also make the batch of queries corresponding to the vector $\mathbf{r_1} + \mathbf{r_2}$. From Lemma B.5, we know that with probability at least $1 - \frac{1}{n}$, one of the query vector triplets among the first $\sqrt{\alpha^\star} \log n$ triplets is good. Moreover, it is also easy to check if a query vector triplet is good or not and therefore it is easy to identify one. Once a good query vector triplet $(\mathbf{v}^\star + \mathbf{r}^\star, (q^\star - 1)\mathbf{r}^\star, \mathbf{v}^\star + q^\star\mathbf{r}^\star)$ is identified, it is also possible to correctly identify matching good query vectors among the latter $c'\alpha^\star k \log(n/k)$ query vector triplets with respect to the good vector triplet. We now have the following lemma characterizing the number of matching good query vector triplets:

Lemma B.9. *The number of matching good query vector triplets from $\alpha^\star c' k \log(n/k)$ randomly chosen triplets is at least $c' k \log(n/k)$ with probability at least $1 - \left(\frac{k}{n}\right)^{\tilde{c}k}$ for some constant $\tilde{c} > 0$.*

Proof. For a randomly drawn query vector triplet, we know that it is matching good with probability at least $\frac{1}{\sqrt{\alpha^\star}}$ from Lemma B.7. Since there are $\alpha^\star c' k \log(n/k)$ query vector triplets drawn at random independently, the expected number of matching-good

triplets is at least $\sqrt{\alpha^\star}c'k\log(n/k)$. Further, by using Chernoff bound [26], we can show that

$$\begin{aligned}
&\Pr(\text{Number of matching good triplets} < c'k\log(n/k)) \\
&= \Pr(\text{Number of matching good triplets} < \sqrt{\alpha^\star}c'k\log(n/k)\left(1-\frac{\sqrt{\alpha^\star}-1}{\sqrt{\alpha^\star}}\right) \\
&\leq \exp\left(-\frac{(\sqrt{\alpha^\star}-1)^2c'k\log(n/k)}{2\sqrt{\alpha^\star}}\right).
\end{aligned}$$

□

From Lemma B.8, we know that for every matching good query vector triplet $(\mathbf{v}'+\mathbf{r}', (q-1)\mathbf{r}', \mathbf{v}'+q\mathbf{r}')$, we can label the elements in $\{\langle \mathbf{v}', \beta^i\rangle\}_{i=1}^L$ correctly and from Lemma B.9, we know that we have aggregated de-noised query measurements corresponding to $c'k\log(n/k)$ vectors randomly sampled from $\{+1,-1\}^n$. However, since we have specifically picked $c'k\log(n/k)$ matching good vectors after the entire scheme, we do not know which query vectors will be matching good apriori and therefore we need to have the following guarantee:

Lemma B.10. *From $\alpha^\star c'k\log(n/k)$ vectors randomly chosen from $\{+1,-1\}^n$, any $c'k\log(n/k)$ vectors scaled by a factor of $1/\sqrt{c'k\log(n/k)}$ will satisfy the $\delta-$ RIP property with high probability.*

The proof of this lemma is delegated to Section B.3. Now we are ready to proof the main theorem in this setting.

Proof of Theorem 3.5. The total number of batches of queries made is at most $3c'\alpha^\star k\log(n/k)\log n$. Further, recall that size of each batch that is sufficient to recover the denoised means accurately is $c_4\log n\log(\sigma/\epsilon)^{2/3}$. Hence the total number of queries is $O\Big(k(\log n)^3\exp(\sigma/\epsilon)^{2/3}\Big)$ as mentioned in the theorem statement. From Lemma B.8 and Lemma B.10, we know that for every vector $\{\boldsymbol{\beta}^i\}_{i=1}^L$, we have

$c'k\log(n/k)$ linear query measurements such that the measurement matrix scaled by $1/\sqrt{c'k\log(n/k)}$ has the $\delta - \mathsf{RIP}$ property. Therefore, it is possible to obtain the best k-sparse approximation of all the vectors $\boldsymbol{\beta}^1, \boldsymbol{\beta}^2, \ldots, \boldsymbol{\beta}^L$ by using efficient algorithms such as Basis Pursuit. □

Now Theorem 3.3 follows as a corollary.

Proof of Theorem 3.3 for general L. Notice that the query with the largest magnitude of query response that we will make is $\mathbf{v}+(4z^\star+1)\mathbf{r}$ where $\mathbf{v}$ is sampled from $\{+1,-1\}^n$ and $\mathbf{r}$ is sampled from $\{-2z^\star, -2z^\star+1, \ldots, 2z^\star-1, 2z^\star\}$. Therefore, we must have

$$\begin{aligned}
&\mathbb{E}|\langle \mathbf{v} + (4z^\star+1)\mathbf{r}, \boldsymbol{\beta}^i\rangle|^2 \\
&= \mathbb{E}|\langle \mathbf{v}, \boldsymbol{\beta}^i\rangle|^2 + (4z^\star+1)^2\mathbb{E}|\langle \mathbf{r}, \boldsymbol{\beta}^i\rangle|^2 \\
&= 1 + (4z^\star+1)\sum_{i=1}^{2z^\star} i^2 \\
&= 1 + \frac{z^\star(2z^\star+1)(4z^\star+1)^2}{3}.
\end{aligned}$$

since $||\boldsymbol{\beta}^i||_2 = 1$. Since the variance of the noise $\mathbb{E}\eta^2$ is σ^2, we must have that

$$\mathsf{SNR} = \frac{1}{\sigma^2}\Big(1 + \frac{z^\star(2z^\star+1)(4z^\star+1)^2}{3}\Big).$$

Substituting the above expression in the statement of Theorem 3.5 and using the fact that $z^\star$ is a constant, we get the statement of the corollary. □

B.3 Proof of Lemma B.10

First, let us introduce a few notations. For a given any set of indices $\mathbf{T} \subset [n]$, denote by $\mathbf{X_T}$ the set of all vectors in $\mathbb{R}^n$ that are zero outside of T. We start by stating the Johnson-Linderstrauss Lemma proved in [14].

Lemma B.11. *[Lemma 5.1 in [14]] Let* $\mathbf{A}$ *be a* $m \times n$ *matrix such that every element in* $\mathbf{A}$ *is sampled independently and uniformly at random from* $\{1/\sqrt{m}, -1/\sqrt{m}\}$*. For any set* $T \subset [n]$ *such that* $|T| = k$ *and any* $0 < \delta < 1$*, we have*

$$(1-\delta)||\mathbf{x}||_2 \leq ||\mathbf{A}\mathbf{x}||_2 \leq (1+\delta)||\mathbf{x}||_2 \quad \text{for all } \mathbf{x} \in \mathbf{X_T}$$

with probability at least $1 - 2(12/\delta)^k e^{-\frac{m}{2}(\delta^2/8 - \delta^3/24)}$.

We are now ready to prove Lemma B.10. Since there are $\binom{n}{k}$ distinct subsets of $[n]$ that are of size k, we take a union bound over all the subsets and therefore the failure probability of Lemma B.11 for all sets of indices of size k (definition of δ-RIP) is at most

$$2(12/\delta)^k \binom{n}{k} e^{-\frac{m}{2}(\delta^2/8 - \delta^3/24)}.$$

We need that from $\alpha m (\alpha > 1)$ vectors randomly sampled from $\{\frac{1}{\sqrt{m}}, \frac{-1}{\sqrt{m}}\}^n$ any m vectors satisfy the δ-RIP property for some value of m. Therefore, the probability of failure is at most

$$2\left(\frac{12}{\delta}\right)^k \binom{n}{k} \binom{\alpha m}{m} e^{-\frac{m}{2}(\delta^2/8 - \delta^3/24)}.$$

By Stirling's approximation and the fact that both αm and m is large, we get that

$$\binom{\alpha m}{m} \approx \sqrt{\frac{\alpha}{2\pi m(\alpha-1)}} \Big(\frac{\alpha^\alpha}{(\alpha-1)^{\alpha-1}}\Big)^m$$

Further we can also upper bound the binomial coefficients $\binom{n}{k}$ by $\left(\frac{en}{k}\right)^k$. Hence we can upper bound the failure probability as

$$\exp\Big(- m(\delta^2/16 - \delta^3/48) + m \log\Big(\frac{\alpha^\alpha}{(\alpha-1)^{\alpha-1}}\Big) + k\log(en/k) + \log(12/\delta) + \log 2\Big)$$

Therefore, if we substitute $m = c'k\log(en/k)$ for some constant $c' > 0$, we must have the failure probability to be upper bounded as $e^{-c''m(1+o(1))}$ for some $c'' > 0$ as long as we have

$$c'(\frac{\delta^2}{16} - \frac{\delta^3}{48}) > c' \log\Big(\frac{\alpha^\alpha}{(\alpha-1)^{\alpha-1}}\Big) + 1$$

implying that

$$\frac{\alpha^\alpha}{(\alpha-1)^{\alpha-1}} < \exp\Big(\frac{\delta^2}{16} - \frac{\delta^3}{48} - \frac{1}{c'}\Big).$$

Hence, by choosing the constant c' appropriately large, the term in the exponent on the right hand side can be made positive. Since the left hand side of the equation is always greater than 1, there will exist an α satisfying the equation.

B.4 Proofs of Lemma 3.3 and 3.4

Let X be a random variable which is distributed according to $\mathcal{M}$ and suppose we obtain T samples $y_1, y_2, \ldots, y_T \sim \mathcal{M}$. We will divide these T samples into $B := \Big\lceil T/t \Big\rceil$ batches each of size t. In that case let us denote $S^j_{1,t}$ and $S^j_{2,t}$ to be the sample mean and the sample variance of the j^{th} batch i.e.

$$S^j_{1,t} = \sum_{i \in \text{Batch } j} \frac{y_i}{t} \qquad \text{and} \qquad S^j_{2,t} = \frac{1}{t-1} \sum_{i \in \text{Batch } j} (y_i - (S^j_{1,t}))^2.$$

We will estimate the true mean $\mathbb{E}X$ and the true variance $\text{var}X$ by computing $\hat{M}_1$ and $\hat{M}_2$ respectively (See Algorithm 3.5) where

$$\hat{M}_1 \triangleq \mathsf{median}(\{S^j_{1,t}\}_{j=1}^B) \quad \text{and} \quad \hat{M}_2 \triangleq \mathsf{median}(\{S^j_{2,t}\}_{j=1}^B).$$

Proof of Lemma 3.3. For a fixed batch j, we can use Chebychev's inequality to say that

$$\Pr\Big(\big|S^j_{1,t} - \mathbb{E}X\big| \geq \epsilon_1\Big) \leq \frac{\mathrm{var}X}{t\epsilon_1^2}$$

We have

$$\mathrm{var}X = \mathbb{E}X^2 - (\mathbb{E}X)^2 = \frac{1}{2}\Big(2\sigma^2 + \mu_1^2 + \mu_2^2\Big) - \frac{1}{4}(\mu_1 + \mu_2)^2 = \sigma^2 + \frac{(\mu_1 - \mu_2)^2}{4}$$

Noting that we must have $t \geq 1$ as well, we obtain

$$\Pr\Big(\big|S^j_{1,t} - \mathbb{E}X\big| \geq \epsilon_1\Big) \leq \frac{\sigma^2 + (\mu_1 - \mu_2)^2/4}{t\epsilon_1^2} \leq \frac{1}{3}$$

for $t = O(\Big\lceil(\sigma^2 + (\mu_1 - \mu_2)^2)/\epsilon_1^2\Big\rceil)$. Therefore for each batch j, we define an indicator random variable $Z_j = \mathbb{1}[\big|S^j_{1,t} - \mathbb{E}X\big| \geq \epsilon_1]$ and from our previous analysis we know that the probability of Z_j being 1 is less than 1/3. It is clear that $\mathbb{E}\sum_{j=1}^B Z_j \leq B/3$ and on the other hand $|\hat{M}_1 - \mathbb{E}X| \geq \epsilon_1$ iff $\sum_{j=1}^B Z_j \geq B/2$. Therefore, using the Chernoff bound, we have

$$\Pr\Big(\Big|\hat{M}_1 - \mathbb{E}X\Big| \geq \epsilon_1\Big) \leq Pr\Big(\Big|\sum_{j=1}^B Z_j - \mathbb{E}\sum_{j=1}^B Z_j\Big| \geq \frac{\mathbb{E}\sum_{j=1}^B Z_j}{2}\Big) \leq 2e^{-B/36}.$$

Hence, for $B = 36\log\eta^{-1}$, the estimate $\hat{M}_1$ is atmost ϵ_1 away from the true mean $\hat{M}_1$ with probability at least $1 - 2\eta$. Therefore the total sample complexity required is $T = O(\log\eta^{-1}\Big\lceil(\sigma^2 + (\mu_1 - \mu_2)^2)/\epsilon_1^2\Big\rceil)$ proving the lemma. □

