On Learning Mixture of Linear Regressions in the Non-Realizable Setting

Avishek Ghosh [1] **Arya Mazumdar** [1] **Soumyabrata Pal** [2] **Rajat Sen** [3]

Abstract

While mixture of linear regressions (MLR) is a well-studied topic, prior works usually do not analyze such models for prediction error. In fact, *prediction* and *loss* are not well-defined in the context of mixtures. In this paper, first we show that MLR can be used for prediction where instead of predicting a label, the model predicts a list of values (also known as *list-decoding*). The list size is equal to the number of components in the mixture, and the loss function is defined to be minimum among the losses resulted by all the component models. We show that with this definition, a solution of the empirical risk minimization (ERM) achieves small probability of prediction error. This begs for an algorithm to minimize the empirical risk for MLR, which is known to be computationally hard. Prior algorithmic works in MLR focus on the *realizable* setting, i.e., recovery of parameters when data is probabilistically generated by a mixed linear (noisy) model. In this paper we show that a version of the popular alternating minimization (AM) algorithm finds the best fit lines in a dataset even when a realizable model is not assumed, under some regularity conditions on the dataset and the initial points, and thereby provides a solution for the ERM. We further provide an algorithm that runs in polynomial time in the number of datapoints, and recovers a good approximation of the best fit lines. The two algorithms are experimentally compared.

1. Introduction

For unsupervised learning, a standard way to represent the presence of sub-populations within an overall population is the use of mixture models. In many important applications ranging from financial models to genetics, a mixture model is used to fit the heterogeneous data. Mixture models, in particular mixture of Gaussian distributions, have been extensively studied from both the perspective of statistical inference and algorithmic approaches (Dasgupta, 1999; Achlioptas & McSherry, 2005; Kalai et al., 2010; Belkin & Sinha, 2010; Arora & Kannan, 2001; Moitra & Valiant, 2010; Feldman et al., 2008; Chan et al., 2014; Acharya et al., 2017; Hopkins & Li, 2018; Diakonikolas et al., 2018; Kothari et al., 2018; Hardt & Price, 2015).

A relatively recent use of mixtures is to model functional relationships (as in supervised learning). Most prominent among these is the *mixed linear regression* problem (De Veaux, 1989). In this setting, each sample is a tuple of (covariates, label). The label is stochastically generated by picking a linear relation uniformly from a set of two or more linear functions, evaluating this function on the covariates and possibly adding noise. The goal is to learn the set of unknown linear functions. The problem has some recent attention (Chaganty & Liang, 2013; Faria & Soromenho, 2010; Städler et al., 2010; Li & Liang, 2018; Kwon & Caramanis, 2018; Viele & Tong, 2002; Yi et al., 2014; 2016; Balakrishnan et al., 2017; Klusowski et al., 2019), with an emphasis on the Expectation Maximization (EM) algorithm and other alternating minimization (AM) techniques.

However, it is to be noted, all of these previous works on mixed-linear regression and their algorithmic guarantees, assumes a generative model for the data, and then provides sample-complexity guarantees for parameter-estimation. On the other hand, in the non-realizable setting, where one is just provided with a set of (covariates, label) pairs, algorithmic methods to find two or more optimal lines were never explored. This is in sharp contrast with high-dimensional Gaussian mixtures, where in the non-realizable setting, the k-means, k-median, and similar objective functions are incredibly popular, and the algorithmic guarantees focus on approximating the objective of cost function (lack of generative modeling implies no parameter estimation to be done). For example, the famous k-means++ algorithm provides an approximate answer with a cost within $\log k$-factor of the optimal value (Vassilvitskii & Arthur, 2006).

This begs the question, in the non-realizable data-dependent setting what is a loss/risk function that one should aim to minimize to find the best-fitting lines? Given a set of lines, our objective should be to assign a data-sample to the line that

---

[1]Halıcıoğlu Data Science Institute (HDSI), UC San Diego, USA [2]Google Research, India [3]Google Research, Palo Alto, USA. Correspondence to: Soumyabrata Pal <soumyabrata@google.com>.

best explains the relation between the covariates and label in the data sample. The general loss function provided in Eq. (1) below captures this fact, which is the minimum of some base loss function evaluated for all the lines - in short the min-loss. On the other hand, the lines should be chosen such that the total error of all the data-samples due to such assignment is minimized. A special case would be that motivated from the k-means objective where we resort to a squared error metric. This is provided in Eq. (2)), and appeared in (Yi et al., 2014). In fact, it was shown in (Yi et al., 2014) that optimizing this loss function is computationally hard.

The definition of the min-loss also motivates the question whether we can use the learned model for prediction using this metric? Indeed, mixture models are mainly studied for statistical inference; it is not clear how one can use a learned mixture of linear regressions for predicting the label, given the covariates. We claim that this can be done by predicting a *list of labels*, instead of predicting a single label for a given set of covariates. This notion of predicting a list is not unusual: for robust linear regression this has been proposed in (Karmalkar et al., 2019). However, in our setting this is a more natural solution, as each element on the list correspond to the label produced by each component model in the mixture. While this is intuitively satisfying, the min-loss and the list-decoding paradigm still needs validation in a learning theoretic sense.

In this paper, we show that indeed it is possible to obtain generalization bounds in the sense of PAC learning functional mixture models, using the min-loss and list decoding. That further motivates the algorithmic question of finding optimal solution for the empirical risk minimization (ERM) problem. Since the ERM for this problem is NP-hard, we propose two approximation algorithms. It is shown that the algorithms provide provable guarantees under the non-realizable setting in terms of converging to either the optimal parameters, or approximating the risk function.

1.1. Summary of Contributions

While the main technical contribution of this paper is analysis of two algorithms in the non-realizable data dependent setting, we believe the proposition of using mixture models for prediction tasks can have potential application in variety of use-cases where it is allowable to predict a list of potential outcomes, instead of a unique answer. The application domains can be recommender systems (Resnick & Varian, 1997) or text auto-completion (Kannan et al., 2016), where making multiple suggestions to a user is common. Thus a learning-theoretic validation of the list-decoding is useful. Our results are summarized below.

