Proceedings of Machine Learning Research vol 75:1–20, 2018 31st Annual Conference on Learning Theory

Learning Mixtures of Linear Regressions with Nearly Optimal Complexity

Yuanzhi Li YUANZHIL@CS.PRINCETON.EDU
Princeton University, Computer Science Department

Yingyu Liang YLIANG@CS.WISC.EDU
University of Wisconsin-Madison, Computer Sciences Department

Editors: Sebastien Bubeck, Vianney Perchet and Philippe Rigollet

Abstract

Mixtures of Linear Regressions (MLR) is an important mixture model with many applications. In this model, each observation is generated from one of the several unknown linear regression components, where the identity of the generated component is also unknown. Previous works either assume strong assumptions on the data distribution or have high complexity. This paper proposes a fixed parameter tractable algorithm for the problem under general conditions, which achieves global convergence and the sample complexity scales nearly linearly in the dimension. In particular, different from previous works that require the data to be from the standard Gaussian, the algorithm allows the data from Gaussians with different covariances. When the conditional number of the covariances and the number of components are fixed, the algorithm has nearly optimal sample complexity $N = \tilde{O}(d)$ as well as nearly optimal computational complexity $\tilde{O}(Nd)$, where d is the dimension of the data space. To the best of our knowledge, this approach provides the first such recovery guarantee for this general setting.

1. Introduction

This paper studies the problem of learning Mixtures of Linear Regressions (MLR). In this model, one is given i.i.d. observations from a mixture of k unknown linear regression components, and the goal is to recover the hidden parameters in the k linear regressions. In particular, each component i has a sampling probability p_i, a data distribution $\mathcal{D}_i$, a hidden parameter w_i, and each observation (x, α) is generated by first sampling a component i according to p_i's, then sampling x from $\mathcal{D}_i$ and setting $\alpha = \langle x, w_i \rangle$.

The MLR model is a popular mixture model and has many applications due to its effectiveness in capturing non-linearity and its model simplicity (De Veaux, 1989; Jordan and Jacobs, 1994; Faria and Soromenho, 2010; Zhong et al., 2016). It has also been a recent theoretical topic for analyzing benchmark algorithms for nonconvex optimization (e.g., (Chaganty and Liang, 2013; Klusowski et al., 2017)) or designing new algorithms (e.g., (Chen et al., 2014)). However, most of the existing works either restrict to very special settings (e.g., x of different components all from the standard Gaussian, or only $k = 2$ components) (Chen et al., 2014; Yi et al., 2014; Zhong et al., 2016; Balakrishnan et al., 2017; Klusowski et al., 2017), or have high sample or computational complexity far from optimal (Chaganty and Liang, 2013; Sedghi et al., 2016).

Moreover, to the best of our knowledge, all the existing works require the $\mathcal{D}_i$ being identical. Most works requiring them to be the standard Gaussian, with the exception of those using tensor

methods. However, since the ultimate goal of MLR is to use different linear classifiers to capture different types of data points, it is important to allow different types to have different covariances, and was mentioned as an important open problem in (Sedghi et al., 2016).

We propose a novel fixed parameter tractable algorithm for learning Mixtures of Linear Regressions in a setting significantly more general than those in previous works. In particular, our setting allows $k \geq 2$ components of data from different distributions $\mathcal{D}_i = \mathcal{N}(0, \mathbf{\Sigma}_i^2)$ with $\mathbf{I} \preceq \mathbf{\Sigma}_i \preceq \sigma \mathbf{I}$, and only requires a necessary separation between the ground truth parameters that any two weight parameters should be at least Δ apart for some separation parameter Δ. The algorithm can recover the ground truth to any additive error ε using $N = d \log\left(\frac{d}{\varepsilon}\right) \text{poly}\left(\frac{k\sigma}{p_{\min}\Delta}\right) + n$ examples and $Nd \cdot \text{polylog}(k, d, \sigma, \frac{1}{\varepsilon}, \frac{1}{\Delta}, \frac{1}{p_{\min}})$ computational time, where $p_{\min} = \min_i p_i$ and n is a minor term for fixed k. It is tractable in the number of components k, the bound on the differences between the different variances σ, the separation parameter Δ, and the minimum proportion $p_{\min}$ of the components. When these parameters are fixed, it can recover the ground truth to any additive error ε, with nearly optimal sample complexity which is nearly linear in d, and with nearly optimal computational complexity which is nearly linear in Nd.

Novel algorithmic techniques are proposed since existing ones are not known to generalize to this setting. One main technical contribution of our work is a new "method of moments descent" technique, that allows us to break ties between different mixture components *gradually*: Unlike most of the previous algorithms which use method of moments to obtain a warm start in one shot, we use it to find a direction to perform one "gradient descent" step and gradually refine our solution. We believe our techniques are potentially useful in even more general cases.

Organization. Section 2 reviews the related work, and Section 3 formalizes the problem and presents our result. An overview of the intuition for designing and analyzing the algorithm is provided in Section 4 while the algorithm and the key lemmas are presented in Section 5. The formal proofs are provided in the appendix.

2. Related Work

Mixtures of Linear Regressions is a popular mixture model (e.g., (De Veaux, 1989; Grün et al., 2007) and (Faria and Soromenho, 2010)), also known as Hierarchical Mixture of Experts in (Jordan and Jacobs, 1994) in the machine learning community. It has many applications, such as trajectory clustering (Gaffney and Smyth, 1999) and phase retrieval (Balakrishnan et al., 2017), and has as special cases some popular models, such as piecewise linear regression and locally linear regression.

Learning MLR in general is NP-hard (Yi et al., 2014). Recent interests have been in providing various efficient algorithms for recovering the parameters in MLR under assumptions about the data generation model (Chaganty and Liang, 2013; Chen et al., 2014; Yi et al., 2014; Zhong et al., 2016; Klusowski et al., 2017). They are either under restricted assumptions about the data (mixtures of two component or x all from the standard Gaussian) (Chen et al., 2014; Yi et al., 2014; Balakrishnan et al., 2017; Klusowski et al., 2017), or have high sample or computational complexity (Chaganty and Liang, 2013; Sedghi et al., 2016).

Some works study specific algorithms for the problem, such as the Expectation Maximization (EM) algorithm (Khalili and Chen, 2007; Yi et al., 2014; Balakrishnan et al., 2017; Klusowski et al., 2017). It is known that without careful initialization EM is only guaranteed to have local convergence (Klusowski et al., 2017). A grid search method for initialization is proposed in (Yi et al.,

2014) but is only for the two-component case. It is unclear how to generalize these guarantees to our more general setting where the data x from different components are from different Gaussians. Moreover, EM also often suffers from a high computational cost.

Another line of works used tensor methods for MLR (Chaganty and Liang, 2013; Sedghi et al., 2016). The third-order moment is directly estimated in (Chaganty and Liang, 2013) using samples from Gaussian distribution and is estimated from a linear regression problem in (Sedghi et al., 2016). A significant drawback of tensor methods is high sample and computational complexity, due to the high cost in estimating and operating over the tensors.

(Chen et al., 2014) provided a convex relaxation formulation and showed that their algorithm is information-theoretically optimal. However, it is only for the two-component case and suffers from high computational cost in nuclear norm minimization.

(Zhong et al., 2016) provided a non-convex objective function that is locally strongly convex in the neighborhood of the ground truth, and proposed to first use a tensor method for initialization and then optimize the provided objective, achieving a global convergence guarantee. The overall algorithm is fixed parameter tractable in the number of components, and achieves nearly optimal sample and time complexity when this parameter is constant. However, it requires all components have the standard Gaussian distribution. It is unclear how to generalize the result to our more general setting where the data x from different components are from different Gaussians. Furthermore, due to the tensor initialization, the algorithm needs complicated assumptions on the moments, while our only essential assumption is that the weight parameters can be separated, which is much simpler and more general (in fact, it is essentially necessary for obtaining any recovery guarantees).

(Yi et al., 2016) gives an improved way of using the tensor method plus alternative minimization so the sample complexity linearly depend on d. However, their algorithm requires that all the data are from the standard Gaussian, and the sample complexity also depends on the minimal singular value of certain moment matrix, which can be $\Delta^{\Omega(k)}$ small in our setting.

3. Problem Definition and Our Result

In the Mixtures of Linear Regressions (MLR) model, the data $(x, \alpha) \in \mathbb{R}^{d+1}$ is generated by

$$z \sim \text{multinomial}(p),\ x \sim \mathcal{D}_z,\ \alpha = \langle w_z, x \rangle \tag{1}$$

where $p \in \mathbb{R}^k$ is the proportion of different components satisfying $\sum_{i=1}^k p_i = 1$, $\mathcal{D}_i$ is the distribution of the i-th component, and $\{w_i \in \mathbb{R}^d\}_{i=1}^k$ are the ground truth parameters. The goal is then to recover $\{w_i\}_i$ given a dataset $\{(x_\ell, \alpha_\ell)\}_{\ell=1}^N$, where each (x_ℓ, α_ℓ) is i.i.d. generated by (1).

Notations. $[k]$ is used to denote the set $\{1, 2, \ldots, k\}$. With high probability or w.h.p. means with probability $1 - d^{-C}$ for some sufficiently large constant $C > 1$. $1_{\mathcal{E}}$ is the indicator function of the event $\mathcal{E}$.

Assumptions. We make the following assumptions about the distributions $\mathcal{D}_i$'s and w_i's.

(A1) Each $\mathcal{D}_i = \mathcal{N}(0, \mathbf{\Sigma}_i^2)$, where $\mathbf{I} \preceq \mathbf{\Sigma}_i \preceq \sigma \mathbf{I}$ for some $\sigma \geq 1$.