Proof of Lemma 3.4. We have

$$\mathbb{E}S^j_{2,t} = \mathbb{E}\frac{1}{t-1}\sum_{i \in \text{Batch } j}(y_i - (S^j_{1,t}))^2$$

$$
\begin{aligned}
&= \mathbb{E}\frac{1}{t(t-1)} \sum_{\substack{i_1,i_2 \in \text{Batch } j \\ i_1<i_2}} (y_{i_1} - y_{i_2})^2 \\
&= \frac{1}{t(t-1)} \sum_{\substack{i_1,i_2 \in \text{Batch } j \\ i_1<i_2}} \mathbb{E}(y_{i_1} - y_{i_2})^2 \\
&= \frac{1}{t(t-1)} \sum_{\substack{i_1,i_2 \in \text{Batch } j \\ i_1<i_2}} \mathbb{E}y_{i_1}^2 + \mathbb{E}y_{i_2}^2 - 2\mathbb{E}\left[y_{i_1}y_{i_2}\right] \\
&= \frac{1}{t(t-1)} \sum_{\substack{i_1,i_2 \in \text{Batch } j \\ i_1<i_2}} 2\sigma^2 + \mu_1^2 + \mu_2^2 - \frac{(\mu_1+\mu_2)^2}{2} \\
&= \sigma^2 + \frac{(\mu_1-\mu_2)^2}{4} = \text{var}X.
\end{aligned}
$$

Hence the estimator $S_{2,t}^j$ is an unbiased estimator since it's expected value is the true variance of X. Again, we must have

$$
\mathbb{E}(S_{2,t}^j)^2 = \mathbb{E}\frac{1}{t^2(t-1)^2}\Big(\sum_{\substack{i_1,i_2 \in \text{Batch } j \\ i_1<i_2}} (y_{i_1} - y_{i_2})^2 \Big)^2.
$$

Claim B.1. *We have*

$$
\mathbb{E}\left[(y_{i_1} - y_{i_2})^2 (y_{i_3} - y_{i_4})^2\right] \leq 48\Big(\sigma^2 + \frac{(\mu_1-\mu_2)^2}{4}\Big)^2
$$

for any i_1, i_2, i_3, i_4 such that $i_1 < i_2$ and $i_3 < i_4$.

Proof. In order to prove this claim consider three cases:

Case 1 (i_1, i_2, i_3, i_4 are distinct): In this case, we have that $y_{i_1} - y_{i_2}$ and $y_{i_3} - y_{i_4}$ are independent and therefore,

$$
\mathbb{E}\left[(y_{i_1} - y_{i_2})^2 (y_{i_3} - y_{i_4})^2\right] = \mathbb{E}\left[(y_{i_1} - y_{i_2})^2\right] \mathbb{E}\left[(y_{i_3} - y_{i_4})^2\right] \leq 4\Big(\sigma^2 + \frac{(\mu_1-\mu_2)^2}{4}\Big)^2.
$$

Case 2 ($i_1 = i_3, i_2 = i_4$): In this case, we have

$$\mathbb{E}\left[(y_{i_1} - y_{i_2})^2 (y_{i_3} - y_{i_4})^2\right] = \mathbb{E}\left[(y_{i_1} - y_{i_2})^4\right].$$

Notice that

$$y_{i_1} - y_{i_2} \sim \frac{1}{2}\mathcal{N}(0, 2\sigma^2) + \frac{1}{4}\mathcal{N}(\mu_1 - \mu_2, 2\sigma^2) + \frac{1}{4}\mathcal{N}(\mu_2 - \mu_1, 2\sigma^2)$$

and therefore we get

$$\mathbb{E}\left[(y_{i_1} - y_{i_2})^4\right] = 48\sigma^4 + 12\sigma^2(\mu_1 - \mu_2)^2 + \frac{(\mu_1 - \mu_2)^4}{2} \leq 48\Big(\sigma^2 + \frac{(\mu_1 - \mu_2)^2}{4}\Big)^2.$$

Case 3 ($\{i_1, i_2, i_3, i_4\}$ has 3 unique elements): Without loss of generality let us assume that $i_1 = i_3$. In that case we have

$$\mathbb{E}\left[(y_{i_1} - y_{i_2})^2 (y_{i_1} - y_{i_4})^2\right] = \mathbb{E}_{y_{i_1}} \mathbb{E}\left[(y_{i_2} - y_{i_1})^2 (y_{i_4} - y_{i_1})^2 \mid y_{i_1}\right]$$

Notice that for a fixed value of y_{i_1}, we must have $y_{i_2} - y_{i_1}, y_{i_4} - y_{i_1}$ to be independent and identically distributed i.e.

$$y_{i_2} - y_{i_1}, y_{i_4} - y_{i_1} \sim \frac{1}{2}\mathcal{N}(\mu_1 - y_{i_1}, \sigma^2) + \frac{1}{2}\mathcal{N}(\mu_2 - y_{i_1}, \sigma^2).$$

Therefore,

$$\begin{aligned}\mathbb{E}\left[(y_{i_2} - y_{i_1})^2 (y_{i_4} - y_{i_1})^2 \mid y_{i_1}\right] &= \mathbb{E}\left[(y_{i_2} - y_{i_1})^2 \mid y_{i_1}\right] \mathbb{E}\left[(y_{i_2} - y_{i_1})^2 \mid y_{i_1}\right] \\ &= \frac{1}{4}\Big(2\sigma^2 + (\mu_1 - y_{i_1})^2 + (\mu_2 - y_{i_1})^2\Big)^2.\end{aligned}$$

Again, we have

$$y_{i_1} - \mu_1 \sim \frac{1}{2}\mathcal{N}(0, \sigma^2) + \frac{1}{2}\mathcal{N}(\mu_2 - \mu_1, \sigma^2)$$

$$y_{i_1} - \mu_2 \sim \frac{1}{2}\mathcal{N}(0, \sigma^2) + \frac{1}{2}\mathcal{N}(\mu_1 - \mu_2, \sigma^2).$$

Hence,

$$\mathbb{E}\Big(2\sigma^2 + (\mu_1 - y_{i_1})^2 + (\mu_2 - y_{i_1})^2\Big)^2 = 4\sigma^4 + \mathbb{E}(\mu_1 - y_{i_1})^4 + \mathbb{E}(\mu_1 - y_{i_2})^4$$
$$+ 4\sigma^2(\mathbb{E}(\mu_1 - y_{i_1})^2 + \mathbb{E}(\mu_1 - y_{i_2})^2) + \mathbb{E}((\mu_1 - y_{i_1})^2(\mu_2 - y_{i_1})^2).$$

We have

$$\mathbb{E}(\mu_1 - y_{i_1})^4 = \mathbb{E}(\mu_2 - y_{i_1})^4 = 3\sigma^4 + \frac{(\mu_1 - \mu_2)^4}{2} + 3\sigma^2(\mu_1 - \mu_2)^2$$
$$\mathbb{E}(\mu_1 - y_{i_1})^2 = \mathbb{E}(\mu_2 - y_{i_1})^2 = \sigma^2 + \frac{(\mu_1 - \mu_2)^2}{2}$$
$$\mathbb{E}((\mu_1 - y_{i_1})^2(\mu_2 - y_{i_1})^2) = \mathbb{E}((\mu_1 - y_{i_1})^2(\mu_2 - \mu_1 + \mu_1 - y_{i_1})^2)$$
$$= \mathbb{E}[(\mu_1 - y_{i_1})^4 + (\mu_1 - \mu_2)^2(\mu_1 - y_{i_1})^2 + 2(\mu_2 - \mu_1)(\mu_1 - y_{i_1})^3]$$
$$= 3\sigma^4 + \frac{(\mu_1 - \mu_2)^4}{2} + 5\sigma^2(\mu_1 - \mu_2)^2.$$

Plugging in, we get

$$\mathbb{E}\Big(2\sigma^2 + (\mu_1 - y_{i_1})^2 + (\mu_2 - y_{i_1})^2\Big)^2 = 17\sigma^4 + \frac{3(\mu_1 - \mu_2)^4}{2} + 13\sigma^2(\mu_1 - \mu_2)^2.$$

Hence, we obtain

$$\mathbb{E}\left[(y_{i_1} - y_{i_2})^2(y_{i_1} - y_{i_4})^2\right] \leq 7\Big(\sigma^2 + \frac{(\mu_1 - \mu_2)^2}{2}\Big)^2.$$

which proves the claim. □

From Claim B.1, we can conclude that

$$\mathbb{E}(S_{2,t}^j)^2 \leq 12\Big(\sigma^2 + \frac{(\mu_1 - \mu_2)^2}{4}\Big)^2.$$

From this point onwards, the analysis in this lemma is very similar to Lemma 3.3. We can use Chebychev's inequality to say that

$$\Pr\Big(\big|S_{2,t}^j - \mathrm{var}X\big| \geq \epsilon_2\Big) \leq \frac{\mathrm{var}S_{2,t}^j}{t\epsilon_2^2} \leq \frac{\mathbb{E}(S_{2,t}^j)^2}{t\epsilon_2^2}.$$

Therefore, we obtain by noting that $t \geq 1$ as well,

$$\Pr\Big(\big|S_{2,t}^j - \mathrm{var}X\big| \geq \epsilon_2\Big) \leq \frac{12(\sigma^2 + (\mu_1 - \mu_2)^2/4)^2}{t\epsilon_2^2} \leq \frac{1}{3}$$

for $t = O(\Big\lceil (\sigma^2 + (\mu_1 - \mu_2)^2)^2/\epsilon_2^2 \Big\rceil)$. At this point, doing the same analysis as in Lemma 3.3 shows that $B = 36 \log \eta^{-1}$ batches of batchsize t is sufficient to estimate the variance within an additive error of ϵ_2 with probability at least $1 - 2\eta$. Therefore the total sample complexity required is $T = O(\log \eta^{-1} \Big\lceil (\sigma^2 + (\mu_1 - \mu_2)^2)^2/\epsilon_2^2 \Big\rceil)$ thus proving the lemma.

□

B.5 Proof of Lemma 3.6

We use $O\Big(\frac{\log \eta^{-1}}{\epsilon^2}\Big)$ samples to recover $\hat{\mu_1}$ and $\hat{\mu_2}$ using the method of moments. According to the guarantee provided in Theorem 3.8, we must have with probability at least $1 - 1/\eta$,

$$|\hat{\mu}_i - \mu| \leq 2(\epsilon + \sqrt{\epsilon})\sqrt{\sigma^2 + (\mu_1 - \mu_2)^2} \quad \text{for } i = 1, 2.$$

Therefore, we have

$$|\mu_1 - \mu_2| - |\mu_1 - \hat{\mu}_1| - |\mu_2 - \hat{\mu}_2| \leq |\hat{\mu}_1 - \hat{\mu_2}| \leq |\mu_1 - \mu_2| + |\mu_1 - \hat{\mu}_1| + |\mu_2 - \hat{\mu}_2|$$

$$||\hat{\mu}_1 - \hat{\mu_2}| - |\mu_1 - \mu_2|| \leq 4(\epsilon + \sqrt{\epsilon})\sqrt{\sigma^2 + (\mu_1 - \mu_2)^2}.$$

Case 1 ($\sigma \geq 1$): We will substitute $\epsilon = 1/256\sigma$. In that case, we must have

$$||\hat{\mu_1} - \hat{\mu_2}| - |\mu_1 - \mu_2|| \leq \frac{1}{2}\sqrt{\sigma + \frac{(\mu_1 - \mu_2)^2}{\sigma}} \leq \frac{\sqrt{\sigma}}{2} + \frac{|\mu_1 - \mu_2|}{2}.$$

and therefore,

$$-\frac{\sqrt{\sigma}}{2} + \frac{|\mu_1 - \mu_2|}{2} \leq |\hat{\mu_1} - \hat{\mu_2}| \leq \frac{\sqrt{\sigma}}{2} + \frac{3\,|\mu_1 - \mu_2|}{2}.$$

Therefore, if $|\mu_1 - \mu_2| = \Omega(\sigma)$, we will have $|\hat{\mu_1} - \hat{\mu_2}| = \Omega(\sigma)$ and on the other hand if, $|\mu_1 - \mu_2| = O(\sigma)$, we will have $|\hat{\mu_1} - \hat{\mu_2}| = O(\sigma)$ as well. The sample complexity required for the test is going to be $O(\sigma^2 \log \eta^{-1})$.