Generalization for min-loss: We show that, under our definitions of list-prediction and min-loss, the optimizer of the ERM problem is indeed a good population predictor, by

proving that the mixture-function class with min-loss has a Rademacher complexity not more than $O(k)$ times than that of the base function class, where k is the number of mixture components. This provides validation of our use of min-loss with the list prediction, and also motivates approximation algorithms for the ERM. (Section 3).

An AM algorithm in the non-realizable setting: We propose a version of alternating minimization (AM) algorithm, that converges to the optimal models (lines that minimizes the ERM), if initialized close enough. While the closeness condition can look stringent, in practice the algorithm can be made to converge with few restarts (different choices of initial points). The main technical difficulty in analyzing this algorithm comes from the non generative aspect of data, and we characterize a natural *bias* in this framework and show that, AM converges exponentially provided this bias is (reasonably) small. Additionally, in the non-generative framework, we define a (geometric) data dependent gap indicating a natural complexity measure of the mixture problem, and characterize our results as a function of it. We emphasize that in previous works on AM (Yi et al., 2014; 2016) focused on the generative model in the noise less setting, and hence did not have any bias or data dependent gap. Furthermore, our AM algorithm converges without the requirement of re-sampling fresh data at each iterations, in contrast to the previous works on AM (Jain et al., 2013; Netrapalli et al., 2015; Yi et al., 2014; 2016). (Section 4.1).

A sub-sampling based algorithm: Instead of trying to find the optimal set of lines that minimizes the ERM, we can alternatively seek to find some set of lines that provides a value of the loss function close to the minimum. In this vein, we proposed a sampling-based algorithm that fits lines to all possible partitions of a smaller random subset of the dataset. Since the subset is smaller, this can be done exhaustively. The found lines can then be verified against the entirety of the dataset to find the best choice. Note that, the guarantee this algorithm can provide is very different from the AM algorithm. Specifically, under some mild assumptions, the algorithm can provide a $O(1/\sqrt{\log n})$ additive approximation to objective (3) where n is the number of observed examples. The run-time is polynomial in n but exponential in d (the covariate dimension), provided that the number of mixtures is a constant. In practice, by modifying the algorithm the dependence on d can be made polynomial, but as of now we do not have a theoretical analysis for that version of the algorithm. (Section 4.3).

Practical considerations: experiments: We also conduct experiments on both non-linear synthetic data and real data using the different algorithms that we propose. We reach some interesting conclusions from our empirical study, namely the following: 1) The AM algorithm is very practical in terms of running time but its performance crucially depends on its initialization 2) A minor modification of

the sub-sampling based algorithm has good empirical performance both in terms of generalization error and time complexity making it a viable alternative to AM algorithm in real world applications, and 3) The modified sub-sampling based algorithm can serve as a good initialization for the AM algorithm. (Section 5).

1.2. Related Work

As mentioned all prior works on mixture of linear regressions are in the realizable setting, where the aim is statistical inference. Most papers aim to to do parameter estimation with near-optimal sample complexity, with a large fraction focusing on the performance of the ubiquitous expectation maximization (EM) algorithm.

Notably, in (Balakrishnan et al., 2017), it was shown that the EM algorithm is able to find the correct lines if initialized with close-enough estimates. Furthermore, in the finite sample setting, (Balakrishnan et al., 2017) shows convergence within an ℓ_2 norm ball of the actual parameters, and (Klusowski et al., 2019) then extends in to an appropriately defined cone. In (Yi et al., 2014), the initial estimates were chosen by the spectral method to obtain nearly optimal sample complexity for 2 lines, and then (Yi et al., 2016) extends this to k lines. Interestingly, for the special case of 2 lines, (Kwon & Caramanis, 2018) shows that the any random initialization suffices. The above works assume the covariates to be standard Gaussian random vectors. Finally, in (Li & Liang, 2018), the assumption of standard Gaussian covariates is relaxed (to allow Gaussians with different covariances) and near-optimal sample and computational complexity is achieved, albeit not via the EM algorithm. Another line of research focuses on understanding the convergence rate of AM, and in (Ghosh & Kannan, 2020; Shen & Sanghavi, 2019) it is shown that AM, or its variants can converge at a double exponential (super-linear) rate.

In another line of work in the realizable setting, one is allowed to design covariates to query for corresponding labels (Yin et al., 2019; Krishnamurthy et al., 2019; Mazumdar & Pal, 2020; 2022; Pal et al., 2021). However, none of these works is directly comparable to our setting.

2. Problem Formulation

We are interested in learning a mixture of functions from $\mathcal{X} \to \mathcal{Y}$ for $\mathcal{X} \subseteq \mathbb{R}^d$, that best fits a data distribution $\mathcal{D}$ over $(\mathcal{X}, \mathcal{Y})$. The learner is given access to n samples $\{x_i, y_i\}_{i=1}^n$ from the distribution $\mathcal{D}$. As in usual PAC learning there exists a base function class $\mathcal{H} : \mathcal{X} \to \mathcal{Y}$ and the individual functions of the learned mixture should belong to $\mathcal{H}$. However, we will work in the paradigm of list decoding where the learner is allowed to output a list of responses given a test sample x each of which corresponds to a mixture component function applied

to x. We now formally define a list-decodable function class.

Definition 2.1. For a base function class $\mathcal{H}$, we can construct a k-list-decodable vector valued function class, denoted by $\bar{\mathcal{H}}_k$ such that any $\bar{h} \in \bar{\mathcal{H}}_k$ is defined as

$$\bar{h} = (h_1(\cdot), \cdots, h_k(\cdot))$$

for some set of h_i's such that $h_i \in \mathcal{H}$ for all i. Thus $\bar{h}$'s map $\mathcal{X} \to \mathcal{Y}^k$ and form the new function class $\bar{\mathcal{H}}_k$.