(A2) For every $i \in [k]$, $p_i \geq p_{\min}$ for some $p_{\min} > 0$.

(A3) Each $\|w_i\|_2 \leq 1$, and for some $\Delta \in (0, 1)$, $\|w_i - w_j\|_2 \geq \Delta$ for any $i \neq j \in [k]$.

Assumption **(A1)** allows the data x in different components to come from Gaussian distributions with different unknown covariances.[1] This is more general than all the previous works that assume they all come from the standard Gaussian distribution. This also causes difficulties in applying known techniques for MLR, and thus requires new algorithmic approaches. Moreover, our result can also be easily generalized to the case that the mixtures come from *different* subspaces. That is, there can be zero singular values for Σ_i's and the *non-zero* singular values of each component is in $[1, \sigma]$.

Assumption **(A2)** controls the imbalance of the components. We should require that there are enough data from each component so that it is possible to recover the corresponding parameter. On the other hand, our technique can also be generalized to the case when there is enough difference between the probabilities. In this case, we could also treat some components as noise and only recover the leading ones.

Assumption **(A3)** assumes that the ground truth parameters are separated vectors, which is indeed required for exact recovery. Previous works also in general have some form of separation assumptions, many of which are much more sophisticated than ours (e.g., (Zhong et al., 2016; Yi et al., 2016)).

Our result. We are now ready to present our result formally.

Theorem 1 (Main) *Assume the model (1) and assumptions **(A1)-(A3)**. Then Algorithm 6 takes* $N = d \log\left(\frac{d}{\varepsilon}\right) \cdot poly\left(\frac{k\sigma}{\Delta p_{\min}}\right) + \left(\frac{\sigma}{\Delta p_{\min}}\right)^{O(k^2)}$ *data points and in time* $Nd \cdot polylog(k, d, \sigma, \frac{1}{\Delta}, \frac{1}{p_{\min}}, \frac{1}{\varepsilon})$ *outputs a set of vectors* $\{v_i\}_{i=1}^k$ *that with high probability satisfy*

$$\|v_i - w_{\pi(i)}\|_2 \leq \varepsilon, \forall i \in [k], \text{ for some permutation } \pi.$$

The theorem shows that the proposed algorithm achieves global convergence. The run time is polylog in $1/\varepsilon$ for recovery error ε, i.e., the algorithm can achieve exact recovery efficiently. Furthermore, in the case where k, σ, $p_{\min}$, and Δ are fixed constants, the sample complexity is nearly linear in the dimension d of the data space, which is nearly optimal in the key parameter d. The algorithm still works for wider range of k, σ, $p_{\min}$, and Δ, but with an exponential dependence on k.

Table 1 shows the comparison with some recent works. Since for $k = 2$ our settings and results subsumes the existing ones, we mainly compare to previous works handling multiple components $k \geq 2$. Algorithms using the tensor method have poly$(1/\varepsilon)$ dependence (Chaganty and Liang, 2013; Yi et al., 2014; Sedghi et al., 2016). This can be improved by using tensor method only for initialization. (Zhong et al., 2016) provided such an algorithm fixed parameter tractable in the number of components, achieving $N = \tilde{O}(k^k d)$ sample complexity and $\tilde{O}(Nd)$ computational complexity. However, the result is only for the case where the components have data x from the same distribution $\mathcal{D}_i = \mathcal{N}(0, \mathbf{I})$. (Yi et al., 2016) provided an algorithm with sample complexity nearly linear in d and polynomial in k but again it is only for the case with $\mathcal{D}_i = \mathcal{N}(0, \mathbf{I})$, and furthermore, the sample complexity depends on the minimal singular value of certain moment matrix, which can

1. In the standard linear regression model, the covariance of x can be assumed to be the identity by doing a linear transformation. However, in the mixture of linear regression models, different components have different covariances and thus can not be simultaneously transformed to the identity since which data point comes from which component is unknown.

	main model assumptions	sample complexity N	computational complexity
(Yi et al., 2016)	$\mathcal{D}_i = \mathcal{N}(0, \mathbf{I}), k \geq 2$, separation $\Delta > 0$, singular value of some moment matrix σ_k	$\text{poly}(k)\frac{d}{\sigma_k^5 \Delta^2}$	$\text{poly}(k)d^3$
(Zhong et al., 2016)	$\mathcal{D}_i = \mathcal{N}(0, \mathbf{I}), k \geq 2$, separation $\Delta > 0$	$O(d(k\log(d))^k)$	$O(Nd\log(d/\varepsilon))$
(Sedghi et al., 2016)	$\mathcal{D}_i$ are the same, $k \geq 2$, singular values of weight matrix $\geq s > 0$	$O\left(\frac{k^4 d^3}{\varepsilon^2 s^2}\right)$ for Gaussian input	much higher than $\tilde{O}(d^2)$
(Klusowski et al., 2017)	$\mathcal{D}_i = \mathcal{N}(0, \mathbf{I}), k = 2$, local convergence of EM algorithm	$\tilde{O}(d)$	$\tilde{O}(Nd)$
Ours	$\mathcal{D}_i = \mathcal{N}(0, \mathbf{\Sigma}_i^2), \mathbf{I} \preceq \mathbf{\Sigma}_i \preceq \sigma\mathbf{I}, k \geq 2$, separation $\|w_i - w_j\| \geq \Delta > 0 (\forall i \neq j)$	$d\log\left(\frac{d}{\varepsilon}\right)\text{poly}\left(\frac{k\sigma}{\Delta}\right)$ + minor term	$\tilde{O}(Nd)$

Table 1: Comparison with some recent related works. Please refer to the papers for details about the model assumptions and dependence on some other less important parameters, which are omitted here for clarity. In particular, the separation parameters in the related work have different meaning from ours and more complicated.

also be $\left(\frac{1}{\Delta}\right)^k$ small in our setting. (Sedghi et al., 2016) provided algorithms for the case where there are $k \geq 2$ components and $\mathcal{D}_i$ are the same (but can be distributions other than Gaussians). It is based on tensor methods and when applied to Gaussian inputs has high sample and computational complexity.

We also note that it is interesting to compare to results for learning mixture of Gaussians. When the covariance matrix is not axis-aligned, to the best of our knowledge, there is no algorithm for learning mixture of Gaussians with sample complexity linear in the dimension. Thus, solving the mixture of Gaussian first and then rescale the covariances to identity would clearly fail in our setting. Our result shows how to make use of this small amount of side information (the label α) to lower the sample and computational complexity significantly. We refer to for example (Ashtiani et al., 2017) for some discussions.

4. Overview

For the major part of our paper we will focus on learning the weight for one of the components. This can be iterated straightforwardly to learn all the weights, which will be presented at the end.

Our algorithm for learning one weight has two phases. In the first phase, we use method of moments to obtain a warm start. In the second phase, we use gradient descent on a *concave* function to get a more accurate solution.

Method of moments algorithm On a high level, our algorithm is based on the following simple strategy: At each iteration t, we maintain a vector a_t, and the hope is that $\min_{i\in[k]}\{\|\mathbf{\Sigma}_i(w_i - a_t)\|_2\}$ is getting smaller and smaller as t grows, so eventually a_t will be sufficiently close to one w_i. Since $\alpha - \langle a_t, x\rangle = \langle x, w_z - a_t\rangle$ comes from a mixture of one dimension Gaussian distributions with variances $\{\|\mathbf{\Sigma}_i(w_i - a_t)\|_2^2\}_{i=1}^k$, existing algorithms such as (Moitra and Valiant, 2010) can be used to estimate them. Suppose the next vector a_{t+1} is simply chosen as $a_t + \eta r$ for a random vector $r \sim \mathcal{N}(0, \mathbf{I})$. With at least $1/4$ probability, we know that r is positively correlated with $w_j - a_t$ for $j = \arg\min_i\{\|\mathbf{\Sigma}_i(w_i - a_t)\|_2^2\}$, and thus $\|\mathbf{\Sigma}_j(w_j - a_t - \eta r)\|_2^2$ will be smaller than $\|\mathbf{\Sigma}_j(w_j - a_t)\|_2^2$ for sufficiently small η. If this happens, we can let $a_{t+1} = a_t + \eta r$ as the next vector. This process is fundamentally different from many of the existing tie breaking algorithms such as (Li and Yuan, 2017), since we do not have any control over which component the algorithm

is converging to: the algorithm may switch target components on the fly arbitrarily, but the minimal of $\{\|\mathbf{\Sigma}_i(w_i - a_t)\|_2^2\}_{i=1}^k$ is always decreasing.

However, this simple strategy is too expensive in terms of the sample and computational complexity. In each iteration, since r is just a random vector, $\|\mathbf{\Sigma}_j(w_j - a_t - \eta r)\|_2^2$ can only be smaller than $\|\mathbf{\Sigma}_j(w_j - a_t)\|_2^2$ for a factor no more than $\frac{1}{d}$. Thus, we need at least d iterations to finish the whole process. Moreover, to guarantee decreasing, we need to estimate $\|\mathbf{\Sigma}_i(w_i - a_t)\|_2^2$ to accuracy at least $O\left(\frac{1}{d}\right)$ in each iteration, requiring a lot of samples.