Case 2 ($\gamma \leq \sigma \leq 1$): We will substitute $\epsilon = \sigma/256$. Now, we must have

$$||\hat{\mu_1} - \hat{\mu_2}| - |\mu_1 - \mu_2|| \leq \frac{\sqrt{\sigma}}{2}\sqrt{\sigma^2 + (\mu_1 - \mu_2)^2} \leq o(\sigma) + \frac{\sqrt{\sigma}\,|\mu_1 - \mu_2|}{2}.$$

and therefore,

$$-o(\sigma) + (1 - \frac{\sqrt{\sigma}}{2})\,|\mu_1 - \mu_2| \leq |\hat{\mu_1} - \hat{\mu_2}| \leq o(\sigma) + (1 + \frac{\sqrt{\sigma}}{2})\,|\mu_1 - \mu_2|\,.$$

Therefore, if $|\mu_1 - \mu_2| = \Omega(\sigma)$, we will have $|\hat{\mu_1} - \hat{\mu_2}| = \Omega(\sigma)$ and on the other hand if, $|\mu_1 - \mu_2| = O(\sigma)$, we will have $|\hat{\mu_1} - \hat{\mu_2}| = O(\sigma)$ as well. The sample complexity required for the test is going to be $O(\log \eta^{-1}/\gamma^2)$.

Case 3 ($\sigma \leq \gamma$): We will substitute $\epsilon = 1/256$. In that case we have

$$||\hat{\mu_1} - \hat{\mu_2}| - |\mu_1 - \mu_2|| \leq \frac{17}{64}\sqrt{\sigma^2 + (\mu_1 - \mu_2)^2} \leq \frac{17\sigma}{64} + \frac{17\,|\mu_1 - \mu_2|}{64}.$$

and therefore

$$-\frac{17\sigma}{64} + \frac{47\,|\mu_1 - \mu_2|}{64} \leq |\hat{\mu_1} - \hat{\mu_2}| \leq \frac{17\sigma}{64} + \frac{81\,|\mu_1 - \mu_2|}{64}.$$

Hence, we have

$$-\frac{17\sigma}{81} + \frac{64\,|\hat{\mu}_1 - \hat{\mu}_2|}{81} \leq |\mu_1 - \mu_2| \leq \frac{17\sigma}{47} + \frac{64\,|\hat{\mu}_1 - \hat{\mu}_2|}{47}.$$

This implies that if $|\hat{\mu}_1 - \hat{\mu}_2| \leq 15\gamma/32$, then $|\mu_1 - \mu_2| \leq \gamma$ and we will fit a single gaussian to recover the means. On the other hand, if $|\hat{\mu}_1 - \hat{\mu}_2| \geq 15\gamma/32$, then $|\mu_1 - \mu_2| \leq 13\gamma/81$ and we will use EM algorithm. The sample complexity required is $O(\log \eta^{-1})$ samples.

APPENDIX C

MISSING PROOFS IN CHAPTER 5

C.1 Missing Proofs from Section 5.2

Proof of Lemma 5.2 when $|\cup_{i\in\mathcal{C}}\mathcal{S}(i)|$ is provided. Suppose we are given $|\cup_{i\in\mathcal{C}}\mathcal{S}(i)|$ for all sets $\mathcal{C}\subseteq[n]$ satisfying $|\mathcal{C}|\leq s$. Notice that the set $\cap_{i\in\mathcal{C}}\mathcal{S}(i)$ is equivalent to the set $\mathsf{occ}(C,\mathbf{1}_{|C|})$ or the number of unknown vectors in $\mathcal{V}$ whose restriction to the indices in $\mathcal{C}$ is the all one vector and in particular, $\mathsf{occ}((i),1)=\mathcal{S}(i)$. Note that for each family of t sets $\mathcal{A}_1,\mathcal{A}_2,\ldots,\mathcal{A}_t$, we must have

$$\left|\bigcup_{i=1}^{t}\mathcal{A}_i\right|=\sum_{u=1}^{t}(-1)^{u+1}\sum_{1\leq i_1<i_2<\cdots<i_u\leq t}\left|\bigcap_{b=1}^{u}\mathcal{A}_{i_b}\right|.$$

We now show using induction on s that the quantities

$$\left\{\left|\bigcup_{i\in\mathcal{S}}\mathsf{occ}((i),1)\right|\ \forall\ \mathcal{T}\subseteq[n],|\mathcal{T}|\leq s\right\}$$

are sufficient to compute $|\mathsf{occ}(C,\boldsymbol{a})|$ for all subsets C of indices of size at most s, and any binary vector $\boldsymbol{a}\in\{0,1\}^{\leq s}$.

Base case ($t=1$):

The base case follows since we can infer $|\mathsf{occ}((i),0)|=\ell-|\mathsf{occ}((i),1)|$ from $|\mathsf{occ}((i),1)|$ for all $i\in[n]$.

Inductive Step: Let us assume that the statement is true for $r<s$ i.e., we can compute $|\mathsf{occ}(\mathcal{C},\boldsymbol{a})|$ for all subsets $\mathcal{C}$ satisfying $|\mathcal{C}|\leq r$ and any binary vector $\boldsymbol{a}\in\{0,1\}^{\leq r}$ from the quantities $\left\{\left|\bigcup_{i\in\mathcal{S}}\mathsf{occ}((i),1)\right|\ \forall\ \mathcal{T}\subseteq[n],|\mathcal{T}|\leq r\right\}$ provided as

input. Now, we prove that the statement is true for $r+1$ under the induction hypothesis. Note that we can also rewrite $\mathsf{occ}(\mathcal{C}, \boldsymbol{a})$ for each set $\mathcal{C} \subseteq [n], \boldsymbol{a} \in \{0,1\}^{|\mathcal{C}|}$ as

$$\mathsf{occ}(\mathcal{C}, \boldsymbol{a}) = \bigcap_{j \in \mathcal{C}'} \mathcal{S}(j) \bigcap_{j \in \mathcal{C} \setminus \mathcal{C}'} \mathcal{S}(j)^c$$

where $\mathcal{C}' \subseteq \mathcal{C}$ corresponds to the indices in $\mathcal{C}$ for which the entries in $\boldsymbol{a}$ is $\mathtt{1}$. Fix any set $i_1, i_2, \ldots, i_{r+1} \in [n]$. Then we can compute $\left|\bigcap_{b=1}^{r+1} \mathcal{S}(i_b)\right|$ using the following equation:

$$(-1)^{r+3} \left|\bigcap_{b=1}^{r+1} \mathcal{S}(i_b)\right| = \sum_{u=1}^{r} (-1)^{u+1} \sum_{\substack{j_1, j_2, \ldots, j_u \in \{i_1, i_2, \ldots, i_{r+1}\} \\ j_1 < j_2 < \cdots < j_u}} \left|\bigcap_{b=1}^{u} \mathcal{S}(j_b)\right| - \left|\bigcup_{b=1}^{r+1} \mathcal{S}(i_b)\right|.$$

Finally for each proper subset $\mathcal{Y} \subset \{i_1, i_2, \ldots, i_{r+1}\}$, we can compute

$$\left|\bigcap_{i_b \notin \mathcal{Y}} \mathcal{S}(i_b) \bigcap_{i_b \in \mathcal{Y}} \mathcal{S}(i_b)^c\right|$$

using the following set of equations:

$$\begin{aligned}
\left|\bigcap_{i_b \notin \mathcal{Y}} \mathcal{S}(i_b) \bigcap_{i_b \in \mathcal{Y}} \mathcal{S}(i_b)^c\right| &= \left|\bigcap_{i_b \notin \mathcal{Y}} \mathcal{S}(i_b) \bigcap \Big(\bigcup_{i_b \in \mathcal{Y}} \mathcal{S}(i_b)\Big)^c\right| \\
&= \left|\bigcap_{i_b \notin \mathcal{Y}} \mathcal{S}(i_b)\right| - \left|\bigcap_{i_b \notin \mathcal{Y}} \mathcal{S}(i_b) \bigcap \Big(\bigcup_{i_b \in \mathcal{Y}} \mathcal{S}(i_b)\Big)\right| \\
&= \left|\bigcap_{i_b \notin \mathcal{Y}} \mathcal{S}(i_b)\right| - \left|\bigcup_{i_b \in \mathcal{Y}} \Big(\bigcap_{i_b \notin \mathcal{Y}} \mathcal{S}(i_b) \bigcap \mathcal{S}(i_b)\Big)\right|.
\end{aligned}$$

The first term is already pre-computed and the second term is again a union of intersection of sets. for each $j_b \in \mathcal{Y}$, let us define $\mathcal{H}(j_b) := \bigcap_{i_b \notin \mathcal{Y}} \mathcal{S}(i_b) \bigcap \mathcal{S}(j_b)$. Therefore we have

$$\left| \bigcup_{j_b \in \mathcal{Y}} \mathcal{H}(j_b) \right| = \sum_{u=1}^{|\mathcal{Y}|} (-1)^{u+1} \sum_{\substack{j_1, j_2, \ldots, j_u \in \mathcal{Y} \\ j_1 < j_2 < \cdots < j_u}} \left| \bigcap_{b=1}^{u} \mathcal{H}(j_b) \right|.$$

We can compute $\left|\bigcup_{j_b \in \mathcal{Y}} \mathcal{H}(j_b)\right|$ because the quantities on the right hand side of the equation have already been pre-computed (using our induction hypothesis). Therefore, the lemma is proved.

□

Proof of Lemma 5.2 when $|\cap_{i \in \mathcal{C}} \mathcal{S}(i)|$ is provided. Suppose we are given $|\cap_{i \in \mathcal{C}} \mathcal{S}(i)|$ for all sets $\mathcal{V} \subseteq [n]$ satisfying $|\mathcal{V}| \leq s$. We will omit the subscript $\mathcal{V}$ from hereon for simplicity. As in Lemma 5.2, the set $\cap_{i \in \mathcal{C}} \mathcal{S}(i)$ is equivalent to the set $\mathsf{occ}(C, \mathbf{1}_{|C|})$ or the number of unknown vectors in $\mathcal{V}$ whose restriction to the indices in $\mathcal{C}$ is the all one vector and in particular, $\mathsf{occ}((i), 1) = \mathcal{S}(i)$. We will re-use the equation that for t sets $\mathcal{A}_1, \mathcal{A}_2, \ldots, \mathcal{A}_t$, we must have

$$\left| \bigcup_{i=1}^{t} \mathcal{A}_i \right| = \sum_{u=1}^{t} (-1)^{u+1} \sum_{1 \leq i_1 < i_2 < \cdots < i_u \leq t} \left| \bigcap_{b=1}^{u} \mathcal{A}_{i_b} \right|.$$

We now show using induction on s that the quantities

$$\left\{ \left| \bigcap_{i \in \mathcal{S}} \mathsf{occ}((i), 1) \right| \ \forall\, \mathcal{T} \subseteq [n], |\mathcal{T}| \leq s \right\}$$

are sufficient to compute $|\mathsf{occ}(C, \boldsymbol{a})|$ for all subsets C of indices of size at most s, and any binary vector $\boldsymbol{a} \in \{0,1\}^{\leq s}$.

Base case ($t = 1$):

The base case follows since we can infer $|\mathsf{occ}((i),0)| = \ell - |\mathsf{occ}((i),1)|$ from $|\mathsf{occ}((i),1)|$ for all $i \in [n]$.