We will omit the k in $\bar{\mathcal{H}}$ when clear from context. As in PAC learning, we need to define a loss measure to quantify the quality of learning. For the list decodable setting, it is natural to compete on the minimum loss achieved by any of the values in the list for a particular example (Kothari et al., 2018). In order to formally define the min-loss setting, let $\ell : \mathcal{Y} \times \mathcal{Y} \to \mathbb{R}^+$ be a base loss function. Then the min-loss is defined as follows,

$$\mathcal{L}(y, \bar{h}(x)) := \min_{j \in [k]} \ell(y, \bar{h}(x)_j) = \min_{j \in [k]} \ell(y, h_j(x))$$

$$L(\bar{h}) := \frac{1}{n} \sum_{i=1}^n \mathcal{L}(y_i, \bar{h}(x_i)). \tag{1}$$

In much of this paper we will specialize to a setting where the base function class is linear i.e $\mathcal{H} = \{\langle \theta, \cdot \rangle : \forall \theta \in \mathbb{R}^d \text{ s.t } \|\theta\|_2 \leq w\}$. In this case, we can follow a simplified notation for the min-loss,

$$L(\theta_1, ..., \theta_k) = \frac{1}{n} \sum_{i=1}^n \min_{j \in [k]} \left\{ (y_i - \langle x_i, \theta_j \rangle)^2 \right\}. \tag{2}$$

$$\text{with}(\theta_1^*, ..., \theta_k^*) = \operatorname*{argmin}_{\{\theta_j\}_{j=1}^k} L(\theta_1, ..., \theta_k). \tag{3}$$

Our focus in this paper is on generalization of the above learning problem. In Section 3 we analyze the Rademacher complexity (Mohri et al., 2018) of learning the mixture function class with respect to the min-loss. However, it is known that the Emprirical Risk Minimization (ERM) problem in Eq. 3 even for the linear setting is NP-Hard (Yi et al., 2014). Therefore, we propose two algorithms to (approximately) solve this ERM problems in Sections 4.1 and 4.3 under different sets of assumptions, even when the dataset does not follow the correct generative model.

3. Generalization Guarantees for Supervised Learning

Our main result in this section is that the Rademacher complexity (Mohri et al., 2018) of learning a mixture of k functions for the min-loss defined above is bounded by k times the Rademacher complexity of the base function class from which the mixture components are drawn. We assume that the base loss function ℓ in Section 2 is μ-Lipschitz i.e

$|\ell(y,y_1) - \ell(y,y_2)| \leq \mu|y_1 - y_2|$ for all $y_1, y_2 \in \mathcal{Y}$. For the sake of completeness recall that the empirical Rademacher complexity of $\mathcal{H}$ (Mohri et al., 2018) is defined as,

$$\hat{\mathfrak{R}}_{S_x}(\mathcal{H}) = \frac{1}{n}\mathbb{E}_{\boldsymbol{\sigma}}\left[\sup_{h \in \mathcal{H}}\sum_{i=1}^{n}\sigma_i h(x_i)\right], \qquad (4)$$

where $\boldsymbol{\sigma}$ is a set of Rademacher RV's. As a notational convenience, we use $\mathcal{D}_x$ to denote the marginal distribution over the covariates and S_x as the set of observed covariates. The Rademacher complexity is then defined as $\hat{\mathfrak{R}}_n(\mathcal{H}) = \mathbb{E}_{S_x \sim \mathcal{D}_x^n}[\hat{\mathfrak{R}}_S(\mathcal{H})]$. We are interested in the (empirical) Radmacher complexity of the mixture function class along with the min-loss which is defined as

$$\hat{\mathfrak{R}}_S(\bar{\mathcal{H}}_k) = \frac{1}{n}\mathbb{E}_{\boldsymbol{\sigma}}\left[\sup_{\bar{h} \in \bar{\mathcal{H}}_k}\sum_{i=1}^{n}\sigma_i \mathcal{L}(y_i, \bar{h}(x_i))\right]$$

where $\{x_i, y_i\}$ are n samples from $\mathcal{D}^n$. Our first result is the following theorem.

Theorem 3.1. *We have the following bound,*

$$\hat{\mathfrak{R}}_S(\bar{\mathcal{H}}_k) \leq k\mu\hat{\mathfrak{R}}_{S_x}(\mathcal{H}) \qquad (5)$$

for any $S = \{x_i, y_i\}_{i=1}^n$ and $S_x = \{x_i\}_{i=1}^n$. Therefore, it also follows that $\hat{\mathfrak{R}}_n(\bar{\mathcal{H}}_k) \leq k\mu\hat{\mathfrak{R}}_n(\mathcal{H})$.

We provide the proof in Appendix B. In the rest of the paper our focus will be on the setting where the base function class is linear i.e $\mathcal{H} = \{\langle\theta,\cdot\rangle : \forall\theta \in \mathbb{R}^d \text{ s.t } \|\theta\|_2 \leq w\}$. It is well-known that when $\mathcal{X} \subseteq \{x \in \mathbb{R}^d : \|x\|_2 \leq R\}$ that $\hat{\mathfrak{R}}_n(\mathcal{H}) = O(wR/\sqrt{n})$ (Mohri et al., 2018, Chapter 11). Therefore, our above result would immediately yield that the Rademacher complexity of the mixture function class with min-loss satisfies $\hat{\mathfrak{R}}_n(\bar{\mathcal{H}}_k) = O((k\mu wR)/\sqrt{n})$.

4. Algorithms in the Non-realizable Setting

4.1. AM Algorithm

Recall that our goal is to address the problem of mixed linear regression, without any generative assumption. Given the data-label pairs $\{x_i, y_i\}_{i=1}^n$, where $x_i \in \mathbb{R}^d$ and $y_i \in \mathbb{R}$, our goal is to fit k linear predictors, namely $\{\theta_j^*\}_{j=1}^k$. We analyze an alternating minimization (AM) algorithm, and show that, provided *suitable* initialization, AM converges at an exponential speed.

We first fix a few notation here. Suppose that the optimal parameters $\{\theta_j^*\}_{j=1}^k$ partition the dataset $\{x_i, y_i\}_{i=1}^n$ in k sets, $\{S_j^*\}_{j=1}^k$, where

$$S_j^* = \{i \in [n] : (y_i - \langle x_i, \theta_1^*\rangle)^2 = \min_{j \in [k]}(y_i - \langle x_i, \theta_j^*\rangle)^2\},$$

and similarly for $S_2^*, \ldots, S_k^*$. In words, S_1^* is the set of observations, where θ_1^* is a better (linear) predictor compared to $\theta_2^*, \ldots, \theta_k^*$.