The first key idea of our algorithm is to replace sampling from $\mathcal{N}(0, \mathbf{I})$ by sampling from $\mathcal{N}(0, \mathbf{U}\mathbf{U}^\top)$ for some $\mathbf{U} \in \mathbb{R}^{d\times k}$ whose span is known to contain a vector with good correlation with $\mathbf{\Sigma}_j(w_j - a_t)$. To get this subspace, we rely on the method of moments. Note that

$$\mathbb{E}[(\alpha - \langle a_t, x\rangle)^2 xx^\top] = \sum_{i=1}^k p_i \left(2\mathbf{\Sigma}_i^2(w_i - a_t)(w_i - a_t)^\top \mathbf{\Sigma}_i^2 + \|\mathbf{\Sigma}_i(w_i - a_t)\|_2^2 \mathbf{\Sigma}_i^2\right). \quad (2)$$

When all $\mathbf{\Sigma}_i = \mathbf{I}$, we have $\mathbb{E}[(\alpha - \langle a_t, x\rangle)^2 xx^\top] \propto \mathbf{I} + \mathbf{U}\mathbf{U}^\top$ for some $\mathbf{U} \in \mathbb{R}^{d\times k}$ whose span is the subspace spanned by $\mathbf{\Sigma}_i^2(w_i - a_t)$'s. In this case, using a random vector from $\mathbf{U}$ will make the per-iteration improvement as large as $1/k$, much better than a random vector from the entire space.

However, such simple process does not carry on to the case when $\mathbf{\Sigma}_i$'s are different, since they are reweighed by $\|\mathbf{\Sigma}_i(w_i - a_t)\|_2^2$ in the summation (2). As mentioned, we have little control over this reweighing so $\sum_{i=1}^k p_i \|\mathbf{\Sigma}_i(w_i - a_t)\|_2^2 \mathbf{\Sigma}_i^2$ can be arbitrarily away from $\mathbf{I}$.

The second key idea of our algorithm is to combine higher moments with the polynomial method to obtain a good subspace $\mathbf{U}$. We will use a set of carefully designed coefficients $c_0, \cdots, c_k$ such that in the summation $\sum_i c_i \mathbb{E}[(\alpha - \langle a_t, x\rangle)^{2i} xx^\top]$, the $\mathbf{\Sigma}_i^2$ terms will get canceled and all the $\mathbf{\Sigma}_i^2(w_i - a_t)(w_i - a_t)^\top \mathbf{\Sigma}_i^2$ terms get preserved. The $\{c_i\}_{i=0}^k$ are the coefficients of a polynomial constructed to have properties that can ensure the cancellation and preservation. More intuition about the construction of this polynomial is given later in Section 5.1.

We note that many previous algorithms use tensor decomposition as the method of moments gadget (e.g., (Sedghi et al., 2016; Zhong et al., 2016)) to learn the mixtures in one shot. Their algorithms, while being novel and inspiring, either require the data distribution for different components to be spherical Gaussian, or have high complexity to tolerate derivation from spherical Gaussian.

Gradient descent algorithm If we only use the method of moments, then we will need $\left(\frac{\sigma}{\varepsilon}\right)^{O(k)}$ sample to achieve error ε. The dependence on ε is not desired. To achieve the polylog dependence on the final error ε, we only use the method of moments to get a warm start, and then apply gradient descent beginning from the warm start.

This step is a "local" convergence step by using gradient descent to minimize the *concave* function

$$g(v) = \mathbb{E}[\log(|\langle w - v, x\rangle| + \zeta)].$$

Without ζ, the approach is similar to the classical Gravitational allocation (Holden et al., 2017). However, without it, when v is very close to one of the w_i's, $\log(|\langle w - v, x\rangle|)$ will be close to zero and becomes less *smooth*. Thus, we add ζ to ensure smoothness for the convergence of SGD. As we will show, even with a fairly large ζ, SGD will converge *with high probability*. Similar local convergence algorithms were also used in previous works (e.g., (Klusowski et al., 2017)). However, with our objective function, the proof is *significantly simpler*.

The proof is by lower bounding the correlation between the negative gradient and the difference of the current solution from the ground truth, and then applying standard optimization analysis to get the convergence. The correlation is (a variant) of inverse Gaussians and thus can be bounded; see Section 5.2 for more intuition.

5. Algorithm

In this section, we describe our algorithm in three subsections, describing the three parts as mentioned in the overview respectively.

5.1. Warm Start for Learning One of the Weights

Here we present our algorithm for obtaining a warm start for the weight for one of the components w_i, whose algorithmic ideas and analysis are at the core of this paper. This algorithm outputs a point a_T such that $\min\{\|a_T - w_i\|_2\}_{i=1}^k \le O(\sigma^2\varepsilon)$. The total sample complexity and running time of this algorithm are proportional to $\left(\frac{\sigma}{\varepsilon}\right)^{O(k^2)}$. Eventually, we will take $\varepsilon = \text{poly}\left(\frac{p_{\min}\Delta}{\sigma}\right)$ to enter the warm start for the gradient descent in the next subsection.

MOMENTDESCENT (Algorithm 1) describes the details. It begins with $a_0 = 0$ and iterates to improve it to a_T. In each iteration, it first uses a set of samples to compute two quantities: σ_t^2 which is an estimation of $\min\{\|\mathbf{\Sigma}_i^2(w_i - a_t)\|_2\}_{i=1}^k$, and $\mathbf{U}_t$ which is an estimation of the span of $\{\mathbf{\Sigma}_i^2(w_i - a_t)\}_{i=1}^k$. Then it picks a random vector v from the span of $\mathbf{U}_t$ and tests if moving a_t along v can decrease σ_t^2; this is repeated a few times to guarantee success with high probability.

MOMENTDESCENT uses two subroutines. ONEDMIXTURE (Algorithm 2) is adopted from existing work (Moitra and Valiant, 2010) and is used to compute σ_t^2, an estimation of $\min\{\|\mathbf{\Sigma}_i^2(w_i - a_t)\|_2\}_{i=1}^k$. So we focus on the other subroutine POWERW (Algorithm 3).

POWERW tries to identify the subspace spanned by $\{\mathbf{\Sigma}_i^2 w_i\}_{i=1}^k$, given labels α_ℓ from regression weights $\{w_i\}_{i=1}^k$.[2] As mentioned in the overview, the moments will contain both the signal $\mathbf{\Sigma}_i w_i w_i^\top \mathbf{\Sigma}_i$ and the noise $\mathbf{\Sigma}_i^2$. For example,

$$\mathbb{E}[\alpha^2 x x^\top] = \sum_{i=1}^k p_i \left(2\mathbf{\Sigma}_i w_i w_i^\top \mathbf{\Sigma}_i + \|\mathbf{\Sigma}_i w_i\|_2^2 \mathbf{\Sigma}_i^2\right).$$

The crucial piece here is to mix the moments with carefully designed coefficients $\{c_p\}_{p=0}^k$, so that $\mathbb{E}[\mathbf{M}] = \sum_{p=0}^k \frac{c_p}{(2p-1)!!}\mathbb{E}[\alpha^{2p} x x^\top]$ will mostly contain only the signal. Later, we will show that if we let c_p to be the coefficients of z^{2p} in some polynomial $f(z) = \prod_{p=1}^s (z^2 - z_p)$ with carefully chosen $z_1, \cdots, z_s$ that are closely related to $\{\|\mathbf{\Sigma}_i w_i\|_2^2\}_{i=1}^k$, then

$$\mathbb{E}[\mathbf{M}] = \sum_{i=1}^k p_i(\mathbf{X}_i + \mathbf{Y}_i)$$

where $\mathbf{X}_i$ is proportional to $\mathbf{\Sigma}_i^2 w_i w_i^\top \mathbf{\Sigma}_i^2 f'(\|\mathbf{\Sigma}_i w_i\|_2)$ and $\mathbf{Y}_i$ is proportional to $\mathbf{\Sigma}_i^2 f(\|\mathbf{\Sigma}_i w_i\|_2)$. Therefore, if $j = \arg\min_i \|\mathbf{\Sigma}_i w_i\|_2$, then we would like f to be small and $f'(\|\mathbf{\Sigma}_j w_j\|_2)$ to be large. Furthermore, we would like f' and f'' to be bounded to tolerate errors in estimating $\|\mathbf{\Sigma}_i w_i\|_2$'s.

2. When used in MOMENTDESCENT, it is given labels α_ℓ from regression weights $(w_i - a_t)$'s, so it will estimate the subspace spanned by $\{\mathbf{\Sigma}_i^2(w_i - a_t)\}_{i=1}^k$.

Algorithm 1 MOMENTDESCENT(k, δ, ε)

Input: Number of mixture components k, failure probability δ, and error ε.

Output: a_T which is close to some w_i up to error $O(\sigma^2\varepsilon)$ with probability $1-\delta$.