Inductive Step: Let us assume that the statement is true for $r < s$ i.e., we can compute $|\mathsf{occ}(\mathcal{C}, \boldsymbol{a})|$ for all subsets $\mathcal{C}$ satisfying $|\mathcal{C}| \leq r$ and any binary vector $\boldsymbol{a} \in \{0,1\}^{\leq r}$ from the quantities $\left\{\left|\bigcap_{i\in\mathcal{S}} \mathsf{occ}((i),1)\right| \ \forall\ \mathcal{T} \subseteq [n], |\mathcal{T}| \leq r\right\}$ provided as input. Now, we prove that the statement is true for $r+1$ under the induction hypothesis. Note that we can also rewrite $\mathsf{occ}(\mathcal{C}, \boldsymbol{a})$ for any set $\mathcal{C} \subseteq [n], \boldsymbol{a} \in \{0,1\}^{|\mathcal{C}|}$ as

$$\mathsf{occ}(\mathcal{C}, \boldsymbol{a}) = \bigcap_{j\in\mathcal{C}'} \mathcal{S}(j) \bigcap_{j\in\mathcal{C}\setminus\mathcal{C}'} \mathcal{S}(j)^c$$

where $\mathcal{C}' \subseteq \mathcal{C}$ corresponds to the indices in $\mathcal{C}$ for which the entries in $\boldsymbol{a}$ is $\mathtt{1}$. Fix any set $i_1, i_2, \ldots, i_{r+1} \in [n]$. Then we can compute $\left|\bigcup_{b=1}^{r+1} \mathcal{S}(i_b)\right|$ using the following equation:

$$\left|\bigcup_{b=1}^{r+1} \mathcal{S}(i_b)\right| = \sum_{u=1}^{r+1} (-1)^{u+1} \sum_{\substack{j_1,j_2,\ldots,j_u \in \{i_1,i_2,\ldots,i_{r+1}\} \\ j_1<j_2<\cdots<j_u}} \left|\bigcap_{b=1}^{u} \mathcal{S}(j_b)\right|.$$

Finally for any proper subset $\mathcal{Y} \subset \{i_1, i_2, \ldots, i_{r+1}\}$, we can compute

$$\left|\bigcap_{i_b\notin\mathcal{Y}} \mathcal{S}(i_b) \bigcap_{i_b\in\mathcal{Y}} \mathcal{S}(i_b)^c\right|$$

using the following set of equations:

$$\begin{aligned}
\left|\bigcap_{i_b\notin\mathcal{Y}} \mathcal{S}(i_b) \bigcap_{i_b\in\mathcal{Y}} \mathcal{S}(i_b)^c\right| &= \left|\bigcap_{i_b\notin\mathcal{Y}} \mathcal{S}(i_b) \bigcap \Big(\bigcup_{i_b\in\mathcal{Y}} \mathcal{S}(i_b)\Big)^c\right| \\
&= \left|\bigcap_{i_b\notin\mathcal{Y}} \mathcal{S}(i_b)\right| - \left|\bigcap_{i_b\notin\mathcal{Y}} \mathcal{S}(i_b) \bigcap \Big(\bigcup_{i_b\in\mathcal{Y}} \mathcal{S}(i_b)\Big)\right| \\
&= \left|\bigcap_{i_b\notin\mathcal{Y}} \mathcal{S}(i_b)\right| - \left|\bigcup_{i_b\in\mathcal{Y}} \Big(\bigcap_{i_b\notin\mathcal{Y}} \mathcal{S}(i_b) \bigcap \mathcal{S}(i_b)\Big)\right|.
\end{aligned}$$

The first term is already pre-computed and the second term is again a union of intersection of sets. For any $i_b \in \mathcal{Y}$, let us define $\mathcal{H}(j_b) := \bigcap_{i_b \notin \mathcal{Y}} \mathcal{S}(i_b) \bigcap \mathcal{S}(j_b)$. Therefore we have

$$\left| \bigcup_{j_b \in \mathcal{Y}} \mathcal{H}(j_b) \right| = \sum_{u=1}^{|\mathcal{Y}|} (-1)^{u+1} \sum_{\substack{j_1, j_2, \ldots, j_u \in \mathcal{Y} \\ j_1 < j_2 < \cdots < j_u}} \left| \bigcap_{b=1}^{u} \mathcal{H}(j_b) \right|.$$

We can compute $\left|\bigcup_{j_b \in \mathcal{Y}} \mathcal{H}(j_b)\right|$ because the quantities on the right hand side of the equation have already been pre-computed (using our induction hypothesis). Therefore, the lemma is proved.

□

Proof of Lemma 5.3. Note that $\mathsf{Trimmed}(\mathcal{V})$ is the largest subset of vectors in $\mathcal{V} \equiv \{\boldsymbol{v}^{(1)}, \boldsymbol{v}^{(2)}, \ldots, \boldsymbol{v}^{(\ell)}\}$ such that the support of any vector in $\mathsf{Trimmed}(\mathcal{V})$ is not contained within the support of any other vector in $\mathsf{Trimmed}(\mathcal{V})$. Let us fix a vector $\boldsymbol{v} \in \mathsf{Trimmed}(\mathcal{V})$. For any other vector $\boldsymbol{v}' \in \mathsf{Trimmed}(\mathcal{V})$ there must exist an index $i_{\boldsymbol{v},\boldsymbol{v}'} \in \mathsf{supp}(\boldsymbol{v})$ such that $i_{\boldsymbol{v},\boldsymbol{v}'} \notin \mathsf{supp}(\boldsymbol{v}')$. Clearly the vector $\boldsymbol{v}$ constrained to the set of indices $\mathcal{C} \triangleq \cup_{\boldsymbol{v}' \in \mathsf{Trimmed}(\mathcal{V}), \boldsymbol{v}' \neq \boldsymbol{v}} \{i_{\boldsymbol{v},\boldsymbol{v}'}\}$ is an all-one vector but $\boldsymbol{v}'_{|\mathcal{C}} \neq \mathbf{1}$ for all $\boldsymbol{v}' \in \mathcal{V}, \boldsymbol{v}' \neq \boldsymbol{v}$. This is true for all vectors in $\mathsf{Trimmed}(\mathcal{V})$ and since $|\mathsf{Trimmed}(\mathcal{V}) \setminus \{\boldsymbol{v}\}| \leq \ell - 1$, we must have $\mathsf{Trimmed}(\mathcal{V})$ to be $(\ell-1)-$good. □

Proof of Lemma 5.4. As stated in the Lemma, suppose it is known if $|\cap_{i \in \mathcal{C}} \mathcal{S}(i)| > 0$ or not for all sets $\mathcal{C} \subseteq [n]$ satisfying $|\mathcal{C}| \leq s + 1$. Further assume that the set of unknown vectors $\mathcal{V}$ is s-good. Consider any vector $\boldsymbol{v} \in \mathsf{Trimmed}(\mathcal{V})$. Since $\mathcal{V}$ is s-good, there must exist an ordered set $\mathcal{C} \subseteq [n]$ such that $\boldsymbol{v}_{|\mathcal{C}}$ is the all 1 vector but $\boldsymbol{v}'_{|\mathcal{C}}$ is not the all 1 vector for any other vector $\boldsymbol{v}' \in \mathsf{Trimmed}(\mathcal{V})$. Therefore, we must have $|\cap_{i \in \mathcal{C}} \mathcal{S}(i)| > 0$. But, on the other hand, notice that if $|\mathsf{Trimmed}(\mathcal{V})| \geq 2$, there must exist an index $j \in \cup_{\boldsymbol{v} \in \mathsf{Trimmed}(\mathcal{V})} \mathsf{supp}(\boldsymbol{v})$ such that $\left|\cap_{i \in \mathcal{C} \cup \{j\}} \mathcal{S}(i)\right| = 0$ since the support

of $\boldsymbol{v}$ does not contain the support of all other vectors. Algorithm 5.2 precisely checks for this condition and therefore this completes the proof.

□

Proof of Lemma 5.5. Consider the special case when $|\mathsf{Trimmed}(\mathcal{V})| = 1$ i.e. there exists a particular vector $\boldsymbol{v}$ in $\mathcal{V}$ whose support subsumes the support of all the other unknown vectors in $\mathcal{V}$. In that case, for each set $\mathcal{C} \subseteq \cup_{\boldsymbol{v} \in \mathsf{Trimmed}(\mathcal{V})} \mathsf{supp}(\boldsymbol{v})$, $|\mathcal{C}| \leq \ell$, we must have that $\left|\bigcup_{i \in \mathcal{C}} \mathcal{S}(i)\right| > 0$ (as there is only a single vector in $\mathsf{Trimmed}(\mathcal{V})$). On the other hand, if $|\mathsf{Trimmed}(\mathcal{V})| \geq 2$, then we know that $\mathsf{Trimmed}(\mathcal{V})$ is $(\ell - 1)$-good and therefore, for each vector $\boldsymbol{v} \in \mathsf{Trimmed}(\mathcal{V})$, there exists an ordered set and an index $\mathcal{C}, \{j\} \subseteq \cup_{\boldsymbol{v} \in \mathsf{Trimmed}(\mathcal{V})} \mathsf{supp}(\boldsymbol{v})$, $|\mathcal{C}| \leq \ell - 1$ such that $\mathcal{C}$ belongs to the support of $\boldsymbol{v}$ but does not belong to the support of any other vector; hence $\left|\bigcap_{i \in \mathcal{C}} \mathcal{S}(i)\right| > 0$ but $\left|\bigcap_{i \in \mathcal{C} \cup \{j\}} \mathcal{S}(i)\right| = 0$. In other words, there exists a set of size ℓ that is a subset of the union of support of vectors in $\mathsf{Trimmed}(\mathcal{V})$ but there does not exist any unknown vector that has $\mathtt{1}$ in all the indices indexed by the aforementioned set. Again, Algorithm 5.2 precisely checks this conditions and therefore this completes the proof. □

Algorithm C.1 ESTIMATE(m, B) Estimating $\mathbb{E}X$ for $X \sim \mathcal{P}$

Require: I.i.d samples $x^{(1)}, x^{(2)}, \ldots, x^{(m)} \sim \mathcal{P}$
1: Set $t = m/B$
2: **for** $i = 1, 2, \ldots, B$ **do**
3: Set Batch i to be the samples $x^{(j)}$ for $j \in \{it + 1, it + 2, \ldots, (i + 1)t\}$.
4: Set $S_1^i = \sum_{j \in \text{ Batch } i} \frac{x^{(j)}}{t}$
5: **end for**
6: Return $\mathsf{median}(\{S_1^i\}_{i=1}^B)$

Lemma C.1. *The set* $\mathsf{Trimmed}(\mathcal{V})$ *is unique.*

Proof. We will prove this lemma by contradiction. Suppose there exists two distinct sets $\mathcal{T}_1, \mathcal{T}_2 \subset \mathcal{V}$ such that $|\mathcal{T}_1| = |\mathcal{T}_2| = |\mathsf{Trimmed}(\mathcal{V})|$. Since $\mathcal{T}_1, \mathcal{T}_2$ are distinct, there must exist a vector $\boldsymbol{v} \in \mathcal{T}_2 \setminus \mathcal{T}_1$. If $\mathsf{supp}(\boldsymbol{v})$ is not contained with the support of some

vector in $\mathcal{T}_1$ and there is no other vector in $\mathcal{V}$ whose support contains $\boldsymbol{v}$, then clearly, $\boldsymbol{v}$ can be added to $\mathcal{T}_1$ implying that $\mathcal{T}_1$ cannot be the largest deduplicated set. On the other hand, suppose $\mathsf{supp}(\boldsymbol{v})$ is contained within the support of some vector $\boldsymbol{v}'$ in $\mathcal{T}_1$. However, this implies that $\mathcal{T}_2$ cannot be a valid deduplicated set as the support of $\boldsymbol{v}$ is contained with the support of $\boldsymbol{v}'$ and therefore, $\boldsymbol{v}$ cannot belong to a deduplicated set. This implies that the vector $\boldsymbol{v}$ cannot exist without violating some constrained of $\mathsf{Trimmed}(\mathcal{V})$ and therefore, the set $\mathsf{Trimmed}(\mathcal{V})$ is unique. □

C.2 Technical Lemmas

Lemma C.2 (Hoeffding's inequality for bounded random variables)**.** *Let $X_1, \ldots, X_m$ be independent random variables strictly bounded in the interval $[a, b]$. Let $\mu = m^{-1} \sum_i \mathbb{E}X_i$. In that case, we must have*

$$\Pr\Big(\Big| \frac{1}{m} \sum_{i=1}^{m} X_i - \mu \Big| \geq t \Big) \leq 2 \exp\Big(- \frac{2mt^2}{(b-a)^2} \Big).$$

Lemma C.3 (Gaussian concentration inequality)**.** *Consider a random variable Z distributed according to $\mathcal{N}(0, \sigma^2)$. In that case, we must have $\Pr(|Z| \geq t) \leq 2\exp(-t^2/2)$ for any $t > 0$.*

Lemma C.4 (Gaussian anti-concentration inequality)**.** *Consider a random variable Z distributed according to $\mathcal{N}(0, \sigma^2)$. In that case, we must have $\Pr(|Z| \leq t) \leq \sqrt{\frac{2}{\pi}} \cdot \frac{t}{\sigma}$ for any $t < \sigma\sqrt{\pi}/\sqrt{2}$.*

Proof. By simple calculations, we can have

$$\Pr(|Z| < t) \leq \int_{-t}^{t} \frac{e^{-x^2/2\sigma^2}}{\sqrt{2\pi\sigma}} dx \leq \sqrt{\frac{2}{\pi}} \cdot \frac{t}{\sigma}.$$

□

Lemma C.5. *Suppose* $|\cup_{\boldsymbol{v}\in\mathcal{V}}\mathsf{supp}(\boldsymbol{v})| \le n/2$. *In that case, we can compute* $\cup_{\boldsymbol{v}\in\mathcal{V}}\mathsf{supp}(\boldsymbol{v})$ *correctly using* $O(\ell^2(R^2+\sigma^2)(\log n)^3/\delta^2)$ *samples with probability at least* $1-1/\mathsf{poly}(n)$.