4.1.1. GRADIENT ALTERNATING MINIMIZATION (AM)

We now propose a provable algorithm in this section. The algorithm to find the best linear predictors in mixed setup is the classical Alternating Minimization (AM) (Yi et al., 2014; 2016). Our proposed approach in formally given in Algorithm 1. Every iteration of the algorithm has in two steps: (a) given data, we first estimate which, out of k predictors obtains minimum loss in fitting the data (through a quadratic loss function); and (b) after this, we take a gradient step with a chosen step size to update the underlying parameter estimates.

In the first step, based on the current estimates, i.e., $\{\theta_j^{(t)}\}_{j=1}^k$, Algorithm 1 first finds the set of indices, $S_j^{(t)}$ for $j \in [k]$, and partitions the observations. We first calculate the residual on each of the possible predictors and choose the one minimizing it, and thereafter a partition is formed by by simply collecting the observations corresponding to a particular predictor. In this way, we form $S_j^{(t)}$ for all $j \in [k]$ at iteration t.

In the second step, we take one iteration of gradient descent over the losses in each partition $S_j^{(t)}$, with an appropriately chosen step size. The loss corresponding to each observation in partition $S_j^{(t)}$ is the quadratic loss. Note that unlike classical AM, we do not obtain the next iterate by solving a linear regression problem—instead we take a gradient step with respect to an appropriately defined loss function. This is done for theoretical tractability of Algorithm 1.

Estimation without-resampling at each iteration: One attractive feature of our algorithm is that, unlike classical AM for mixtures of regression, our algorithm does not require any re-sampling (fresh samples for every iteration). Note that AM based algorithms typically require resampling to nullify the independence issues across iterations, and simplifies the analysis. (Netrapalli et al., 2015) uses it for phase retrieval, (Jain et al., 2013) for low-rank matrix completion, (Ghosh et al., 2020) for distributed clustering and (Yi et al., 2014; 2016) uses it for mixed linear regression.

Note that, since our approach is entirely *data-driven*, the optimal parameters $\{\theta_j^*\}_{j=1}^k$ actually depend on the entire data $\{x_i, y_i\}_{i=1}^n$. Hence, re-sampling would not make sense in the framework. In the sequel, we characterize the cost of not having the flexibility of resampling in our non-realizable setting. In particular we show that we require a stronger initialization condition compared to that of (Yi et al., 2014; 2016), where the authors use the generative framework as well as re-sampling.

4.2. Theoretical Guarantees for Gradient AM

In this section, we obtain theoretical guarantees for Algorithm 1. Recall that we have n samples $\{x_i, y_i\}_{i=1}^n$, where $x_i \in \mathbb{R}^d$, and we do not assume any generative model for y_i. Furthermore, we want to fit k lines to the data—in other

Algorithm 1 Gradient AM for Mixture of Regressions

1: **Input:** $\{x_i, y_i\}_{i=1}^n$, Step size γ
2: **Initialization:** Initial iterate $\{\theta_j^{(0)}\}_{j=1}^k$
3: **for** $t = 0, 1, \ldots, T-1$ **do**
4:     Partition:
5:     Construct $\{S_j^{(t)}\}_{j=1}^k$ such that

$$S_j^{(t)} = \{i \in [n] : (y_i - \langle x_i, \theta_j^{(t)} \rangle)^2$$
$$= \min_{j' \in [k]} (y_i - \langle x_i, \theta_{j'}^{(t)} \rangle)^2\} \forall j \in [k]$$

6:     Gradient Step:

$$\theta_j^{(t+1)} = \theta_j^{(t)} - \frac{\gamma}{n} \sum_{i \in S_j^{(t)}} \nabla F_i(\theta_j^{(t)}), \forall j \in [k]$$

7:     where $F_i(\theta_j^{(t)}) = (y_i - \langle x_i, \theta_j^{(t)} \rangle)^2$
8: **end for**
9: **Output:** $\{\theta_j^{(T)}\}_{j=1}^k$

words, we want to estimate $\{\theta_j^*\}_{j=1}^k$.

Without loss of generality, and for scaling purposes, we take

$$\max_{i \in [n]} \|x_i\| \leq 1 \text{ and}$$

$$\max_{i \in [n]} |y_i| \leq 1.$$

Note that the upper-bound of 1 can indeed be scaled by the maximum norm of the data.

Recall that $\{S_j^*\}_{j=1}^n$ are the true partition given by the optimal parameters $\{\theta_j^*\}_{j=1}^n$. Before providing the theoretical result, let us define a few problem dependent geometric parameters. We define the (problem dependent) separation as

$$\Delta = \min_{j \in [k]} \min_{i \notin S_j^*} |y_i - \langle x_i, \theta_j^* \rangle|.$$

Note that, under the generative model, $\Delta \leq \min_{j \neq \ell} \|\theta_\ell^* - \theta_j^*\|$, which is the usual definition of separation, as proposed in (Yi et al., 2016). The above is a natural extension to the non-generative framework.

We make the following assumption:

Assumption 4.1. For all $i \in S_j^*$, we have $\max_i |y_i - \langle x_i, \theta_j^* \rangle| \leq \lambda$ and $\max_i \|\nabla F_i(\theta_j^*)\| \leq \mu$, where $F_i(\theta) = (y_i - \langle x_i, \theta \rangle)^2$. Furthermore, (d, k, Δ, λ) satisfies $k \exp\left(-C \frac{(\Delta - 2\lambda)^2}{(\max_{j \in [k]} \|\theta_j^*\|)^2} d\right) \leq 1 - c$, where $c < 1$ is a constant.

Let us explain the above assumption in detail. In particular, if y_i has a generative model (without noise), $\lambda = 0, \mu = 0$. This is because, for $i \in S_j^*$, we have $y_i = \langle x_i, \theta_j^* \rangle$, and the

above term vanishes. The previous works on mixed linear regression, for example (Yi et al., 2014; 2016; Ghosh & Kannan, 2020), analyzes this setup exactly. A generative model is assumed and the analysis is done in the noise less case. However, we do not assume any generative model here and hence we need to control these bias parameters. Shortly, we show that, provided μ is sufficiently small, Algorithm 1 converges at an exponential speed.