$a_0 \leftarrow 0$. Set $T \leftarrow \Theta(k\sigma \log \frac{\sigma}{\varepsilon})$ and $q \leftarrow \Theta\left(\log \frac{k\sigma}{\varepsilon\delta}\right)$.
for $t = 0, 1, \cdots, T-1$ **do**
  Sample $m = (\frac{\sigma}{p_{\min}\varepsilon})^{O(k^2)}$ many samples $\{(x_i, \alpha_i)\}_{i=1}^m$.
  For every $i \in [m]$, $\alpha_i \leftarrow \alpha_i - \langle x_i, a_t\rangle$.
  Let $\{\sigma_i^2\}_{i=1}^k \leftarrow$ ONEDMIXTURE$(\{\alpha_i\}_{i=1}^m, k, \varepsilon^2/(k\sigma)^2)$.
  Let $\sigma_t^2 \leftarrow \min\{\sigma_i^2\}_{i=1}^k$.
  $\mathbf{U}_t \leftarrow$ POWERW$(\{x_i\}_{i=1}^m, \{\alpha_i\}_{i=1}^m, k, \varepsilon)$
  for $j \in [q]$ **do**
    Pick a random $\gamma \in \mathbb{R}^k$ such that $\gamma \sim \mathcal{N}(0, \mathbf{I})$ and let $v = \frac{\mathbf{U}_t\gamma}{\|\mathbf{U}_t\gamma\|_2}$.
    Sample m many samples $\{(x_i, \alpha_i)\}_{i=1}^m$.
    For every $i \in [m]$, let $\alpha_i' \leftarrow \alpha_i - \langle x_i, a_t + \eta_t v\rangle$, where $\eta_t = \Theta\left(\frac{\sigma_t}{\sigma\sqrt{k}}\right)$.
    Let $\{(\sigma_i')^2\}_{i=1}^k \leftarrow$ ONEDMIXTURE$(\{\alpha_i'\}_{i=1}^m, k, \varepsilon^2/(k\sigma)^2)$,
    Let $(\sigma')^2 \leftarrow \min\{(\sigma_i')^2\}_{i=1}^k$
    if $(\sigma')^2 \le \left(1 - \frac{1}{150k\sigma}\right)\sigma_t^2$ **then**
      $a_{t+1} \leftarrow a_t + \eta_t v$.
      break;
    end if
  end for
end for

Algorithm 2 ONEDMIXTURE $(\{z_i\}_{i=1}^m, k, \varepsilon)$

Input: $\{z_i\}_{i=1}^m$ where each $z_i \in \mathbb{R}$ comes from a mixture of one dimension (mean zero) Gaussian distribution, number of mixture components k, and error ε.

Output: $\{\sigma_i^2\}_{i=1}^k$, the variance of each component up to additive error ε.

See the algorithm in (Moitra and Valiant, 2010). Their theorem implies that the output is up to additive error ε with $O\left(\frac{\sigma_{\max}}{p_{\min}\varepsilon}\right)^{O(k)}$ samples, where $\sigma_{\max}^2$ is the maximum variance of those mixtures and $p_{\min}$ is the minimal probability that one mixture occurs.)

The following lemma shows that such a polynomial can be efficiently constructed. Using this lemma, COEFF (Algorithm 4) constructs the coefficients c_p's which are used in POWERW.

Lemma 2 (Coefficients) *For every $k \ge 2$, every $\rho > 1$, every $r_1, \cdots, r_k \in [\frac{1}{\rho}, \rho]$, and every $\varepsilon > 0$, one can find in time $O(k \log k)$ an integer $0 < s \le k$ and centers $1/\rho \le z_1 \le \cdots \le z_s \le \rho$ such that for $f(x) = \prod_{p=1}^s (x^2 - z_p)$ the following holds.*

1. *For $r = \min\{r_i\}_{i=1}^k$ and every $i \in [k]$, $|f(\sqrt{r_i})| \le \varepsilon|\sqrt{r}f'(\sqrt{r})|$.*
2. *$|\sqrt{r}f'(\sqrt{r})| \ge \left(\frac{\varepsilon}{\rho}\right)^k$.*
3. *For all x with $x^2 \in [1/\rho, \rho]$, $|f'(x)| \le 2k\rho^k$ and $|f''(x)| \le 4k^2\rho^k$.*

Algorithm 3 POWERW($\{x_i\}_{i=1}^m, \{\alpha_i\}_{i=1}^m, k, \varepsilon$)

Input: $\{x_i\}_{i=1}^m$ where each $x_i \in \mathbb{R}^d$ comes from a mixture of Gaussian distributions, and α_i the label of x_i, number of mixture components k, and error ε

Output: $\mathbf{U} \in \mathbb{R}^{d\times k}$, ε close to the subspace spanned by $\mathbf{\Sigma}_1^2 w_1, \cdots, \mathbf{\Sigma}_k^2 w_k$

1: $\{\sigma_i^2\}_{i=1}^k \leftarrow$ ONEDMIXTURE($\{\alpha_i\}_{i=1}^m, k, \varepsilon^{(g)}$) for $\varepsilon^{(g)} = \left(\frac{\varepsilon}{\sigma}\right)^{4k}$.

2: $\{c_i\}_{i=0}^k \leftarrow$ COEFF($\{\sigma_i^2\}_{i=1}^k, \varepsilon^{(p)}$) for $\varepsilon^{(p)} = \varepsilon$.

3:

$$\mathbf{M} \leftarrow \frac{1}{m}\sum_{p=0}^{k} \frac{c_p}{(2p-1)!!} \sum_{i=1}^{m} \alpha_i^{2p} x_i x_i^\top. \tag{3}$$

4: $\mathbf{U} \leftarrow$ the top-k singular vectors of $\mathbf{M}$.

Algorithm 4 COEFF($\{r_i\}_{i=1}^k, \varepsilon$)

Input: $\{r_i\}_{i=1}^k$ where each $r_i \in \mathbb{R}$, and error ε.

Output: $\{c_i\}_{i=0}^k$ where each $c_i \in \mathbb{R}$.

1: Let $z_1, \cdots, z_s$ be a center of $r_1, \cdots, r_k$ defined by Lemma 2.

2: Let c_i be the coefficient of x^{2i} in the polynomial:

$$f(x) = \prod_{p=1}^{s} (x^2 - z_p). \tag{4}$$

Putting things together, we can prove the main lemma regarding the per-iteration improvement of Algorithm 1.

Lemma 3 *For every $t \in \{0, 1, \cdots, T-1\}$ and $\delta > 0$, as long as $\sigma_t = \Omega(\sigma\varepsilon)$, then with probability at least $1 - \delta$,*

$$\sigma_{t+1}^2 \le \left(1 - \frac{1}{200k\sigma}\right)\sigma_t^2.$$

Using this Lemma and by the choice of our parameters we immediately have the following guarantee for the output of Algorithm 1.

Lemma 4 *With probability at least $1 - \delta$, $\min_i \|w_i - a_T\|_2 \le O(\sigma^2\varepsilon)$.*

5.2. Learning One of the Weights from Warm Start

Here we describe how to use gradient descent on a concave function for faster convergence to one of the w_i's, given the warm start computed by the algorithm in the last subsection.

Algorithm 5 describes the details. The gradient descent is to minimize the function

$$g(v) = \mathbb{E}[\log(|\langle w - v, x\rangle| + \zeta)]$$

Algorithm 5 GRADIENTDESCENT(k, v, ε)

Input: k the number of clusters, a warm start v, and the final error ε.

Output: $v^{(T)}$, recovered weight parameter up to additive error ε.

1: Let $v^{(0)} \leftarrow v$, $T \leftarrow \Theta\left(\frac{d}{p_{\min}^2}\log\frac{\zeta}{\varepsilon}\right)$, where $\zeta = \min\left\{\frac{\Delta}{2\sigma}, \frac{\Delta p_{\min}}{64}\right\}$.

2: **for** $t = 0, 1, \cdots, T-1$ **do**

3: Sample $m = \text{poly}\left(\frac{1}{\Delta}, \frac{1}{p_{\min}}, \sigma, \log T\right)$ many samples $\mathcal{S}_{t+1} = \{x_i, \alpha_i\}_{i=1}^m$.

4: Update: For properly chosen learning rate $\eta_t = \Theta\left(\frac{\zeta p_{\min}}{d}\right) \times \left(1 - \Theta\left(\frac{p_{\min}^2}{d}\right)\right)^t$

$$v^{(t+1)} = v^{(t)} + \eta_t \frac{1}{|\mathcal{S}_{t+1}|} \sum_{(x,\alpha)\in\mathcal{S}_{t+1}} \frac{\text{sign}(\alpha - \langle v^{(t)}, x\rangle)}{|\alpha - \langle v^{(t)}, x\rangle| + \zeta} x. \tag{5}$$

5: **end for**

Algorithm 6 Learning Mixtures of Linear Regressions

Input: Dataset $\mathcal{D} = \{(x_\ell, \alpha_\ell)\}_{\ell=1}^N$, number of components k, error ε. (Parameters $\sigma, \Delta, p_{\min}$ are known to all the algorithms)

Output: $\{v_i\}_{i=1}^k$, recovered weight parameters up to additive error ε.

1: **for** $i = 1, \ldots, k$ **do**

2: $a \leftarrow$ MOMENTDESCENT$(k-i+1, \delta, \varepsilon_w)$, where $\varepsilon_w = \text{poly}\left(\frac{p_{\min}\Delta}{\sigma}\right)$ and $\delta = \text{poly}\left(\frac{1}{d}\right)$.

3: $v_i \leftarrow$ GRADIENTDESCENT$(k-i+1, a, \varepsilon_g)$, where $\varepsilon_g = \min\left\{\varepsilon, \left(\frac{p_{\min}\Delta}{\sigma d}\right)^{\Omega(k^2)}\right\}$.

4: Remove from $\mathcal{D}$ all the data (x_ℓ, α_ℓ) such that $|\langle x_\ell, v_i\rangle - \alpha_\ell| \le \varepsilon_g \sigma \cdot \text{polylog}(d)$.

5: **end for**

where ζ is added to make the $\log(\cdot)$ smooth. The key property used is that we have a large correlation between the negative gradient and the difference of the current solution from the ground truth. Suppose we begin with a warm start close enough to w_1, then the correlation is $\mathbb{E}\left[\frac{\text{sign}(\alpha - \langle v^{(t)}, x\rangle)\langle w_1 - v^{(t)}, x\rangle}{|\alpha - \langle v^{(t)}, x\rangle| + \zeta}\right]$. This is (a variant of) inverse Gaussians and can be bounded by a function of the norms $\|w_i - v^{(t)}\|_2$ for $i \in [k]$. Since $\|w_1 - v^{(t)}\|_2$ is much smaller than the other norms $\|w_i - v^{(t)}\|_2$ for $i \neq 1$, the correlation can be shown to be large. The convergence then follows from standard analysis.