Proof. For each $i \in [n]$, suppose we want to test whether $i \in \cup_{\boldsymbol{v}\in\mathcal{V}}\mathsf{supp}(\boldsymbol{v})$ or not. Consider the random variable $y^2\boldsymbol{x}_i^2$ when $(\boldsymbol{x}, y) \sim \mathcal{P}_r$. Notice that

$$\begin{aligned}\mathbb{E}y^2\boldsymbol{x}_i^2 &= \frac{1}{\ell}\sum_{\boldsymbol{v}\in\mathcal{V}}\mathbb{E}y^2\boldsymbol{x}_i^2 \mid \boldsymbol{v} \\ &= \frac{1}{\ell}\sum_{\boldsymbol{v}\in\mathcal{V}}\Big(\sum_{j\in[n]}\boldsymbol{v}_j^2 + 2\boldsymbol{v}_i^2\Big)\begin{cases} = \frac{1}{\ell}\sum_{\boldsymbol{v}\in\mathcal{V}}||\boldsymbol{v}||_2^2 \text{ if } |\mathcal{S}_{\mathcal{V}}(i)| = 0 \\ \ge \frac{1}{\ell}\sum_{\boldsymbol{v}\in\mathcal{V}}||\boldsymbol{v}||_2^2 + \frac{2\delta^2}{\ell} \text{ if } |\mathcal{S}_{\mathcal{V}}(i)| \neq 0\end{cases}\end{aligned}$$

where the final inequality follows from the fact that the magnitude of any non-zero entry of any unknown vector must be at least δ. For simplicity of notation, we will denote $A = \frac{1}{\ell}\sum_{\boldsymbol{v}\in\mathcal{V}}||\boldsymbol{v}||_2^2$ to be average norm of the unknown vectors. We will estimate $\mathbb{E}y^2\boldsymbol{x}_i^2$ by computing the following sample average

$$\frac{\ell}{m}\cdot\sum_{j=1}^{m}\Big(y^{(j)}\boldsymbol{x}_i^{(j)}\Big)^2.$$

From the definition of $\mathcal{P}_r$, we must have $y \sim \mathcal{N}(0, \zeta^2+\sigma^2), |\zeta| \le R$ since $\boldsymbol{v} \in \{0,1\}^n, ||\boldsymbol{v}||_2 \le R$ for all $\boldsymbol{v} \in \mathcal{V}$. By using Gaussian concentration inequalities, we must have $\Pr(|y| > t) \le \exp(-t^2/2(R^2+\sigma^2))$. Therefore, with probability $1-n^{-10}$, we have $|y| < 20\sqrt{R^2+\sigma^2}\log n$. Similarly, with probability $1-n^{-10}$, $|\boldsymbol{x}_i|$ is bounded from above by $20\log n$. a Subsequently, we use Hoeffding's inequality to say that

$$\Pr\Big(\Big|\frac{\ell}{m}\cdot\sum_{j=1}^{m}\Big(y^{(j)}\boldsymbol{x}_i^{(j)}\Big)^2 - \mathbb{E}y^2\boldsymbol{x}_i^2\Big| \ge \frac{\delta^2 2}{2\ell}\Big) \le \exp\Big(-\Omega\Big(\frac{m\delta^2}{\ell^2(R^2+\sigma^2)(\log n)^2}\Big)\Big).$$

Hence, with $m = O(\ell^2(R^2+\sigma^2)(\log n)^3/\delta^2)$ samples, we can estimate if $\big|\bigcap_{i\in\mathcal{C}}\mathcal{S}_{\mathcal{V}}(i)\big| > 0$ or not correctly with probability at least $1-1/\mathsf{poly}(n)$. We can take a union bound

over all the n indices to estimate $\mathbb{E}y^2\boldsymbol{x}_i^2$ correctly within an additive error of $\delta^2/2\ell$ for all $i \in [n]$. We will cluster all the indices such that a pair of distinct indices $u, v \in [n]$ are in the same group if

$$\left| \frac{\ell}{m} \cdot \sum_{j=1}^{m} \left(y^{(j)} \boldsymbol{x}_u^{(j)} \right)^2 - \frac{\ell}{m} \cdot \sum_{j=1}^{m} \left(y^{(j)} \boldsymbol{x}_v^{(j)} \right)^2 \right| \leq \frac{\delta^2}{\ell}.$$

Clearly, any two indices $u, v \in [n]$ that satisfy $|\mathcal{S}_{\mathcal{V}}(u)| = |\mathcal{S}_{\mathcal{V}}(v)| = 0$ must belong to the same cluster. Since the size of the union of the support is at most $n/2$, the largest cluster must correspond to the indices where the entry is zero in all the unknown vectors. Subsequently, all those indices that do not belong to the largest cluster (after the clustering step) must belong to $\cap_{\boldsymbol{v} \in \mathcal{V}} \mathsf{supp}(\boldsymbol{v})$. Furthermore, no index $i \in [n]$ such that $|\mathcal{S}_{\mathcal{V}}(i)| \neq 0$ can belong to the largest cluster. This complete the proof of the lemma. □

Finally, we will also use the following well-known lemma stating that we can compute estimates of the expectation of any one-dimensional random variable with only a few samples similar to sub-gaussian random variables.

Lemma C.6. *For a random variable $x \sim \mathcal{P}$, there exists an algorithm (see Algorithm C.1 in Appendix C.1) that can compute an estimate u of $\mathbb{E}x$ such that $|u - \mathbb{E}x| \leq \epsilon$ with $O(\log \gamma^{-1}\mathbb{E}x^2/\epsilon^2)$ with probability at least $1 - \gamma$.*

Proof of Lemma C.6. Suppose we obtain m independent samples $x^{(1)}, x^{(2)}, \ldots, x^{(m)} \sim \mathcal{P}$. We use the median of means trick to compute u, an estimate of $\mathbb{E}x$. We will partition m samples obtained from $\mathcal{P}$ into $B = \lceil m/m' \rceil$ batches each containing m' samples each. In that case let us denote S^j to be the sample mean of the j^{th} batch i.e.

$$S^j = \sum_{s \in \text{Batch } j} \frac{x^{(s)}}{m'}.$$

We will estimate the true mean $\mathbb{E}x$ by computing u where $u \triangleq \mathsf{median}(\{S^j\}_{j=1}^B)$. For a fixed batch j, we can use Chebychev's inequality to say that

$$\Pr\Big(\big|S^j - \mathbb{E}x\big| \geq \epsilon\Big) \leq \frac{\mathbb{E}x^2}{t\epsilon^2} \leq \frac{1}{3}$$

for $t = O(\mathbb{E}x^2/\epsilon^2)$. Therefore for each batch j, we define an indicator random variable $Z_j = \mathbf{1}[|S^j - \mathbb{E}x| \geq \epsilon]$ and from our previous analysis we know that the probability of Z_j being 1 is less than 1/3. It is clear that $\mathbb{E}\sum_{j=1}^B Z_j \leq B/3$ and on the other hand $|u - \mathbb{E}x| \geq \epsilon$ iff $\sum_{j=1}^B Z_j \geq B/2$. Therefore, due to the fact that Z_j's are independent, we can use Chernoff bound to conclude the following:

$$\Pr\Big(|u - \mathbb{E}x| \geq \epsilon\Big) \leq \Pr\Big(\Big|\sum_{j=1}^B Z_j - \mathbb{E}\sum_{j=1}^B Z_j\Big| \geq \frac{\mathbb{E}\sum_{j=1}^B Z_j}{2}\Big) \leq 2e^{-B/36}.$$

Hence, for $B = 36\log\gamma^{-1}$, the estimate u is at most ϵ away from the true mean $\mathbb{E}x$ with probability at least $1-\gamma$. Therefore the sufficient sample complexity is $m = O(\log\gamma^{-1}\mathbb{E}x^2/\epsilon^2)$. □

APPENDIX D

USEFUL THEORETICAL TOOLS

D.1 Minimum Distance Estimator and Scheffe Estimator

Minimum Distance Estimator: We next review the minimum distance estimator from [51], which is a general method to determine the sample complexity of a learning problem. We start with the following definition that characterizes a widely used combinatorial property of

Definition D.1. *The Vapnik-Chervonenkis (VC) dimension of a family of sets $\mathcal{H}$, denoted by $\mathsf{VC}(\mathcal{H})$, is defined as the cardinality of the largest set $\mathcal{C}$ such that $|\mathcal{H} \cap \mathcal{C}| = 2^{|\mathcal{C}|}$ i.e. $\mathcal{H} \cap \mathcal{C}$ contains all the subsets of $\mathcal{C}$.*

Suppose we are given a set of n samples $x^1, x^2, \ldots, x^n$ distributed according to some unknown distribution f and we are estimating f from an infinite class of distributions $\mathcal{F}$ parameterized by $\theta \in \Theta$. Let us define the family of sets $\mathcal{A}$ to be

$$\mathcal{A} = \{\{f_\theta > f_{\theta'}\} : \theta, \theta' \in \Theta, \theta \neq \theta'\}.$$

the union of set of intervals in the sample space where one of the distribution has larger density than the other. For some set $A \in \mathcal{A}$, let $\mu_{n,A} = (1/n) \sum_{i=1}^{n} \mathbb{1}\left[x^i \in A\right]$ be the empirical mass of the distribution f to the set A using n samples. For a particular distribution f_θ, define Δ_θ

$$\Delta_\theta = \sup_{A \in \mathcal{A}} \left| \int_A f_\theta - \mu_{n,A} \right|$$

to be the distance of f_θ to the empirical distribution. We choose the best estimate $f_{\theta^\star}$ such that $\Delta_{\theta^\star}$ satisfies the following:

$$\Delta_{\theta^\star} \leq \inf_{\theta \in \Theta} \Delta_\theta + \frac{1}{n}.$$

In Chapter 6, [51], the following theoretical guarantee on the candidate distribution $f_{\theta^\star} \in \mathcal{F}$ returned by the minimum distance estimator is proved:

$$||f_{\theta^\star} - f||_{\mathsf{TV}} \leq 3 \inf_{\theta \in \Theta} ||f_\theta - f||_{\mathsf{TV}} + 2\Delta + \frac{3}{2n}$$

where $\Delta = \sup_{A \in \mathcal{A}} \left| \int_A f - \mu_{n,A} \right|$. Further, Chapter 4, [51] shows that $\mathbb{E}\Delta$ is related to the VC-Dimension of the family of sets $\mathcal{A}$ in the following way:

$$\mathbb{E}\Delta \leq \sqrt{\frac{\mathsf{VC}(A)}{n}}.$$

In order to bound the deviation of Δ from its expected value, we can use McDiarmid's inequality (Chapter 2, [51]) to conclude that with probability at least $1 - 2\eta$

$$\Delta \leq \mathbb{E}_{\sim f}\Delta + \sqrt{\frac{\log \eta^{-1}}{2n}} \leq \sqrt{\frac{2VC(\mathcal{A}) + \log \eta^{-1}}{2n}}.$$

Putting everything together, we get that the following statement holds

$$||f_{\theta^\star} - f||_{\mathsf{TV}} \leq 3 \inf_{\theta \in \Theta} ||f_\theta - f||_{\mathsf{TV}} + 2\sqrt{\frac{2VC(\mathcal{A}) + \log \eta^{-1}}{2n}} + \frac{3}{2n} \tag{D.1}$$

with probability at least $1 - \eta$.

Although the minimum distance estimator returns a distribution from an infinite class of candidate distributions such that the returned distribution is close in total

variation distance, equation D.1 does not provide any guarantee on parameter estimation when the unknown distribution $f \in \mathcal{F}$. In order to convert the TV-distance guarantee to a parameter estimation guarantee, we also need to show that the candidate distributions that have parameters far apart from the parameters of the unknown distribution f also have sufficiently large total variation distance from f. Now, since $f \in \mathcal{F}$, we must have $\inf_{\theta \in \Theta} ||f_\theta - f||_{\mathsf{TV}} = 0$. Therefore, for sufficiently large n, we can prove that with high probability $||f_{\theta^\star} - f||_{\mathsf{TV}}$ is small and thus, the minimum distance estimator will not select a candidate distribution having parameters far apart from that of f. Finally, we want to point out over here that the minimum distance estimator has an unbounded time complexity as the class of candidate distributions is infinite. In the following sub-section, we will review a different estimator called the Scheffe estimator that takes a finite set of candidate distributions and therefore runs in time that is polynomial in the size of the candidate set.