Interpretation of μ and λ: We emphasize that λ is (a bound) related to the variance of the data generating process. As we will se subsequently, we can tolerate reasonably large λ as it only appears inside a decaying exponential. It intuitively implies that we can tolerate outlier data as long as its variance is λ bounded. Furthermore, μ is bounding the gradient norm of the loss function for outlier data. μ is large, if data points are far away from the regressor, θ_i^* (an hence an outlier). We can tolerate the outliers upto a gradient norm of μ. This is also connected to the variance of the data generating process.

Finally, the condition on (d, Δ, λ) is quite mild, since the left hand side is an exponential decaying function with dimension d. Hence, for moderate dimension, this mild condition is easily satisfied. Furthermore, it decays exponentially with the gap Δ as well, and so with a separable problem, the above condition holds.

Furthermore, to avoid degeneracy, i.e., $\{|S_j^*|\}_{j=1}^k$ is small, we have, for all $i \in [n]$, $\min_{j \in [k]} \mathbb{P}[i \in S_j^*] \geq \bar{c}$ where $\bar{c}$ is a constant. Note that these type of assumption is quite common, and used in previous literature as well. As an instance, (Yi et al., 2014; 2016; Ghosh & Kannan, 2020) use it for mixture of regressions, (Ghosh et al., 2020) use it for AM in a distributed clustering problem, and (Ghosh et al., 2019) use it for max-affine regression.

We now present our main results. In the following, we consider one iteration of Algorithm 1, and show a contraction in parameter space.

Theorem 4.2. *Let Assumption 4.1 holds, $n \geq Ckd$ and the covariates $\{x_i\}$ have zero mean. Furthermore, suppose*

$$\|\theta_j - \theta_j^*\| \leq \frac{\tilde{c}}{\sqrt{d}} \|\theta_j^*\|.$$

for all $j \in [k]$. Then, running one iteration of Algorithm 1 with $\gamma = 1/4\bar{c}$, yields $\{\theta_j^+\}_{j=1}^k$ satisfying

$$\|\theta_j^+ - \theta_j^*\| \leq \frac{1}{2} \|\theta_j - \theta_j^*\| + c_1 \mu \|\theta_j^*\|$$
$$+ ck \exp\left(-c_1 \frac{(\Delta - 2\lambda)^2}{(\max_j \|\theta_j^*\|)^2} d\right) \|\theta_j^*\|,$$

with probability at least $1 - C_1 \exp(-C_2 d)$.

Remark 4.3 (Initialization). The $1/\sqrt{d}$ factor in the initialization occurs owing to a covering net argument.

We emphasize that in the standard AM with realizable setting, the regressors $\{\theta_i^*\}_{i=1}^k$ are fixed vectors, but in our non-realizable setting, they are dependent on the dataset $\{x_i, y_i\}_{i=1}^n$. Given this, techniques that yield dimension independent initialization like *resampling in each iteration*, *Leave One Out (LOO)* can not be applied. We overcome this challenge by using an ϵ-net, and as a result, we have the dimension dependence. As of now, we are unaware of any technical tools that remove the dimension dependence, and we hypothesize that it may be hard (or even impossible) to remove this condition theoretically. However, as we show in experiments, this condition can be relaxed.

Corollary 4.4. *Suppose μ satisfies*

$$\mu \leq c_2 \left(\frac{\tilde{c}}{\sqrt{d}} - ck\exp\left(-c_1 \frac{(\Delta - 2\lambda)^2}{(\max_j \|\theta_j^*\|)^2} d \right) \right),$$

where c_2 is an universal constant. With proper choice of c_2, we obtain, for all $j \in [k]$,

$$\|\theta_j^+ - \theta_j^*\| \leq \frac{3}{4} \|\theta_j - \theta_j^*\|,$$

which implies a contraction.

Remark 4.5. Note that the above implies that after sufficiently many iterations, the iterate of Algorithm 1 converges exactly to $\{\theta_j^*\}_{j=1}^k$ (exact parameter recovery). This is in contrast to existing works (Balakrishnan et al., 2017; Klusowski et al., 2019) where, in the finite sample noisy setting, the convergence is guaranteed on a set close to the optimal parameters. Note that our non-generative framework may accommodate noise in the observations as well.

Proof Sketch: We first show that by the virtue of the initialization condition and the separation, the partitions produced by Algorithm 1 are *reasonable* good estimates of the true partitions $\{S_j^*\}_{j=1}^k$. We crucially exploit the bounded-ness of covariates along with a discretization (covering) argument to obtain this. Then, we run gradient descent on the estimated partitions, and use the bias (μ, λ) to characterize its performance. The linear convergence rate comes from the analysis of gradient descent on the quadratic loss, and the additional errors originate from (a) biases in the system and (b) estimation error in the partitions.

4.3. Sub-Sample Algorithm

We now provide an approximation algorithm to the objective in Section 2 which requires no initialization conditions like Algorithm 1. It has been established in (Yi et al., 2014) that the problem is NP-hard without any assumptions and (Yi et al., 2014) only show recovery guarantees in the realizable setting assuming isotropic Gaussian covariates. Our sub-sample based Algorithm 2 works without such assumptions on the covariates and provides an additive approximation

---

Algorithm 2 Sub-sampled data driven learning of mixture of k regressions

1: Select a set of A of data-points uniformly at random with replacement from the dataset ($|A|$ specified later).
2: **for** each partition $\{P_i\}_{i=1}^k$ of A **do**
3:      Let $\theta_i := \theta_{\mathrm{ls}}(P_i)$ for all i.
4:      Use θ_i's to partition all the samples into $S = \cup_{i=1}^k S_{\theta_i}(S)$.
5:      Let $\mathcal{E}(\{P_i\}) = \frac{1}{n}\sum_{i=1}^k \mathcal{E}_{ls}(S_{\theta_i}(S), \theta_i)$.
6: **end for**
7: Find θ_i's corresponding to $\{P_i^m\} = \arg\min_{\{P_i\}} \mathcal{E}(\{P_i\})$. Return $\{\theta_i^+\}$ such that $\forall i$, $\theta_i^+ = \theta_{\mathrm{ls}}(S_{\theta_i}(S))$.