Lemma 5 (Gradient descent) *Suppose there exists $i \in [k]$ such that $\|w_i - v\|_2 \le \zeta/\sigma$. Then with high probability, Algorithm 5 outputs a vector $v^{(T)}$ such that $\|w_i - v^{(T)}\| \le \varepsilon$.*

5.3. Learning All the Weights

Here we describe our final algorithm for learning all the weights. It uses the algorithm in the previous subsections to learn the weight of one of the components, removes the data points from that component, and repeats. Note that we can learn the weight up to error ε_g in time $\log(1/\varepsilon_g)$, so ε_g can be made as small as $\left(\frac{p_{\min}\Delta}{\sigma d}\right)^{\Omega(k^2)}$ so that the step of removing the data points introduces essentially no error to later steps within our sample size. So we arrive at our final guarantee in Theorem 1.

6. Conclusion

In this paper, we present a fixed parameter algorithm that solves mixture of linear regression under Gaussian inputs in time nearly linear in the sample size and the dimension. Moreover, our sample complexity also scales nearly linear with the dimension d. In our setting, we allow each mixture to have a different covariance matrix. Thus, unlike the case when the mixtures are spherical, even the best known algorithm for mixture of general Gaussians would require at least d^2 sample complexity to recover the covariance. Our algorithm reduces the sample complexity significantly with the additional one dimensional linear information: it can recover the linear classifier (and thus recover the covariance as well) with $\tilde{O}(d)$ samples. While the dependency on d is nearly optimal, we would also like to point out that when the total number of mixtures are too large, the sample complexity of our algorithm does suffer from an exponential term of k. We believe that with our current set of assumptions, the exponential dependency could be necessary: A lower bound of e^k has been proved in (Moitra and Valiant, 2010) in the very similar setting of learning mixture of Gaussians.

One natural way to get around the exponential dependency is assuming that the covariance $\mathbf{\Sigma}_i$ and the hidden vectors w_i satisfies some smoothness assumption (e.g., (Ge et al., 2015)). However, the level of smoothness is very subtle in our setting, since the naïve application of smoothed analysis often leads to complexity with a large polynomial factor in the dimension. In this paper, near linearity in d is one of our main contributions. We believe that using smoothed analysis while preserving the nearly linear dependency on d is one of the important future directions.

Acknowledgments

Yingyu Liang would like to acknowledge that support for this research was provided by the Office of the Vice Chancellor for Research and Graduate Education at the University of Wisconsin Madison with funding from the Wisconsin Alumni Research Foundation.

References

Zeyuan Allen-Zhu and Yuanzhi Li. Lazysvd: Even faster svd decomposition yet without agonizing pain. In *Advances in Neural Information Processing Systems*, pages 974–982, 2016.

Hassan Ashtiani, Shai Ben-David, and Abbas Mehrabian. Sample-efficient learning of mixtures. *arXiv preprint arXiv:1706.01596*, 2017.

Sivaraman Balakrishnan, Martin J Wainwright, Bin Yu, et al. Statistical guarantees for the em algorithm: From population to sample-based analysis. *The Annals of Statistics*, 45(1):77–120, 2017.

Arun T Chaganty and Percy Liang. Spectral experts for estimating mixtures of linear regressions. In *Proceedings of the 30th International Conference on Machine Learning (ICML-13)*, pages 1040–1048, 2013.

Yudong Chen, Xinyang Yi, and Constantine Caramanis. A convex formulation for mixed regression with two components: Minimax optimal rates. In *Conference on Learning Theory*, pages 560–604, 2014.

Richard D De Veaux. Mixtures of linear regressions. *Computational Statistics & Data Analysis*, 8 (3):227–245, 1989.

Susana Faria and Gilda Soromenho. Fitting mixtures of linear regressions. *Journal of Statistical Computation and Simulation*, 80(2):201–225, 2010.

Scott Gaffney and Padhraic Smyth. Trajectory clustering with mixtures of regression models. In *Proceedings of the fifth ACM SIGKDD international conference on Knowledge discovery and data mining*, pages 63–72. ACM, 1999.

Rong Ge, Qingqing Huang, and Sham M Kakade. Learning mixtures of gaussians in high dimensions. In *Proceedings of the forty-seventh annual ACM symposium on Theory of computing*, pages 761–770. ACM, 2015.

Bettina Grün, Friedrich Leisch, et al. Applications of finite mixtures of regression models. 2007.

Nina Holden, Yuval Peres, and Alex Zhai. Gravitational allocation for uniform points on the sphere. *arXiv preprint arXiv:1704.08238*, 2017.

Michael I Jordan and Robert A Jacobs. Hierarchical mixtures of experts and the em algorithm. *Neural computation*, 6(2):181–214, 1994.

Abbas Khalili and Jiahua Chen. Variable selection in finite mixture of regression models. *Journal of the american Statistical association*, 102(479):1025–1038, 2007.

Jason M Klusowski, Dana Yang, and WD Brinda. Estimating the coefficients of a mixture of two linear regressions by expectation maximization. *arXiv preprint arXiv:1704.08231*, 2017.

Yuanzhi Li and Yang Yuan. Convergence analysis of two-layer neural networks with relu activation. In *Advances in Neural Information Processing Systems*, pages 597–607, 2017.

Ankur Moitra and Gregory Valiant. Settling the polynomial learnability of mixtures of gaussians. In *Foundations of Computer Science (FOCS), 2010 51st Annual IEEE Symposium on*, pages 93–102. IEEE, 2010.

Hanie Sedghi, Majid Janzamin, and Anima Anandkumar. Provable tensor methods for learning mixtures of generalized linear models. In *Artificial Intelligence and Statistics*, pages 1223–1231, 2016.

Xinyang Yi, Constantine Caramanis, and Sujay Sanghavi. Alternating minimization for mixed linear regression. In *International Conference on Machine Learning*, pages 613–621, 2014.

Xinyang Yi, Constantine Caramanis, and Sujay Sanghavi. Solving a mixture of many random linear equations by tensor decomposition and alternating minimization. *arXiv preprint arXiv:1608.05749*, 2016.

Kai Zhong, Prateek Jain, and Inderjit S Dhillon. Mixed linear regression with multiple components. In *Advances in Neural Information Processing Systems*, pages 2190–2198, 2016.

Appendix A. Proof of Warm Start for Learning One of the Weights

We prove the following lemma related to the output of Algorithm 1.

Lemma 4 *With probability at least* $1-\delta$, $\min_i \|w_i - a_T\|_2 \le O(\sigma^2\varepsilon)$.

Before proving this lemma, we first need the following lemma about the clustering, which is crucial for constructing the coefficients. As we shall see, we will use this lemma on $r_i = \|\mathbf{\Sigma}_i(w_i - a_t)\|_2^2$. Roughly speaking, $f(\sqrt{r_i})$ is the weight of $\mathbf{\Sigma}_i^2$ and $f'(\sqrt{r_i})$ is the weight of $\mathbf{\Sigma}_i^2(w_i - a_t)$. Therefore, we would like $f(\sqrt{r_i})$ to be small compare to $f'(\sqrt{r_i})$ to identify the subspace spanned by $\mathbf{\Sigma}_i^2(w_i - a_t)$.

Lemma 2 (Coefficients) *For every $k \ge 2$, every $\rho > 1$, every $r_1, \cdots, r_k \in [\frac{1}{\rho}, \rho]$, and every $\varepsilon > 0$, one can find in time $O(k \log k)$ an integer $0 < s \le k$ and centers $1/\rho \le z_1 \le \cdots \le z_s \le \rho$ such that for $f(x) = \prod_{p=1}^{s}(x^2 - z_p)$ the following holds.*

1. *For $r = \min\{r_i\}_{i=1}^k$ and every $i \in [k]$, $|f(\sqrt{r_i})| \le \varepsilon|\sqrt{r}f'(\sqrt{r})|$.*
2. *$|\sqrt{r}f'(\sqrt{r})| \ge \left(\frac{\varepsilon}{\rho}\right)^k$.*
3. *For all x with $x^2 \in [1/\rho, \rho]$, $|f'(x)| \le 2k\rho^k$ and $|f''(x)| \le 4k^2\rho^k$.*

Proof [Proof of Lemma 2] Let us without loss of generality assume that $r = r_1 \le r_2 \le \cdots \le r_k$. Let us define $z_1 = r_1$, and let $j \in [k]$ be the smallest index such that $r_j \ge z_1 + \frac{\varepsilon}{\rho}$. If no such index exists, we let $s = 1$ and the statements in the lemma are true. If such j exists, let us define:

$$z_2 = r_j, z_3 = r_{j+1}, \cdots, z_s = r_k. \tag{6}$$

Now, we know that

$$|\sqrt{r}f'(\sqrt{r})| = 2r\prod_{p=2}^{s}|r - z_p| \ge \left(\frac{\varepsilon}{\rho}\right)^k. \tag{7}$$

On the other hand, for every $i \ge j$, $f(\sqrt{r_i}) = 0$. For $i < j$ we have:

$$|f(\sqrt{r_i})| = |r_i - r|\prod_{p=2}^{s}|r_i - z_p| \tag{8}$$

$$\le \frac{\varepsilon}{\rho}\prod_{p=2}^{s}|r_i - z_p| \le \varepsilon r\prod_{p=2}^{s}|r - z_p| \le \varepsilon|\sqrt{r}f'(\sqrt{r})|. \tag{9}$$

We now consider the derivative and second order derivative of $f(x)$ for $x^2 \in [0, \rho]$. By elementary calculation, we know that

$$|f'(x)| = \left|\sum_{p=1}^{s} 2x\prod_{q\ne p}(x^2 - z_q)\right| \tag{10}$$

$$\le 2\sum_{p=1}^{s}|x|\prod_{q\ne p}|x^2 - z_q| \tag{11}$$

$$\le 2k\rho^k. \tag{12}$$

Similarly we can get that $|f''(x)| \le 4k^2\rho^k$. ■

We also need the following bound for the k-SVD of a matrix.