Scheffe Estimator: Suppose we are given a set of n samples $x^1, x^2, \ldots, x^n$ distributed according to some unknown distribution f and we are estimating f from a finite class of distributions $\mathcal{F} \equiv \{f^1, f^2, \ldots, f^k\}$. As before, let us define the family of sets $\mathcal{A}$ to be

$$\mathcal{A} = \{\{x : f(x) > g(x)\} : f, g \in \mathcal{F}, f \neq g\}.$$

the union of set of intervals in the sample space where one of the distributions in $\mathcal{F}$ has larger density than the other. The Scheffe estimator runs a tournament such that for any pair of distinct $f^i, f^j \in \mathcal{F}$ such that $i < j$, for $A_{ij} = \{x : f^i(x) > f^j(x)\}$, f^i wins against f^j if

$$\left| \int_{A_{ij}} f^i - \mu_{n,A_{ij}} \right| < \left| \int_{A_{ij}} f^j - \mu_{n,A_{ij}} \right|$$

and f^j wins otherwise. Finally, the winner of the Scheffe tournament (the distribution returned by the Scheffe estimator) is the candidate distibution having most number of wins with ties broken arbitrarily. In Chapter 6, [51], the following theoretical guarantee on the candidate distribution $f_{\theta^\star} \in \mathcal{F}$ returned by the Scheffe estimator is proved:

$$||f_{\theta^\star} - f||_{\mathsf{TV}} \leq 9 \inf_{\theta \in \Theta} ||f_\theta - f||_{\mathsf{TV}} + 8\Delta$$

where $\Delta = \sup_{A \in \mathcal{A}} \left| \int_A f - \mu_{n,A} \right|$. Again, from Theorem 2.1 and Theorem 2.2 [51], we can directly show that

$$\mathbb{E}\Delta \leq \sqrt{\log |\mathcal{F}|} n.$$

As before, we use McDiarmid's inequality to conclude that with probability at least $1 - 2\eta$

$$\Delta \leq \mathbb{E}_{\sim f}\Delta + \sqrt{\frac{\log \eta^{-1}}{2n}} \leq \sqrt{\frac{2\log |\mathcal{F}| + \log \eta^{-1}}{2n}}.$$

Putting everything together, we get that the following statement holds

$$||f_{\theta^\star} - f||_{\mathsf{TV}} \leq 9 \inf_{\theta \in \Theta} ||f_\theta - f||_{\mathsf{TV}} + 8\sqrt{\frac{2\log |\mathcal{F}| + \log \eta^{-1}}{2n}} \tag{D.2}$$

with probability at least $1 - \eta$.

The Scheffe estimator returns a distribution close in total variation distance from the unknown distribution f from a finite set of candidate distributions $\mathcal{F}$. As such, the running time of the Scheffe estimator is $O(n\,|\mathcal{F}|^2)$. As our goal is to estimate the parameters of f, we will consider $\mathcal{F}$ to be the same class of distributions as f i.e.

we will construct a well-designed cover of the infinite class of candidate distributions considered for the minimum distance estimator. However as $\mathcal{F}$ is finite in this case, f might not belong to $\mathcal{F}$. Therefore, converting the total variation guarantee for the Scheffe estimator is trickier and is a two step process. We need to show that the candidate distribution in $\mathcal{F}$ with parameters close to the parameters in f must be close in TV distance as well (by showing an upper bound on the TV distance). On the other hand, the candidate distribution in $\mathcal{F}$ whose parameters are far away from the parameters in f must be far away in TV distance as well (by showing a lower bound in TV distance). As such, for sufficiently large number of samples, the TV distance between the Scheffe estimator and f must be small implying that the Scheffe estimator could not have been a candidate distribution with parameters far away from the ground truth.

D.2 Jennrich's Algorithm for Unique Canonical Polyadic (CP) Decomposition

In this section, we state Jennrich's Algorithm for CP decomposition (see Sec 3.3, [112]) that we use in this paper. Recall that we are provided a symmetric tensor $\mathcal{A}$ of order 3 and rank R as input i.e. a tensor $\mathcal{A}$ that can be expressed in the form below:

$$\mathcal{A} = \sum_{r=1}^{R} \underbrace{\boldsymbol{z}^r \otimes \boldsymbol{z}^r \otimes \boldsymbol{z}^r}.$$

Our goal is to uniquely recover the latent vectors $\boldsymbol{z}^1, \boldsymbol{z}^2, \ldots, \boldsymbol{z}^R$ from the input tensor $\mathcal{A}$ provided that the vectors $\boldsymbol{z}^1, \boldsymbol{z}^2, \ldots, \boldsymbol{z}^R$ are linearly independent. Let $\mathcal{A}_{\cdot,\cdot,i}$ denote the i^{th} matrix slice through $\mathcal{A}$.

For the sake of completeness, we describe in brief why Algorithm D.1 works. Note that $\sum_{i\in[n]} \boldsymbol{a}_i \mathcal{A}_{\cdot,\cdot,i}$ is the weighted sum of matrix slices through $\mathcal{A}$ each weighted by $\boldsymbol{a}_i$. Therefore, it is easy to see that

Algorithm D.1 JENNRICH'S ALGORITHM($\mathcal{A}$)

Require: A symmetric rank-R tensor $\mathcal{A} \in \mathbb{R}^n \otimes \mathbb{R}^n \otimes \mathbb{R}^n$ of order 3.

1: Choose $\boldsymbol{a}, \boldsymbol{b} \in \mathbb{R}^n$ uniformly at random such that it satisfies $||\boldsymbol{a}||_2 = ||\boldsymbol{b}||_2 = 1$.
2: Compute $\boldsymbol{T}^{(1)} \triangleq \sum_{i\in[n]} \boldsymbol{a}_i \mathcal{A}_{\cdot,\cdot,i}, \boldsymbol{T}^{(2)} \triangleq \sum_{i\in[n]} \boldsymbol{b}_i \mathcal{A}_{\cdot,\cdot,i}$.
3: **if** $\mathsf{rank}(T^1) < R$ **then**
4: Return Error
5: **end if**
6: Solve the general eigen-value problem $\boldsymbol{T}^{(1)}\boldsymbol{v} = \lambda_v \boldsymbol{T}^{(2)}\boldsymbol{v}$.
7: Return the eigen-vectors $\boldsymbol{v}$ corresponding to the non-zero eigen-values.

$$\boldsymbol{T}^{(1)} \triangleq \sum_{i\in[n]} \boldsymbol{a}_i \mathcal{A}_{\cdot,\cdot,i} = \sum_{r=1}^{R} \langle \boldsymbol{z}^r, \boldsymbol{a}\rangle \boldsymbol{z}^r \otimes \boldsymbol{z}^r = \boldsymbol{Z}\boldsymbol{D}^{(1)}\boldsymbol{Z}^T$$

$$\boldsymbol{T}^{(2)} \triangleq \sum_{i\in[n]} \boldsymbol{b}_i \mathcal{A}_{\cdot,\cdot,i} = \sum_{r=1}^{R} \langle \boldsymbol{z}^r, \boldsymbol{b}\rangle \boldsymbol{z}^r \otimes \boldsymbol{z}^r = \boldsymbol{Z}\boldsymbol{D}^{(2)}\boldsymbol{Z}^T$$

where $\boldsymbol{Z}$ is a $n \times R$ matrix whose columns form the vectors $\boldsymbol{z}^1, \boldsymbol{z}^2, \ldots, \boldsymbol{z}^R$; $\boldsymbol{D}^{(1)}, \boldsymbol{D}^{(2)}$ are $R \times R$ diagonal matrices whose entry at the i^{th} position in the diagonal is $\langle \boldsymbol{z}^r, \boldsymbol{a}\rangle$ and $\langle \boldsymbol{z}^r, \boldsymbol{b}\rangle$ respectively. Clearly, the matrices $\boldsymbol{T}^{(1)}, \boldsymbol{T}^{(2)}$ are of rank R if and only if the vectors $\boldsymbol{z}^1, \boldsymbol{z}^2, \ldots, \boldsymbol{z}^R$ are linearly independent and therefore, this condition is easy to verify in Steps 3-5. Now if the sufficiency condition is met, then the generalized eigenvalue decomposition will reveal the unknown latent vectors since the eigenvalues are going to be distinct with probability 1.

BIBLIOGRAPHY

[1] Acharya, Jayadev, Bhattacharyya, Arnab, and Kamath, Pritish. Improved bounds for universal one-bit compressive sensing. In *2017 IEEE International Symposium on Information Theory (ISIT)* (2017), IEEE, pp. 2353–2357.

[2] Acharya, Jayadev, Diakonikolas, Ilias, Li, Jerry, and Schmidt, Ludwig. Sample-optimal density estimation in nearly-linear time. In *Proceedings of the Twenty-Eighth Annual ACM-SIAM Symposium on Discrete Algorithms* (2017), SIAM, pp. 1278–1289.

[3] Achlioptas, Dimitris, and McSherry, Frank. On spectral learning of mixtures of distributions. In *Conference on Learning Theory* (2005).

[4] Ai, Albert, Lapanowski, Alex, Plan, Yaniv, and Vershynin, Roman. One-bit compressed sensing with non-gaussian measurements. *Linear Algebra and its Applications 441* (2014), 222–239.

[5] Aitkin, M. Mixture applications of the em algorithm in glim. *COMPSTAT 1980* (1980), 537–541.

[6] Anandkumar, Animashree, Ge, Rong, Hsu, Daniel, Kakade, Sham M, and Telgarsky, Matus. Tensor decompositions for learning latent variable models. *Journal of machine learning research 15* (2014), 2773–2832.

[7] Arias-Castro, Ery, and Pu, Xiao. A simple approach to sparse clustering. *Computational Statistics & Data Analysis 105* (2017), 217–228.

[8] Arora, Sanjeev, Ge, Rong, Kannan, Ravi, and Moitra, Ankur. Computing a nonnegative matrix factorization—provably. *SIAM Journal on Computing 45*, 4 (2016), 1582–1611.

[9] Arora, Sanjeev, and Kannan, Ravi. Learning mixtures of arbitrary gaussians. In *Symposium on Theory of Computing* (2001).

[10] Ashtiani, Hassan, Ben-David, Shai, Harvey, Nicholas JA, Liaw, Christopher, Mehrabian, Abbas, and Plan, Yaniv. Near-optimal sample complexity bounds for robust learning of Gaussian mixtures via compression schemes. *Journal of the ACM 67*, 6 (2020), 1–42.

[11] Azizyan, Martin, Singh, Aarti, and Wasserman, Larry. Minimax theory for high-dimensional gaussian mixtures with sparse mean separation. *Advances in Neural Information Processing Systems 26* (2013), 2139–2147.

[12] Bakshi, Ainesh, Diakonikolas, Ilias, Hopkins, Samuel B., Kane, Daniel, Karmalkar, Sushrut, and Kothari, Pravesh K. Outlier-robust clustering of gaussians and other non-spherical mixtures. In *61st IEEE Annual Symposium on Foundations of Computer Science, FOCS 2020, Durham, NC, USA, November 16-19, 2020* (2020), IEEE, pp. 149–159.

[13] Balakrishnan, Sivaraman, Wainwright, Martin J, and Yu, Bin. Statistical guarantees for the em algorithm: From population to sample-based analysis. *The Annals of Statistics 45*, 1 (2017), 77–120.

[14] Baraniuk, Richard, Davenport, Mark, DeVore, Ronald, and Wakin, Michael. The johnson-lindenstrauss lemma meets compressed sensing. *preprint 100*, 1 (2006), 0.

[15] Baraniuk, Richard, Davenport, Mark, DeVore, Ronald, and Wakin, Michael. A simple proof of the restricted isometry property for random matrices. *Constructive Approximation 28*, 3 (2008), 253–263.

[16] Barsov, SS, and Ul'yanov, Vladimir V. Estimates of the proximity of gaussian measures. In *Sov. Math., Dokl* (1987), vol. 34, pp. 462–466.

[17] Batu, Tugkan, Kannan, Sampath, Khanna, Sanjeev, and McGregor, Andrew. Reconstructing strings from random traces. In *Symposium on Discrete Algorithms* (2004).

[18] Belkin, Mikhail, and Sinha, Kaushik. Polynomial learning of distribution families. In *Foundations of Computer Science* (2010).

[19] Bishop, Christopher M. Latent variable models. In *Learning in graphical models*. Springer, 1998, pp. 371–403.

[20] Bishop, Christopher M, et al. *Neural networks for pattern recognition*. Oxford university press, 1995.

[21] Blackwell, Ekin, De Leon, Carlos F Mendes, and Miller, Gregory E. Applying mixed regression models to the analysis of repeated-measures data in psychosomatic medicine. *Psychosomatic medicine 68*, 6 (2006), 870–878.

[22] Boche, Holger, Calderbank, Robert, Kutyniok, Gitta, and Vybíral, Jan. A survey of compressed sensing. In *Compressed Sensing and its Applications*. Springer, 2015, pp. 1–39.

[23] Böhning, Dankmar, Seidel, Wilfried, Alfó, Macro, Garel, Bernard, Patilea, Valentin, and Walther, Günther. Advances in mixture models. *Computational Statistics & Data Analysis 51*, 11 (2007), 5205–5210.