---

of $\tilde{O}(1/\sqrt{\log n})$ with a runtime of $\tilde{O}(n^{c\log k}\exp(k^2 d\log k))$, with high probability.

Before we proceed we need some notation specific to Algorithm 2. Recall that we have n examples $S = \{x_i, y_i\}_{i=1}^n$ in total. For a set of samples $P \subseteq S$, $\theta_{\mathrm{ls}}(P)$ refers to the least-squares solution over that set i.e

$$\theta_{\mathrm{ls}}(P) = \arg\min_\theta \sum_{i \in P}(y_i - \langle\theta, x_i\rangle)^2.$$

Further, $\mathcal{E}_{ls}(P, \theta) = \sum_{i \in P}(y_i - \langle\theta, x_i\rangle)^2$ and

$$S_{\theta_l}(P) = \{i \in P : \text{s.t } l = \arg\min_{j \in [k]}(y_i - \langle\theta_j, x_i\rangle)^2\}$$

i.e the set of data points in P for which the θ_l is the best fit among $\{\theta_j\}_{j=1}^k$.

Let $\{B_i^*\}_{i=1}^k$ be the optimal partition of the samples in S i.e each partition has a different θ_i^* as the solution for least-square and together they minimize the objective in Section 2. For any $B \subseteq S$, let us define $\Sigma_B = (1/|B|)\sum_{i \in B} x_i x_i^T$ and $\theta_B = \Sigma_B^{-1}(1/|B|\sum_{i \in B} x_i y_i)$. The theoretical guarantees on Algorithm 2 are provided under the following mild assumptions.

Assumption 4.6. The dataset A satisfies the following:

- Let $\alpha_i := |B_i^*|/n$ and we have that $\alpha_i \geq \alpha > 0$ for all $i \in [k]$.
- The minimum covariance matrix eigen-values of the partitions, $\lambda_{\min}(\Sigma_{B_i^*}) > \lambda_{\min} > 0$ for all $i \in [k]$.
- $\left\|\Sigma_{B_i^*}^{-1/2} x_j(y_j - \theta_j^* . x_j)\right\| \leq \gamma\sqrt{d}$ almost surely, for all $j \in B_i^*$. Note that this γ just needs to be finite and only appears in lower order terms.
- $\frac{1}{|B_i^*|}\sum_{j \in B_i^*}(y_j - \theta_j^* . x_j)^2 \leq B_d$ i.e the expected squared bias is bounded in each part.
- The response and the covariates satisfy $|y_i| \leq b$, $\|x_i\| \leq R$. We also assume that θ_i^*'s are s.t $\|\theta_i^*\| \leq w$.

The first assumption just ensures that none of the parts of the optimal partitions are $o(n)$. The rest of the assumptions are related to assumptions of Theorem 2 in (Hsu et al., 2011), a result that we crucially use in our analysis, as will be evident in our proof sketch below.

We are now ready to present our main theorem which is an approximation result for the objective in Section 2.

Theorem 4.7. *Under Assumption 4.6, we guarantee with probability at least $1 - 2\delta$ that the solution returned by Algorithm 2 satisfies:* $L(\theta_1^+, \dots, \theta_k^+) \leq L(\theta_1^*, \dots, \theta_k^*) + \epsilon$, *provided* $|A| = \tilde{\Omega}\left(\frac{1}{\epsilon^2} k^2 \frac{\alpha}{\lambda_{\min}} \left(d + \log \frac{k}{\delta}\right)\right)$.

Proof Sketch: Of all the partitions of A, one of them $\{\tilde{P}_i\}$ would conform to the optimal partitioning B_i^* of the whole dataset. In that case the samples from $\tilde{P}_i$ can be akin to a random design regression problem where the examples are drawn i.i.d from B_i^*. Therefore, with a careful application of random design regression results (Hsu et al., 2011), we can argue that the error incurred by $\theta_{\tilde{P}_i^*}$ cannot be too much larger than that of θ_i^* on the samples in B_i^*. This will be true for all $i \in [k]$. The error incurred by the solution of Algorithm 2 is bound to be even smaller, thus concluding the argument.

The above result implies that when the number of sub-sampled datapoints in Algorithm 2 is $\Omega(k^2(\alpha/\lambda_{\min}) \log n(d + \log(k/\delta)))$, we can guarantee an approximation of the order of $1/\sqrt{\log n}$. Therefore, the loop in line 2 of Algorithm 2 implies a runtime of $\tilde{O}(n^{c \log k} \exp(k^2 d \log k))$ where c is an universal constant. Thus our time complexity is polynomial in n but exponential in the dimension d. (Yi et al., 2014) showed that the objective in (3) is NP-Hard with respect to the dimension. We conjecture that the the problem is also hard with respect to n, but an analysis of the same is beyond the scope of this paper.

5. Experiments

In this section, we validate our theoretical findings via experiments.

A note on AM convergence: Note that in many supervised real world data problems with incomplete data, corruptions or latent clusters, the most natural, computationally efficient and scalable algorithm that is widely used in practice is the AM algorithm (and its soft counterpart EM). For real world data, *the assumption of realizibility is naturally violated*, and as we will show subsequently (via Movielens data), that AM converges and attains small error. Furthermore, existing literature have already demonstrated the convergence and superior performance of AM type algorithms on real world data (see (Ghosh et al., 2020; Balázs, 2016) , where in (Ghosh et al., 2020) experiments are done with MNIST, CIFAR 10 and FEMNIST data and (Balázs, 2016) uses

---

Algorithm 3 Sub-sampled data driven learning of mixture of k regressions with random partitions

Require: Input hyper-parameter h
1: Select a set A of data-points uniformly at random with replacement from the dataset ($|A|$ specified in Thm 4.7).
2: **for** each of h random partitions $\{P_i\}_{i=1}^k$ of A **do**
3:     Let $\theta_i := \theta_{\text{rob/ls}}(P_i)$ for all i.
4:     Use θ_i's to partition all the samples into $S = \cup_{i=1}^k S_{\theta_i}(S)$.
5:     Let $\mathcal{E}(\{P_i\}) = \frac{1}{n} \sum_{i=1}^k \mathcal{E}_{rob/ls}(S_{\theta_i}(S), \theta_i)$.
6: **end for**
7: Find θ_i's corresponding to $\{P_i^m\} = \text{argmin}_{\{P_i\}} \mathcal{E}(\{P_i\})$. Return $\{\theta_i^+\}$ such that $\forall i$, $\theta_i^+ = \theta_{\text{rob/ls}}(S_{\theta_i}(S))$.