Lemma 6 *Let $\mathbf{X}_1, \cdots, \mathbf{X}_k$ be k rank-one matrices in $\mathbb{R}^{d\times d}$ such that each $\mathbf{X}_i = x_i x_i^\top$, for every $\varepsilon \ge 0$, every PSD matrix $\mathbf{M} \in \mathbb{R}^{d\times d}$ such that*

$$\left\| \mathbf{M} - \sum_{i=1}^{k} \mathbf{X}_i \right\|_2 \le \varepsilon \|\mathbf{X}_1\|_2 \tag{13}$$

Let $\mathbf{U} \in \mathbb{R}^{d\times k}$ be the matrix consists of the top-k singular vectors of $\mathbf{M}$, then we have

$$\|x_1^\top \mathbf{U}\|_2 \ge \left(1 - (\varepsilon k)^{1/3}\right) \|x_1\|_2 \tag{14}$$

Proof [Proof of Lemma 6] Let us denote $\sigma_1 \ge \cdots \ge \sigma_k \ge \sigma_{k+1} = 0$ as the $k+1$ singular values of $\sum_{i=1}^k \mathbf{X}_i$ with corresponding singular vectors $v_1, \cdots, v_k$ (and v_{k+1}). For every v_i, by definition

$$v_i^\top \left(\sum_{j=1}^{k} \mathbf{X}_j \right) v_i = \sigma_i \tag{15}$$

So we have $v_i^\top \mathbf{X}_1 v_i \le \sigma_i$. Let $\mathbf{V}_i \in \mathbb{R}^{d\times i}$ defined as $\mathbf{V}_i = (v_1, \cdots, v_i)$. By Gap-free Wedin theorem in (Allen-Zhu and Li, 2016) (see Lemma 11), we know that

$$\|(\mathbf{I} - \mathbf{U}\mathbf{U}^\top)\mathbf{V}_i\|_2 \le \frac{\varepsilon \|x_1\|_2^2}{\sigma_i}. \tag{16}$$

Thus, $\|x_1^\top (\mathbf{V}_i \mathbf{V}_i^\top)(\mathbf{I} - \mathbf{U}\mathbf{U}^\top)\|_2 \le \frac{\varepsilon \|x_1\|_2^3}{\sigma_i}$.

On the other hand, since $x_1 \in \text{span}\{v_1, \cdots, v_k\}$,

$$\|x_1^\top (\mathbf{I} - \mathbf{V}_i \mathbf{V}_i^\top)\|_2 = \|x_1^\top (\mathbf{V}_k \mathbf{V}_k^\top - \mathbf{V}_i \mathbf{V}_i^\top)\|_2 \tag{17}$$

$$\le \sum_{j=i+1}^{k} |x_i^\top v_k| \le k\sqrt{\sigma_{i+1}}. \tag{18}$$

Therefore, we know that

$$\|x_1^\top (\mathbf{I} - \mathbf{U}\mathbf{U}^\top)\|_2 \le \frac{\varepsilon \|x_1\|_2^3}{\sigma_i} + k\sqrt{\sigma_{i+1}}. \tag{19}$$

If $\sigma_1 \ge \frac{\|x_1\|_2^2 \varepsilon^{2/3}}{k^{2/3}}$, by picking i to the largest index in $[k]$ such that $\sigma_i \ge \frac{\|x_1\|_2^2 \varepsilon^{2/3}}{k^{2/3}}$, we get that

$$\|x_1^\top (\mathbf{I} - \mathbf{U}\mathbf{U}^\top)\|_2 \le (\varepsilon k)^{1/3} \|x_1\|_2 \tag{20}$$

If $\sigma_1 \le \frac{\|x_1\|_2^2 \varepsilon^{2/3}}{k^{2/3}}$, then we can just use $\|x_1^\top\|_2 \le k\sqrt{\sigma_1}$ to complete the proof. ■

We are now ready to prove the following important lemma about the correlation between $\mathbf{U}$ and $\mathbf{\Sigma}_i^2(w_i - a_t)$.

Lemma 7 *Let $j = \arg\min_{1\le i\le k} \|\mathbf{\Sigma}_i(w_i - a_t)\|_2$, we have that in the t-th iteration of Algorithm 1, the $\mathbf{U}_t$ satisfies*

$$\frac{\|\mathbf{U}_t^\top \mathbf{\Sigma}_j^2(w_j - a_t)\|_2}{\|\mathbf{\Sigma}_j^2(w_j - a_t)\|_2} \ge \frac{1}{2}. \tag{21}$$

Proof [Proof of Lemma 7] Suppose $z \sim \mathcal{N}(0, \mathbf{\Sigma}^2)$, we know that $z = \mathbf{\Sigma} g$ where $g \sim \mathcal{N}(0, \mathbf{I})$. For every vector a,

$$\mathbb{E}\left[\langle z, a\rangle^{2p} z z^\top\right] = \mathbf{\Sigma}\mathbb{E}\left[\langle g, \mathbf{\Sigma} a\rangle^{2p} g g^\top\right]\mathbf{\Sigma} \tag{22}$$

$$= (2p-1)!!\mathbf{\Sigma}\left(2p\mathbf{\Sigma} a a^\top \mathbf{\Sigma}\|\mathbf{\Sigma} a\|_2^{2p-2} + \|\mathbf{\Sigma} a\|_2^{2p}\mathbf{I}\right)\mathbf{\Sigma} \tag{23}$$

$$= (2p-1)!!\|\mathbf{\Sigma} a\|_2^{2p}\left(2p\frac{\mathbf{\Sigma}^2 a a^\top \mathbf{\Sigma}^2}{\|\mathbf{\Sigma} a\|_2^2} + \mathbf{\Sigma}^2\right). \tag{24}$$

Thus, we have

$$\frac{1}{(2p-1)!!}\mathbb{E}\left[\alpha_i^{2p} x_i x_i^\top\right] = \sum_{i=1}^k p_i \|\mathbf{\Sigma}_i(w_i - a_t)\|_2^{2p}\left(2p\frac{\mathbf{\Sigma}_i^2(w_i - a_t)(w_i - a_t)^\top \mathbf{\Sigma}_i^2}{\|\mathbf{\Sigma}_i(w_i - a_t)\|_2^2} + \mathbf{\Sigma}_i^2\right). \tag{25}$$

Since in the t-th iteration, the labels α_i we fit to Algorithm 3 comes from $\alpha_\ell = \langle x_\ell, w^{(\ell)} - a_t\rangle$, we know that

$$\mathbb{E}[\mathbf{M}] = \sum_{i=1}^k p_i \sum_{p=0}^k \left(c_p \|\mathbf{\Sigma}_i(w_i - a_t)\|_2^{2p}\left(2p\frac{\mathbf{\Sigma}_i^2(w_i - a_t)(w_i - a_t)^\top \mathbf{\Sigma}_i^2}{\|\mathbf{\Sigma}_i(w_i - a_t)\|_2^2} + \mathbf{\Sigma}_i^2\right)\right). \tag{26}$$

Let us define the signal matrix $\mathbf{X}_i$ as

$$\mathbf{X}_i = \frac{\mathbf{\Sigma}_i^2(w_i - a_t)(w_i - a_t)^\top \mathbf{\Sigma}_i^2}{\|\mathbf{\Sigma}_i(w_i - a_t)\|_2^2}\left(\sum_{p=0}^k 2pc_p\|\mathbf{\Sigma}_i(w_i - a_t)\|_2^{2p}\right) \tag{27}$$

$$= \frac{\mathbf{\Sigma}_i^2(w_i - a_t)(w_i - a_t)^\top \mathbf{\Sigma}_i^2}{\|\mathbf{\Sigma}_i(w_i - a_t)\|_2^2}\left(f'(\|\mathbf{\Sigma}_i(w_i - a_t)\|_2)\|\mathbf{\Sigma}_i(w_i - a_t)\|_2\right) \tag{28}$$

and the noise matrix $\mathbf{Y}_i$ as

$$\mathbf{Y}_i = \mathbf{\Sigma}_i^2\left(\sum_{p=0}^k c_p\|\mathbf{\Sigma}_i(w_i - a_t)\|_2^{2p}\right) \tag{29}$$

$$= \mathbf{\Sigma}_i^2 f(\|\mathbf{\Sigma}_i(w_i - a_t)\|_2) \tag{30}$$

such that

$$\mathbb{E}[\mathbf{M}] = \sum_{i=1}^k p_i(\mathbf{X}_i + \mathbf{Y}_i). \tag{31}$$

For $j = \arg\min\{\|\mathbf{\Sigma}_i(w_i - a_t)\|_2)\}_{i=1}^k$, let us denote

$$\beta := f'(\|\mathbf{\Sigma}_j(w_j - a_t)\|_2)\|\mathbf{\Sigma}_j(w_j - a_t)\|_2.$$

Let us recall that $\varepsilon^{(g)}$ is the error incurred when estimating $\{\|\mathbf{\Sigma}_i(w_i - a_t)\|_2\}_{i=1}^k$. $\varepsilon^{(p)}$ is the error when constructing the coefficients of the polynomial (for sufficiently large ρ such that $\rho \geq \max\{\|\mathbf{\Sigma}_i(w_i - a_t)\|_2^2)\}_{i=1}^k$ as we will show later in this proof). Thus, by Lemma 2, we know that

$$\begin{aligned}
\|\mathbf{Y}_i\|_2 &\leq \|\mathbf{\Sigma}_i^2\|_2|f(\|\mathbf{\Sigma}_i(w_i - a_t)\|_2)| && (32)\\
&\leq \|\mathbf{\Sigma}_i^2\|_2(|f(\sigma_i)| + 2k\rho^k\,|\sigma_i - \|\mathbf{\Sigma}_i(w_i - a_t)\|_2|) && (33)\\
&\leq \|\mathbf{\Sigma}_i^2\|_2(\varepsilon^{(p)}\beta + 4k\rho^k\varepsilon^{(g)}). && (34)
\end{aligned}$$

Similarly we have

$$\|\mathbf{X}_j\|_2 \geq \sigma_{\min}(\mathbf{\Sigma}_j^2)\beta. \tag{35}$$

And we have $\beta \geq \left(\frac{\varepsilon^{(p)}}{\rho}\right)^k - 8k^2\rho^k\varepsilon^{(g)}\sigma^2$.