[24] Borwein, P., and Erdélyi, T. Littlewood-type problems on subarcs of the unit circle. *Indiana University Mathematics Journal* (1997).

[25] Borwein, Peter. *The Prouhet—Tarry—Escott Problem*. Springer New York, New York, NY, 2002, pp. 85–95.

[26] Boucheron, Stéphane, Lugosi, Gábor, and Massart, Pascal. *Concentration inequalities: A nonasymptotic theory of independence*. Oxford university press, 2013.

[27] Brennan, Matthew, and Bresler, Guy. Optimal average-case reductions to sparse pca: From weak assumptions to strong hardness. In *Conference on Learning Theory* (2019), PMLR, pp. 469–470.

[28] Bun, Mark, Kamath, Gautam, Steinke, Thomas, and Wu, Zhiwei Steven. Private hypothesis selection. *arXiv preprint arXiv:1905.13229* (2019).

[29] Candes, Emmanuel J. The restricted isometry property and its implications for compressed sensing. *Comptes rendus mathematique 346*, 9-10 (2008), 589–592.

[30] Candès, Emmanuel J, Romberg, Justin, and Tao, Terence. Robust uncertainty principles: exact signal reconstruction from highly incomplete frequency information. *IEEE Transactions on Information Theory 52*, 2 (2006), 489–509.

[31] Canonne, Clément L, Kamath, Gautam, McMillan, Audra, Ullman, Jonathan, and Zakynthinou, Lydia. Private identity testing for high-dimensional distributions. *arXiv preprint arXiv:1905.11947* (2019).

[32] Chaganty, Arun Tejasvi, and Liang, Percy. Spectral experts for estimating mixtures of linear regressions. In *International Conference on Machine Learning* (2013), PMLR, pp. 1040–1048.

[33] Chan, Siu-On, Diakonikolas, Ilias, Servedio, Rocco A, and Sun, Xiaorui. Learning mixtures of structured distributions over discrete domains. In *Symposium on Discrete Algorithms* (2013).

[34] Chan, Siu-On, Diakonikolas, Ilias, Servedio, Rocco A, and Sun, Xiaorui. Efficient density estimation via piecewise polynomial approximation. In *Proceedings of the forty-sixth annual ACM symposium on Theory of computing* (2014), ACM, pp. 604–613.

[35] Chen, Yudong, Yi, Xinyang, and Caramanis, Constantine. A convex formulation for mixed regression with two components: Minimax optimal rates. In *Conference on Learning Theory* (2014), PMLR, pp. 560–604.

[36] Cheraghchi, Mahdi, Gabrys, Ryan, Milenkovic, Olgica, and Ribeiro, João. Coded trace reconstruction. *arXiv e-prints* (Mar 2019), arXiv:1903.09992.

[37] Curtiss, DR. Recent extentions of descartes' rule of signs. *Annals of Mathematics* (1918), 251–278.

[38] Dasgupta, Sanjoy. Learning mixtures of gaussians. In *Foundations of Computer Science* (1999), pp. 634–644.

[39] Daskalakis, Constantinos, and Kamath, Gautam. Faster and sample near-optimal algorithms for proper learning mixtures of gaussians. In *Conference on Learning Theory* (2014).

[40] Daskalakis, Constantinos, Tzamos, Christos, and Zampetakis, Manolis. Ten steps of em suffice for mixtures of two gaussians. In *Conference on Learning Theory* (2017), PMLR, pp. 704–710.

[41] Daskalakis, Constantinos, Tzamos, Christos, and Zampetakis, Manolis. Ten steps of em suffice for mixtures of two Gaussians. In *Conference on Learning Theory* (2017), pp. 704–710.

[42] Davies, Sami, Mazumdar, Arya, Pal, Soumyabrata, and Rashtchian, Cyrus. Lower bounds on the total variation distance between mixtures of two gaussians. *arXiv preprint arXiv:2109.01064* (2021).

[43] Davies, Sami, Racz, Miklos Z., and Rashtchian, Cyrus. Reconstructing Trees from Traces. *arXiv e-prints* (Feb 2019), arXiv:1902.05101.

[44] Day, Neil E. Estimating the components of a mixture of normal distributions. *Biometrika 56*, 3 (1969), 463–474.

[45] De, Anindya, O'Donnell, Ryan, and Servedio, Rocco A. Optimal mean-based algorithms for trace reconstruction. In *Symposium on Theory of Computing* (2017).

[46] De, Anindya, O'Donnell, Ryan, and Servedio, Rocco A. Sharp bounds for population recovery. *CoRR abs/1703.01474* (2017).

[47] De Veaux, Richard D. Mixtures of linear regressions. *Computational Statistics & Data Analysis 8*, 3 (1989), 227–245.

[48] De Wolf, Ronald. Efficient data structures from unionfree families of sets, 2012.

[49] Deb, Partha, and Holmes, Ann M. Estimates of use and costs of behavioural health care: a comparison of standard and finite mixture models. *Health economics 9*, 6 (2000), 475–489.

[50] Dempster, AP, Laird, NM, and Rubin, DB. Maximum likelihood from incomplete data. *J*.

[51] Devroye, Luc, and Lugosi, Gábor. *Combinatorial methods in density estimation*. Springer Science & Business Media, 2012.

[52] Devroye, Luc, Mehrabian, Abbas, and Reddad, Tommy. The total variation distance between high-dimensional gaussians. *arXiv preprint arXiv:1810.08693* (2018).

[53] Diakonikolas, Ilias. Learning structured distributions. *Handbook of Big Data 267* (2016).

[54] Diakonikolas, Ilias, Kane, Daniel M, and Stewart, Alistair. Statistical query lower bounds for robust estimation of high-dimensional gaussians and gaussian mixtures. In *2017 IEEE 58th Annual Symposium on Foundations of Computer Science (FOCS)* (2017), IEEE, pp. 73–84.

[55] Diakonikolas, Ilias, Kane, Daniel M, and Stewart, Alistair. List-decodable robust mean estimation and learning mixtures of spherical gaussians. In *Proceedings of the 50th Annual ACM SIGACT Symposium on Theory of Computing* (2018), ACM, pp. 1047–1060.

[56] Donoho, David, and Stodden, Victoria. When does non-negative matrix factorization give a correct decomposition into parts? In *Advances in neural information processing systems* (2004), pp. 1141–1148.

[57] Donoho, DL. Compressed sensing. *IEEE Transactions on Information Theory 52*, 4 (2006), 1289–1306.

[58] Doss, Natalie, Wu, Yihong, Yang, Pengkun, and Zhou, Harrison H. Optimal estimation of high-dimensional Gaussian mixtures. *arXiv preprint arXiv:2002.05818* (2020).

[59] D'yachkov, Arkadii G, Vorobyev, IV, Polyanskii, NA, and Shchukin, V Yu. Bounds on the rate of superimposed codes. In *2014 IEEE International Symposium on Information Theory* (2014), IEEE, pp. 2341–2345.

[60] Erdös, Paul, Frankl, Peter, and Füredi, Zoltán. Families of finite sets in which no set is covered by the union ofr others. *Israel Journal of Mathematics 51*, 1-2 (1985), 79–89.

[61] Everitt, Brian. *Finite mixture distributions*. Springer Science & Business Media, 2013.

[62] Faria, Susana, and Soromenho, Gilda. Fitting mixtures of linear regressions. *Journal of Statistical Computation and Simulation 80*, 2 (2010), 201–225.

[63] Feldman, Jon, O'Donnell, Ryan, and Servedio, Rocco A. Learning mixtures of product distributions over discrete domains. *SIAM Journal on Computing* (2008).

[64] Feller, Avi, Greif, Evan, Ho, Nhat, Miratrix, Luke, and Pillai, Natesh. Weak separation in mixture models and implications for principal stratification. *arXiv preprint arXiv:1602.06595* (2016).

[65] Flodin, Larkin, Gandikota, Venkata, and Mazumdar, Arya. Superset technique for approximate recovery in one-bit compressed sensing. In *Advances in Neural Information Processing Systems* (2019), pp. 10387–10396.

[66] Füredi, Zoltán. On r-cover-free families. *Journal of Combinatorial Theory, Series A 73*, 1 (1996), 172–173.

[67] Gandikota, Venkata, Mazumdar, Arya, and Pal, Soumyabrata. Recovery of sparse linear classifiers from mixture of responses.

[68] Gandikota, Venkata, Mazumdar, Arya, and Pal, Soumyabrata. Recovery of sparse linear classifiers from mixture of responses. In *Advances in Neural Information Processing Systems 33: NeurIPS 2020, December 6-12, 2020, virtual* (2020).

[69] Gandikota, Venkata, Mazumdar, Arya, and Pal, Soumyabrata. Support recovery of sparse signals from a mixture of linear measurements. *arXiv preprint arXiv:2106.05951* (2021).

[70] Gantmakher, Feliks Ruvimovich. *The theory of matrices*, vol. 131. American Mathematical Soc., 1959.

[71] Gopi, Sivakant, Netrapalli, Praneeth, Jain, Prateek, and Nori, Aditya. One-bit compressed sensing: Provable support and vector recovery. In *International Conference on Machine Learning* (2013), pp. 154–162.

[72] Hardt, Moritz, and Price, Eric. Tight bounds for learning a mixture of two gaussians. In *Symposium on Theory of Computing* (2015).

[73] Hardy, Godfrey Harold, Wright, Edward Maitland, et al. *An introduction to the theory of numbers*. Oxford university press, 1979.

[74] Harper, F Maxwell, and Konstan, Joseph A. The movielens datasets: History and context. *Acm transactions on interactive intelligent systems (tiis) 5*, 4 (2015), 1–19.

[75] Hartung, Lisa, Holden, Nina, and Peres, Yuval. Trace reconstruction with varying deletion probabilities. In *Workshop on Analytic Algorithmics and Combinatorics* (2018).

[76] Heinrich, Philippe, and Kahn, Jonas. Strong identifiability and optimal minimax rates for finite mixture estimation. *The Annals of Statistics 46*, 6A (2018), 2844–2870.

[77] Ho, Nhat, and Nguyen, XuanLong. Convergence rates of parameter estimation for some weakly identifiable finite mixtures. *The Annals of Statistics 44*, 6 (2016), 2726–2755.

[78] Holden, Nina, and Lyons, Russell. Lower bounds for trace reconstruction. *arXiv:1808.02336* (2018).

[79] Holden, Nina, Pemantle, Robin, and Peres, Yuval. Subpolynomial trace reconstruction for random strings and arbitrary deletion probability. *arXiv:1801.04783* (2018).

[80] Holenstein, Thomas, Mitzenmacher, Michael, Panigrahy, Rina, and Wieder, Udi. Trace reconstruction with constant deletion probability and related results. In *Symposium on Discrete Algorithms* (2008).

[81] Hopkins, Samuel B, and Li, Jerry. Mixture models, robustness, and sum of squares proofs. In *Symposium on Theory of Computing* (2018).

[82] Hsu, Daniel, and Kakade, Sham M. Learning mixtures of spherical gaussians: moment methods and spectral decompositions. In *Innovations in Theoretical Computer Science* (2013).

[83] Huang, Mian, Li, Runze, and Wang, Shaoli. Nonparametric mixture of regression models. *Journal of the American Statistical Association 108*, 503 (2013), 929–941.

[84] Huleihel, Wasim, Mazumdar, Arya, Médard, Muriel, and Pal, Soumyabrata. Same-cluster querying for overlapping clusters. In *Advances in Neural Information Processing Systems* (2019), pp. 10485–10495.

[85] Jacques, Laurent, Laska, Jason N, Boufounos, Petros T, and Baraniuk, Richard G. Robust 1-bit compressive sensing via binary stable embeddings of sparse vectors. *IEEE Transactions on Information Theory 59*, 4 (2013), 2082–2102.

[86] Jordan, Michael I, and Jacobs, Robert A. Hierarchical mixtures of experts and the em algorithm. *Neural computation 6*, 2 (1994), 181–214.

[87] Kalai, Adam, Moitra, Ankur, and Valiant, Gregory. Disentangling Gaussians. *Communications of the ACM 55*, 2 (2012), 113–120.

[88] Kalai, Adam Tauman, Moitra, Ankur, and Valiant, Gregory. Efficiently learning mixtures of two gaussians. In *Proceedings of the forty-second ACM symposium on Theory of computing* (2010), ACM, pp. 553–562.

[89] Kannan, Sampath, and McGregor, Andrew. More on reconstructing strings from random traces: Insertions and deletions. In *International Symposium on Information Theory* (2005).

[90] Kautz, William, and Singleton, Roy. Nonrandom binary superimposed codes. *IEEE Transactions on Information Theory 10*, 4 (1964), 363–377.

[91] Khalili, Abbas, and Chen, Jiahua. Variable selection in finite mixture of regression models. *Journal of the american Statistical association 102*, 479 (2007), 1025–1038.