---

aircraft profile drag and inventory control data). All of these (non realizable) real data experiments work *without any suitable initialization*. Random initialization with multiple restarts is a standard, well accepted technique for AM.

5.1. Synthetic Dataset:

Note that in Step 2 of Algorithm 2, we iterate through all partitions of the set A; unfortunately, in practice, this step is time-expensive. However it is possible to modify this step slightly to make this algorithm anytime i.e. we can have a budget on time and recover the best possible solution within that time. Below, we propose Algorithm 3 (that is a minor modifications of Algorithm 2) for practical applications:

On the other hand, in Step 2 of Algorithm 3, we iterate through h random partitions (where h is chosen according to the time budget). There are two options two fit each part of the partition of the set A in Step 3 namely 1) we can fit a linear model as was the case in Algorithm 3 2) we can fit a linear regression model that is robust to outliers in Step 3 (can be implemented using `sklearn.linearmodel.RANSACRegressor()` in python). Since random partitions will contain elements from from all sets $\{B_i^*\}_{i=1}^k$, a robust regression model is able to better fit each part of the partition.

Dataset	LR	Alg 1 (Mean, Var)	Alg 3 (Mean,Var)
A	20.13	(16.60,16.98)	(11.84,0.41)
B	19605	N/A	(5002.03, 50925)
C	12134	N/A	(7.24,0.16)

Table 1. Mean (Mean) and Variance (Var) of min-loss on Test data generated from non-linear synthetic datasets. LR corresponds to a simple linear regression model; N/A implies that the algorithm did not converge during training in any implementation.

Non-linear datasets: We implement and compare the per-

u	v	A0 (Mean,Var)	A1 (Mean,Var)	A2(Mean,Var)	A3(Mean, Var)	A4(Mean,Var,DNC)
'1010'	'2116'	(0.709,0)	(0.4586,0.0003)	(0.33,0.0005)	**(0.2416,0.0003)**	(0.5949,0.07,6)
'752'	'1941'	(1.423, 0)	(0.801, 0.001)	(0.416, 0.0005)	**(0.360, 0.001)**	(0.588, 0.13, 3)
'752'	'2116'	(1.475,0)	(0.90, 0.0009)	**(0.36, 0.0023)**	(0.40, 0.003)	(0.79, 0.17, 6)
'752'	'2909'	(1.40, 0)	(0.855,0.0014)	**(0.42,0.0005)**	(0.425,0.002)	(0.96, 0.21, 5)
'1010'	'4725'	(0.71, 0)	(0.4566, 0.0002)	(0.33, 0.0007)	**(0.269, 0.0003)**	(0.51, 0.04, 7)

Table 2. Training error (min-loss) for the 5 different algorithms A0 (Linear Regression), A1 (Algorithm 3 with linear model), A2 (Algorithm 3 with robust linear model), A3 (Algorithm 1 with initialization by Algorithm 3), A4 (Algorithm 1 with random initialization). For each algorithm, we report the mean (Mean) and variance (Var) of min-loss on training data over 30 implementations. For A4, we also report the number of times (DNC) out of 30 that the algorithm did not converge. For each row, the numbers in bold correspond to the model with the minimum average training error.

u	v	A0 (Mean,Var)	A1 (Mean,Var)	A2(Mean,Var)	A3(Mean, Var)	A4(Mean,Var)
'1010'	'2116'	(0.62,0)	(0.394,0.0003)	(0.288,0.0009)	**(0.23,0.0005)**	(0.56,0.044)
'752'	'1941'	(1.364,0)	(0.742,0.001)	(0.399,0.0004)	**(0.3133,0.001)**	(0.6224,0.14)
'752'	'2116'	(1.29,0)	(0.82,0.0012)	**(0.30,0.0014)**	(0.406,0.008)	(0.88,0.20)
'752'	'2909'	(1.45,0)	(0.846,0.0025)	(0.49,0.0009)	**(0.4191,0.0029)**	(1.13,0.31)
'1010'	'4725'	(0.685,0)	(0.4327,0.0004)	(0.289,0.0009)	**(0.258,0.0001)**	(0.622,0.088)

Table 3. Test error (min-loss) for the 5 different algorithms A0 (Linear Regression), A1 (Algorithm 3 with linear model), A2 (Algorithm 3 with robust linear model), A3 (Algorithm 1 with initialization by Algorithm 3), A4 (Algorithm 1 with random initialization). For each algorithm, we report the mean (Mean) and variance (Var) of min-loss on test data over 30 implementations. For each row, the numbers in bold correspond to the model with the minimum average test error.

formance of our algorithms on three non-linear datasets generated by `sklearn` namely `makefriedman1` [A](fri, a), `makefriedman2`[B] (fri, b) and `makefriedman3`[C] (fri, c). Note that all these datasets are non-realizable.