Notice that $\min\{\|\mathbf{\Sigma}_i(w_i - a_t)\|_2)\}_{i=1}^k \leq \min\{\|\mathbf{\Sigma}_i(w_i)\|_2)\}_{i=1}^k$, which implies that $\|a_1\|_2 \leq \sigma^4$. Therefore, we can take $\rho = O\left(\max\left\{2\sigma^{10}, \frac{1}{\varepsilon}\right\}\right)$. Thus, by our choice of parameter, we know that for $\varepsilon^{(e)} \leq \frac{1}{100k}$,

$$\left\|\mathbb{E}[\mathbf{M}] - \sum_{i=1}^k p_i\mathbf{X}_i\right\|_2 \leq \varepsilon^{(e)}\|\mathbf{X}_j\|_2/2. \tag{36}$$

Using the sample complexity bound Lemma 9, by our choice of m we know that

$$\|\mathbf{M} - \mathbb{E}[\mathbf{M}]\|_2 \leq \varepsilon^{(e)}\|\mathbf{X}_j\|_2/2. \tag{37}$$

Thus, apply Lemma 6 on $\mathbf{M}$ we know that

$$\frac{\|\mathbf{U}_t^\top\mathbf{X}_j\mathbf{U}_t\|_2}{\|\mathbf{X}_j\|_2} \geq 1 - \left(\varepsilon^{(e)}k\right)^{1/3} \geq \frac{3}{4}. \tag{38}$$

Indeed, this also implies that

$$\frac{\|\mathbf{U}_t^\top\mathbf{\Sigma}_j^2(w_j - a_t)\|_2}{\|\mathbf{\Sigma}_j^2(w_j - a_t)\|_2} \geq \frac{1}{2} \tag{39}$$

completing the proof. ■

Now we can prove the main lemma regarding the per-iteration improvement of Algorithm 1.

Lemma 3 (Coefficients) *For every $t \in \{0, 1, \cdots, T-1\}$ and $\delta > 0$, as long as $\sigma_t = \Omega(\sigma\varepsilon)$, then with probability at least $1 - \delta$,*

$$\sigma_{t+1}^2 \leq \left(1 - \frac{1}{200k\sigma}\right)\sigma_t^2.$$

Proof [Proof of Lemma 3] At t-th iteration let $j = \arg\min\{\|\mathbf{\Sigma}_i(w_i - a_t)\|_2\}_{i=1}^k$, we know that

$$\frac{\|\mathbf{U}_t^\top \mathbf{\Sigma}_j^2(w_j - a_t)\|_2}{\|\mathbf{\Sigma}_j^2(w_j - a_t)\|_2} \geq \frac{1}{2}. \tag{40}$$

By definition, $v = \frac{\mathbf{U}_t\gamma}{\|\mathbf{U}_t\gamma\|_2}$ for $\gamma \in \mathcal{N}(0, \mathbf{I})$. Thus, using elementary calculation of Gaussian random variables, we have: with probability at least $1/4$,

$$\frac{v^\top \mathbf{\Sigma}_j^2(w_j - a_t)}{\|\mathbf{\Sigma}_j^2(w_j - a_t)\|_2} \geq \frac{1}{10\sqrt{k}} \tag{41}$$

which implies that

$$\begin{aligned}
\|\mathbf{\Sigma}_j(w_j - a_t - \eta v)\|_2^2 &= \|\mathbf{\Sigma}_j(w_j - a_t)\|_2^2 - 2\eta\langle \mathbf{\Sigma}_j(w_j - a_t), \mathbf{\Sigma}_j v\rangle + \eta^2\|\mathbf{\Sigma}_j v\|_2^2 && (42)\\
&= \|\mathbf{\Sigma}_j(w_j - a_t)\|_2^2 - 2\eta\langle \mathbf{\Sigma}_j^2(w_j - a_t), v\rangle + \eta^2\|\mathbf{\Sigma}_j v\|_2^2 && (43)\\
&\leq \|\mathbf{\Sigma}_j(w_j - a_t)\|_2^2 - \frac{\eta}{5\sqrt{k}}\|\mathbf{\Sigma}_j^2(w_j - a_t)\|_2 + \eta^2\sigma. && (44)
\end{aligned}$$

Let $\eta = \frac{\|\mathbf{\Sigma}_j^2(w_j - a_t)\|_2}{10\sigma\sqrt{k}}$. Then we know that

$$\|\mathbf{\Sigma}_j(w_j - a_t - \eta v)\|_2^2 \leq \left(1 - \frac{1}{100k\sigma}\right)\|\mathbf{\Sigma}_j(w_j - a_t)\|_2^2.$$

Thus, since we can estimate $\|\mathbf{\Sigma}_j(w_j - a_t - \eta v)\|_2$ up to accuracy $\varepsilon/(k\sigma)$ using the algorithm proposed in (Moitra and Valiant, 2010), as long as $\sigma_t = \Omega(\sigma\varepsilon)$, we will have that $\sigma_{t+1}^2 \leq \left(1 - \frac{1}{200k\sigma}\right)\sigma_t^2$. ■

This immediately leads to the main lemma regarding the output of Algorithm 1.

Proof [Proof of Lemma 4] By Lemma 3, and by the choice of the parameters in the algorithm, $\sigma_T \leq O(\sigma\varepsilon)$. Then for $j = \min_i\{\|\Sigma_i(w_i - a_T)\|_2\}$ we have $\|\Sigma_j(w_j - a_T)\|_2 \leq O(\sigma\varepsilon)$ and thus $\|w_j - a_T\|_2 \leq O(\sigma^2\varepsilon)$. ■

Appendix B. Proof for Learning One of the Weights from Warm Start

Without loss of generality, let us assume that we have an v such that $\|v - w_1\|_2$ is reasonably small. We will show that the update rule used in the algorithm can recover w_1 up to error ε with this v. It is equivalent to (the empirical version of) the gradient descent update to minimize the following concave objective function:

$$g(v) = \mathbb{E}\left[\log(|\alpha - \langle v, x\rangle| + \zeta)\right].$$

Lemma 5 (Gradient descent) *Suppose there exists $i \in [k]$ such that $\|w_i - v\|_2 \leq \zeta/\sigma$. Then with high probability, Algorithm 5 outputs a vector $v^{(T)}$ such that $\|w_i - v^{(T)}\| \leq \varepsilon$.*

Proof [Proof of Lemma 5] First, suppose we have the gradient on the expectation, i.e., we have $\nabla g(v^{(t)})$. For this gradient descent update rule, by Lemma 10, we know that

$$\begin{aligned}\left\langle -\nabla g(v^{(t)}), w_1 - v^{(t)} \right\rangle &= \mathbb{E}\left[\frac{\operatorname{sign}(\alpha - \langle v^{(t)}, x\rangle)\langle w_1 - v^{(t)}, x\rangle}{|\alpha - \langle v^{(t)}, x\rangle| + \zeta}\right] \\ &= p_1 \mathbb{E}_{y\sim\mathcal{N}(0,1)}\mathbb{E}\left[\frac{\operatorname{sign}(\langle \mathbf{\Sigma}_1(w_1 - v^{(t)}), y\rangle)\langle \mathbf{\Sigma}_1(w_1 - v^{(t)}), y\rangle}{|\langle \mathbf{\Sigma}_1(w_1 - v^{(t)}), y\rangle| + \zeta}\right] \\ &\quad + \sum_{j=2}^{k} p_j \mathbb{E}_{y\sim\mathcal{N}(0,1)}\mathbb{E}\left[\frac{\operatorname{sign}(\langle \mathbf{\Sigma}_j(w_j - v^{(t)}), y\rangle)\langle \mathbf{\Sigma}_j(w_1 - v^{(t)}), y\rangle}{|\langle \mathbf{\Sigma}_j(w_j - v^{(t)}), y\rangle| + \zeta}\right] \\ &\geq \frac{1}{4} p_1 \frac{\|\mathbf{\Sigma}_1(w_1 - v^{(t)})\|_2}{\|\mathbf{\Sigma}_1(w_1 - v^{(t)})\|_2 + \zeta} - \sum_{j=2}^{k} p_j \frac{\|\mathbf{\Sigma}_1(w_1 - v^{(t)})\|_2}{\|\mathbf{\Sigma}_j(w_j - v^{(t)})\|_2}.\end{aligned}$$