[92] Klusowski, Jason M, Yang, Dana, and Brinda, WD. Estimating the coefficients of a mixture of two linear regressions by expectation maximization. *IEEE Transactions on Information Theory 65*, 6 (2019), 3515–3524.

[93] Kolda, Tamara G, and Bader, Brett W. Tensor decompositions and applications. *SIAM review 51*, 3 (2009), 455–500.

[94] Kontkanen, Petri, Myllymaki, Petri, Roos, Teemu, Tirri, Henry, Valtonen, Kimmo, and Wettig, Hannes. Topics in probabilistic location estimation in wireless networks. In *2004 IEEE 15th International Symposium on Personal, Indoor and Mobile Radio Communications* (2004), vol. 2, IEEE, pp. 1052–1056.

[95] Kothari, Pravesh K, Steinhardt, Jacob, and Steurer, David. Robust moment estimation and improved clustering via sum of squares. In *Proceedings of the 50th Annual ACM SIGACT Symposium on Theory of Computing* (2018), ACM, pp. 1035–1046.

[96] Krasikov, I., and Roditty, Y. On a reconstruction problem for sequences. *Journal of Combinatorial Theory, Series A* (1997).

[97] Krishnamurthy, Akshay, Mazumdar, Arya, McGregor, Andrew, and Pal, Soumyabrata. Sample complexity of learning mixture of sparse linear regressions. In *Advances in Neural Information Processing Systems (NeurIPS)* (2019).

[98] Krishnamurthy, Akshay, Mazumdar, Arya, McGregor, Andrew, and Pal, Soumyabrata. Trace reconstruction: Generalized and parameterized. *arXiv preprint arXiv:1904.09618* (2019).

[99] Krishnamurthy, Akshay, Mazumdar, Arya, McGregor, Andrew, and Pal, Soumyabrata. Algebraic and analytic approaches for parameter learning in mixture models. In *Proc. 31st International Conference on Algorithmic Learning Theory (ALT)* (2020), vol. 117, pp. 468–489.

[100] Kwon, Jeongyeol, and Caramanis, Constantine. Em converges for a mixture of many linear regressions. In *International Conference on Artificial Intelligence and Statistics* (2020), PMLR, pp. 1727–1736.

[101] Kwon, Jeongyeol, Qian, Wei, Caramanis, Constantine, Chen, Yudong, and Davis, Damek. Global convergence of the em algorithm for mixtures of two component linear regression. In *Conference on Learning Theory* (2019), PMLR, pp. 2055–2110.

[102] Lehmann, Erich L, and Romano, Joseph P. *Testing statistical hypotheses*. Springer Science & Business Media, 2006.

[103] Li, Yuanzhi, and Liang, Yingyu. Learning mixtures of linear regressions with nearly optimal complexity. In *Conference On Learning Theory* (2018), PMLR, pp. 1125–1144.

[104] Liang, Percy, Bouchard-Côté, Alexandre, Klein, Dan, and Taskar, Ben. An end-to-end discriminative approach to machine translation. In *Proceedings of the 21st International Conference on Computational Linguistics and the 44th annual meeting of the Association for Computational Linguistics* (2006), Association for Computational Linguistics, pp. 761–768.

[105] Lindsay, Bruce G, and Lesperance, Mary L. A review of semiparametric mixture models. *Journal of statistical planning and inference 47*, 1-2 (1995), 29–39.

[106] Liu, Hui, Darabi, Houshang, Banerjee, Pat, and Liu, Jing. Survey of wireless indoor positioning techniques and systems. *IEEE Transactions on Systems, Man, and Cybernetics, Part C (Applications and Reviews) 37*, 6 (2007), 1067–1080.

[107] Lu, Yu, and Zhou, Harrison H. Statistical and computational guarantees of lloyd's algorithm and its variants. *arXiv preprint arXiv:1612.02099* (2016).

[108] Manole, Tudor, and Ho, Nhat. Uniform convergence rates for maximum likelihood estimation under two-component gaussian mixture models. *arXiv preprint arXiv:2006.00704* (2020).

[109] Mazumdar, Arya, and Pal, Soumyabrata. Recovery of sparse signals from a mixture of linear samples. In *International Conference on Machine Learning (ICML)* (2020).

[110] McGregor, Andrew, Price, Eric, and Vorotnikova, Sofya. Trace reconstruction revisited. In *European Symposium on Algorithms* (2014).

[111] McLachlan, Geoffrey J, and Basford, Kaye E. *Mixture models: Inference and applications to clustering*, vol. 38. M. Dekker New York, 1988.

[112] Moitra, Ankur. Algorithmic aspects of machine learning. *Lecture notes* (2014).

[113] Moitra, Ankur. *Algorithmic aspects of machine learning*. Cambridge University Press, 2018.

[114] Moitra, Ankur, and Valiant, Gregory. Settling the polynomial learnability of mixtures of gaussians. In *Foundations of Computer Science* (2010).

[115] Moosman, F, and Peel, D. Finite mixture models. *Wiley 3* (2000), 4.

[116] Nazarov, Fedor, and Peres, Yuval. Trace reconstruction with $\exp(O(n^{1/3})$ samples. In *Symposium on Theory of Computing* (2017).

[117] Neyman, Jerzy, and Pearson, Egon Sharpe. *Contributions to the theory of testing statistical hypotheses*. University of California Press, 2020.

[118] Pearson, Karl. Contributions to the mathematical theory of evolution. *Philosophical Transactions of the Royal Society of London. A 185* (1894), 71–110.

[119] Peres, Yuval, and Zhai, Alex. Average-case reconstruction for the deletion channel: Subpolynomially many traces suffice. In *Symposium on Foundations of Computer Science* (2017).

[120] Plan, Yaniv, and Vershynin, Roman. One-bit compressed sensing by linear programming. *Communications on Pure and Applied Mathematics 66*, 8 (2013), 1275–1297.

[121] Quattoni, Ariadna, Collins, Michael, and Darrell, Trevor. Conditional random fields for object recognition. In *Advances in neural information processing systems* (2005), pp. 1097–1104.

[122] Rabanser, Stephan, Shchur, Oleksandr, and Günnemann, Stephan. Introduction to tensor decompositions and their applications in machine learning. *arXiv preprint arXiv:1711.10781* (2017).

[123] Ruszinkó, Miklós. On the upper bound of the size of the r-cover-free families. *Journal of Combinatorial Theory, Series A 66*, 2 (1994), 302–310.

[124] Scott, Alex D. Reconstructing sequences. *Discrete Mathematics* (1997).

[125] Sedghi, Hanie, Janzamin, Majid, and Anandkumar, Anima. Provable tensor methods for learning mixtures of generalized linear models. In *Artificial Intelligence and Statistics* (2016), PMLR, pp. 1223–1231.

[126] Settles, Burr. Active learning literature survey.

[127] Shen, Yanyao, and Sanghavi, Sujay. Iterative least trimmed squares for mixed linear regression. *arXiv preprint arXiv:1902.03653* (2019).

[128] Sidiropoulos, Nicholas D, and Bro, Rasmus. On the uniqueness of multilinear decomposition of n-way arrays. *Journal of Chemometrics: A Journal of the Chemometrics Society 14*, 3 (2000), 229–239.

[129] Slawski, Martin, Hein, Matthias, and Lutsik, Pavlo. Matrix factorization with binary components. In *Advances in Neural Information Processing Systems* (2013), pp. 3210–3218.

[130] Song, Weixing, Yao, Weixin, and Xing, Yanru. Robust mixture regression model fitting by laplace distribution. *Computational Statistics & Data Analysis 71* (2014), 128–137.

[131] Städler, Nicolas, Bühlmann, Peter, and Van De Geer, Sara. l1-penalization for mixture regression models. *Test 19*, 2 (2010), 209–256.

[132] Stigler, Stephen M. *The history of statistics: The measurement of uncertainty before 1900*. Harvard University Press, 1986.

[133] Stinson, Douglas R, and Wei, Ruizhong. Generalized cover-free families. *Discrete Mathematics 279*, 1-3 (2004), 463–477.

[134] Sun, Yuekai, Ioannidis, Stratis, and Montanari, Andrea. Learning mixtures of linear classifiers. In *ICML* (2014), pp. 721–729.

[135] Suresh, Ananda Theertha, Orlitsky, Alon, Acharya, Jayadev, and Jafarpour, Ashkan. Near-optimal-sample estimators for spherical gaussian mixtures. In *Advances in Neural Information Processing Systems* (2014), pp. 1395–1403.

[136] Tarter, Michael E, and Lock, Michael D. *Model-free curve estimation*, vol. 56. CRC Press, 1993.

[137] Titterington, D Michael, Smith, Adrian FM, and Makov, Udi E. *Statistical analysis of finite mixture distributions*. Wiley, 1985.

[138] Van der Vaart, Aad W. *Asymptotic statistics*, vol. 3. Cambridge university press, 2000.

[139] Verzelen, Nicolas, and Arias-Castro, Ery. Detection and feature selection in sparse mixture models. *The Annals of Statistics 45*, 5 (2017), 1920–1950.

[140] Viele, Kert, and Tong, Barbara. Modeling with mixtures of linear regressions. *Statistics and Computing 12*, 4 (2002), 315–330.

[141] Viswanathan, Krishnamurthy, and Swaminathan, Ram. Improved string reconstruction over insertion-deletion channels. In *Symposium on Discrete Algorithms* (2008).

[142] Wang, Taiyao, and Paschalidis, Ioannis Ch. Convergence of parameter estimates for regularized mixed linear regression models. *arXiv preprint arXiv:1903.09235* (2019).

[143] Weldon, Walter Frank Raphael. I. certain correlated variations in crangon vulgaris. *Proceedings of the Royal Society of London 51*, 308-314 (1892), 1–21.

[144] Weldon, Walter Frank Raphael. Ii. on certain correlated variations in carcinus mænas. *Proceedings of the Royal Society of London 54*, 326-330 (1894), 318–329.

[145] Wolfe, John H. Normix: Computational methods for estimating the parameters of multivariate normal mixtures of distributions. Tech. rep., NAVAL PERSONNEL RESEARCH ACTIVITY SAN DIEGO CALIF, 1967.

[146] Wu, Shanshan, Dimakis, Alexandros G, and Sanghavi, Sujay. Learning distributions generated by one-layer relu networks. *Advances in neural information processing systems 32* (2019), 8107–8117.

[147] Wu, Yihong, and Yang, Pengkun. Optimal estimation of gaussian mixtures via denoised method of moments. *Annals of Statistics 48*, 4 (2020), 1981–2007.

[148] Wu, Yihong, and Yang, Pengkun. Polynomial methods in statistical inference: Theory and practice. *Foundations and Trends® in Communications and Information Theory 17*, 4 (2020), 402–586.

[149] Wu, Yihong, and Zhou, Harrison H. Randomly initialized EM algorithm for two-component Gaussian mixture achieves near optimality in $O(\sqrt{n})$ iterations. *arXiv preprint arXiv:1908.10935* (2019).

[150] Xu, Ji, Hsu, Daniel, and Maleki, Arian. Global analysis of expectation maximization for mixtures of two gaussians. *arXiv preprint arXiv:1608.07630* (2016).

[151] Yi, Xinyang, Caramanis, Constantine, and Sanghavi, Sujay. Alternating minimization for mixed linear regression. In *International Conference on Machine Learning* (2014), pp. 613–621.

[152] Yi, Xinyang, Caramanis, Constantine, and Sanghavi, Sujay. Solving a mixture of many random linear equations by tensor decomposition and alternating minimization. *arXiv preprint arXiv:1608.05749* (2016).

[153] Yin, Dong, Pedarsani, Ramtin, Chen, Yudong, and Ramchandran, Kannan. Learning mixtures of sparse linear regressions using sparse graph codes. *IEEE Transactions on Information Theory 65*, 3 (2019), 1430–1451.

[154] Yin, Dong, Pedarsani, Ramtin, Chen, Yudong, and Ramchandran, Kannan. Learning mixtures of sparse linear regressions using sparse graph codes. *IEEE Transactions on Information Theory 65*, 3 (2019), 1430–1451.

[155] Zhang, Yuchen, Chen, Xi, Zhou, Dengyong, and Jordan, Michael I. Spectral methods meet em: A provably optimal algorithm for crowdsourcing. *Advances in neural information processing systems 27* (2014), 1260–1268.

[156] Zhong, Kai, Jain, Prateek, and Dhillon, Inderjit S. Mixed linear regression with multiple components. In *NIPS* (2016), pp. 2190–2198.

[157] Zhu, Hong-Tu, and Zhang, Heping. Hypothesis testing in mixture regression models. *Journal of the Royal Statistical Society: Series B (Statistical Methodology) 66*, 1 (2004), 3–16.