Note that our assumption on ζ satisfies that

$$\|\mathbf{\Sigma}_1(w_1 - v^{(t)})\|_2 \leq \zeta, \quad \|\mathbf{\Sigma}_j(w_j - v^{(t)})\|_2 \geq 32\zeta/p_{\min}, j \neq 1, \tag{45}$$

Therefore, a direct calculation shows that

$$\left\langle -\nabla g(v^{(t)}), w_1 - v^{(t)} \right\rangle \geq \frac{p_{\min}}{32} \frac{\|\mathbf{\Sigma}_1(w_1 - v^{(t)})\|_2}{\zeta} \geq \frac{p_{\min}\|w_1 - v^{(t)}\|_2}{32\zeta}.$$

However, we only have the empirical version of the gradient given as

$$-\tilde{\nabla} g(v^{(t)}) = \mathbb{E}_{(x_\ell, \alpha_\ell)} \nabla g_\ell(v), \text{ where } -\nabla g_\ell(v^{(t)}) = \frac{\operatorname{sign}(\alpha_\ell - \langle v^{(t)}, x_\ell\rangle)}{|\alpha_\ell - \langle v^{(t)}, x_\ell\rangle| + \zeta} x_\ell.$$

To apply concentration bound on the empirical version, we know that for for every example (x, α),

$$\left\| \frac{\operatorname{sign}(\alpha - \langle v^{(t)}, x\rangle)}{|\alpha - \langle v^{(t)}, x\rangle| + \zeta} x \right\|_2 \leq \frac{\|x\|_2}{\zeta}.$$

Moreover, we know that the true gradient satisfies

$$\left\langle -\nabla g(v^{(t)}), \frac{w_1 - v^{(t)}}{\|w_1 - v^{(t)}\|_2} \right\rangle \geq \frac{p_{\min}}{32\zeta}$$

For every example (x, α), we have

$$\left| \left\langle \frac{\operatorname{sign}(\alpha - \langle v^{(t)}, x\rangle)x}{|\alpha - \langle v^{(t)}, x\rangle| + \zeta}, \frac{w_1 - v^{(t)}}{\|w_1 - v^{(t)}\|_2} \right\rangle \right| \leq \frac{\left| \left\langle \frac{w_1 - v^{(t)}}{\|w_1 - v^{(t)}\|_2}, x \right\rangle \right|}{\zeta}.$$

Using an elementary concentration bound of Gaussian random variables, we know that with $\operatorname{poly}\left(\frac{1}{\zeta}, \frac{1}{p_{\min}}, \sigma\right)$ examples, the estimated gradient $\tilde{\nabla} g(v^{(t)})$ satisfies with high probability that

$$\|\tilde{\nabla} g(v^{(t)})\|_2 \leq \frac{4\sqrt{d}}{\zeta}, \quad \left\langle -\tilde{\nabla} g(v^{(t)}), \frac{w_1 - v^{(t)}}{\|w_1 - v^{(t)}\|_2} \right\rangle \geq \frac{p_{\min}}{64\zeta}.$$

Then when $\eta_t = c\frac{\zeta p_{\min}\|w_1 - v^{(t)}\|_2}{d}$ for a sufficiently small constant $c > 0$, and using the assumptions on $v^{(0)}$ and Δ to satisfy the condition (45), by induction, we have

$$\|w_1 - v^{(t+1)}\|_2^2 \le \left(1 - \Omega\left(\frac{p_{\min}^2}{d}\right)\right)\|w_1 - v^{(t)}\|_2^2$$

completing the proof. ■

Appendix C. Proof for Learning All the weights

Theorem 1 (Main) *Assume the model (1) and assumptions **(A1)-(A3)**. Then Algorithm 6 takes* $N = d\log\left(\frac{d}{\varepsilon}\right)\cdot\left(\frac{\sigma}{\Delta p_{\min}}\right)^{O(k)} + \left(\frac{\sigma}{\Delta p_{\min}\varepsilon}\right)^{O(k^2)}$ *data points and in time* $Nd\cdot polylog(k, d, \sigma, \frac{1}{\Delta}, \frac{1}{p_{\min}}, \frac{1}{\varepsilon})$ *outputs a set of vectors* $\{v_i\}_{i=1}^k$ *that with high probability satisfy*

$$\|v_i - w_{\pi(i)}\|_2 \le \varepsilon, \forall i \in [k], \text{ for some permutation } \pi.$$

Proof [Proof of Theorem 1] The theorem follows from Lemma 5 and Lemma 3, the guarantees for the two subroutines used. Note that we recovers each weight up to $\varepsilon_g \le \left(\frac{p_{\min}\Delta}{\sigma d}\right)^{\Omega(k^2)}$. Therefore, only a $\left(\frac{p_{\min}\Delta}{\sigma d}\right)^{\Omega(k^2)}$ fraction of data points from this component are not removed, and only a $\left(\frac{p_{\min}\Delta}{\sigma d}\right)^{\Omega(k^2)}$ fraction of data points from other components get removed. These only causes polynomially small errors to the quantities computed in later steps and can be tolerated by our analysis. ■

Appendix D. Tools

We shall use the following bounds on the Gaussian moments and it's concentration.

Lemma 8 *Let* $g \sim \mathcal{N}(0, \mathbf{I})$*, then for every unit vector* w*, we have that for every non-negative integer* p*,*

$$\mathbb{E}\left[\langle w, g\rangle^{2p} g g^\top\right] = (2p+1)!!ww^\top + (2p-1)!!(\mathbf{I} - ww^\top).$$

Using a standard Matrix Bernstein bound, we can get:

Lemma 9 (Gaussian sample bound) *Let* $g \sim \mathcal{N}(0, \mathbf{\Sigma}^2)$*, let* $g_1, \cdots, g_m$ *be* m *independent samples of* g*. Then for every vector* w *and every non-negative integer* p *and every* $\delta > 0$*, we have that*

$$\Pr\left[\left\|\frac{1}{m}\sum_{i=1}^m \langle w, g_i\rangle^{2p} g_i g_i^\top - \mathbb{E}\left[\langle w, g\rangle^{2p} g g^\top\right]\right\|_2 = \Omega\left(\sqrt{\frac{\|\mathbf{\Sigma}w\|_2^{4p}\,\|\mathbf{\Sigma}\|_2^4\, d\log\frac{1}{\delta}}{m}}\right)\right] \le \delta \quad (46)$$

The following lemma gives an estimation of a (modified) inverse Gaussian, which is used for analyzing the gradient descent step of our algorithm.

Lemma 10 *Suppose $y \sim \mathcal{N}(0, \mathbf{I})$. For every $\zeta > 0$, for every vectors $a, b \in \mathbb{R}^d$, with $\rho = \frac{\langle a,b\rangle}{\|a\|_2\|b\|_2}$,*

$$\frac{1}{4}\frac{\rho\|a\|_2}{\zeta + \|b\|_2} \le \mathbb{E}\left[\frac{\operatorname{sign}(\langle b, y\rangle)\langle a, y\rangle}{|\langle b, y\rangle| + \zeta}\right] \le \frac{\rho\|a\|_2}{\|b\|_2} \le \frac{\|a\|_2}{\|b\|_2}.$$

Proof [Proof of Lemma 10] Without loss of generality assume $b = \|b\|_2 e_1$ and $a = \|a\|_2(\rho e_1 + \sqrt{1-\rho^2}e_2)$. Then

$$\begin{aligned}\mathbb{E}\left[\frac{\operatorname{sign}(\langle b, y\rangle)\langle a, y\rangle}{|\langle b, y\rangle| + \zeta}\right] &= \mathbb{E}\left[\frac{\|a\|_2(\rho y_1 + \sqrt{1-\rho^2}y_2)\operatorname{sign}(y_1)}{\|b\|_2|y_1| + \zeta}\right] \\ &= \rho\|a\|_2\mathbb{E}\left[\frac{|y_1|}{\|b\|_2|y_1| + \zeta}\right]\end{aligned}$$

We know that

$$\frac{|y_1|}{\|b\|_2|y_1| + \zeta} \le \frac{1}{\|b\|_2},$$

and when $|y_1| \ge 1$

$$\frac{|y_1|}{\|b\|_2|y_1| + \zeta} \ge \frac{1}{\zeta + \|b\|_2}.$$

Therefore, we have

$$\frac{1}{4}\frac{\rho\|a\|_2}{\zeta + \|b\|_2} \le \mathbb{E}\left[\frac{\operatorname{sign}(\langle b, y\rangle)\langle a, y\rangle}{|\langle b, y\rangle| + \zeta}\right] \le \frac{\rho\|a\|_2}{\|b\|_2}.$$

where the first inequality follows from $\mathbb{E}[1_{|y_1|\ge 1}] \ge 1/4$. ■

We will also need the Gap-Free Wedin Theorem from (Allen-Zhu and Li, 2016).

Lemma 11 (Gap-Free Wedin Theorem, Lemma B.3 in (Allen-Zhu and Li, 2016)) *For $\varepsilon \ge 0$, let A, B be two PSD matrices such that $\|A - B\|_2 \le \varepsilon$. For every $\mu \ge 0, \tau > 0$, let U be the column orthonormal matrix consisting of eigenvectors of A with eigenvalue $\le \mu$, let V be column orthonormal matrix consisting of eigenvectors of B with eigenvalue $\ge \mu + \tau$, then we have:*

$$\|U^\top V\| \le \frac{\epsilon}{\tau}.$$