All the three datasets A, B and C comprises of 3200 samples in the train data and 800 samples in the test data. The covariates in datasets A, B and C comprises of 5, 4 and 4 features respectively. We train and test three different algorithms on these datasets and report the average min-loss on the test set in 1. In Table 5 in the appendix we also report the training errors for the sake of completeness The three algorithms are the following: 1) *Linear Regression (LR)* As a baseline, we fitted a linear regression model on the training dataset and reported the train/test error (Mean squared error) 2) *Algorithm 1* For dataset A, we implement Algorithm 1 with $\gamma = 0.1$ and random initialization (every element of $\theta_1^{(0)}$,$\theta_2^{(0)}$ is generated i.i.d according to a Gaussian with mean 0 and standard deviation 10). We report the average train/test min-loss over 50 iterations. For datasets B and C, Algorithm 1 did not converge while training with the many different initializations that we used (including random initializations and initialization with the best solution provided by Algorithm 3 using the linear model). 3) *Algorithm 3 with robust linear model:* For all three datasets A, B and C, this algorithm was implemented 30 times with $h = 1000$ and the average train/test min-loss is reported. Empirically, we can conclude that the performance

of AM algorithm depends crucially on the initialization; this is reflected in the high variance in min-loss for both train and test data (dataset A) and non-convergence for datasets B and C. On the other hand, Algorithm 3 (with the robust linear model) does not have the initialization drawback and has improved empirical performance.

We provide *negative* results of AM on synthetic datasets B and C because these are *outlier datasets* where the AM algorithm has a poor performance and therefore demands further investigation both theoretically and empirically.

5.2. Real-world Data: Movielens Data

For experiments on real datasets, we use the Movielens 1M dataset[1] that consists of 1 million ratings from $m = 6000$ users on $n = 4000$ movies. In our first pre-processing step, we generate a 4-dimensional embedding of the users and movies in the following way. Note that we observe only a fraction of the entries of the entire user-item matrix M. Let the set of users be described by a matrix $U \in \mathbb{R}^{m \times 4}$ and the set of movies be described by a matrix $V \in \mathbb{R}^{n \times 4}$. In that case if Ω is the set of observed entries in M, then a usual way (Lu et al., 2013) to estimate U,V is to optimize the following objective: $\min_{U,V \in \mathbb{R}^{m \times 4}} \sum_{(i,j) \in \Omega} (M_{ij} - \langle U_i, V_j \rangle)^2$ where U_i (V_j) denotes the ith (jth) row of U (V).

---

[1]https://grouplens.org/datasets/movielens/1m/

Subsequently, we choose an arbitrary pair of users (u,v) such that both the users have rated a large number of movies and furthermore, u and v have significantly different preferences. We ensure this by choosing (u,v) such that u has given a high rating to a small fraction of movies they have rated while v has given a high rating to a large fraction of movies they have rated. Next, for user u, we create a dataset (X_u, y_u) where each row of X_u corresponds to the embedding of a movie that user u has rated while the corresponding element of y_u is the rating for the aforementioned movie. We create a similar dataset (X_v, y_v) for user v and subsequently, we combine these datasets (and shuffle) to generate a single dataset (X,y). We split this dataset into train and test (80 : 20); in Table 6 (in the appendix), we report the user ids and number of samples in train and test data.

We compute the min-loss for five different algorithms on the train and test data (averaged over 30 implementations) for each pair of users and report them in Tables 6 and 3. The algorithms that we have implemented are the following: 1) *Linear Regression (A0):* As a baseline, we fitted a simple linear regression model on the training data and computed the mean squared error on the test data (which corresponds to the min loss in case of a single line) 2) *Algorithm 3 with linear model (A1):* We implement Algorithm 3 with $|A| = 150$ and $h = 1000$ and in Steps 3,5 we use the Linear Regression model 3) *Algorithm 3 with robust regression model (A2):* We implement Algorithm 3 with $|A| = 150$, $h = 1000$ and in Steps 3,5 we use the robust regression model 4) *Algorithm 1 (A3)* We implement Algorithm 1 initialized with the lines that is obtained after fitting Algorithm 3 (A1) on the training data 5) *Algorithm 1 (A4)* We implement Algorithm 1 with random initialization $\theta_1^{(0)}, \theta_2^{(0)}$ where each entry of the initialized vectors are generate i.i.d according to a Gaussian with zero mean and variance 2.

6. Conclusion

It is evident from Tables 6 and 3 that Algorithm 3 (using the robust linear module) and Algorithm 1 (initialized by the best solution of Algo. 3 where we use the linear model to fit each part of the partition of A) has the best empirical performance overall. Note that AM (Algorithm 1) often, due to bad initialization, gets stuck to local minima. In Tables 6 and 3, it is clear from the final column (A4) that the variance of the AM algorithm (with random initialization), is very high. Algorithm 3 (using the linear regression module) can provide a viable alternative to spectral clustering as an initialization technique for the AM algorithm. Algorithm 3 (using the robust linear module aka A2) has a similar performance (both in terms of train/test error and time complexity) to Algorithm 1. It is a very interesting future direction to study the guarantees of A2 theoretically. Unlike Algorithm 1, A2 does not suffer from the initialization issue and can be a

potentially viable alternative to AM algorithm in practice.

References

Makefriedman1 dataset. `https://scikit-learn.org/stable/modules/generated/sklearn.datasets.makefriedman1.html`, a. Accessed: 2022-01-25.

Makefriedman2 dataset. `https://scikit-learn.org/stable/modules/generated/sklearn.datasets.makefriedman2.html`, b. Accessed: 2022-01-25.

Makefriedman3 dataset. `https://scikit-learn.org/stable/modules/generated/sklearn.datasets.makefriedman3.html`, c. Accessed: 2022-01-25.

Acharya, J., Diakonikolas, I., Li, J., and Schmidt, L. Sample-optimal density estimation in nearly-linear time. In *Proceedings of the Twenty-Eighth Annual ACM-SIAM Symposium on Discrete Algorithms*, pp. 1278–1289. SIAM, 2017.

Achlioptas, D. and McSherry, F. On spectral learning of mixtures of distributions. In *Conference on Learning Theory*, 2005.

Arora, S. and Kannan, R. Learning mixtures of arbitrary gaussians. In *Symposium on Theory of Computing*, 2001.

Balakrishnan, S., Wainwright, M. J., and Yu, B. Statistical guarantees for the em algorithm: From population to sample-based analysis. *The Annals of Statistics*, 45(1): 77–120, 2017.

Balázs, G. Convex regression: theory, practice, and applications. 2016.

Belkin, M. and Sinha, K. Polynomial learning of distribution families. In *Foundations of Computer Science*, 2010.

Chaganty, A. T. and Liang, P. Spectral experts for estimating mixtures of linear regressions. In *International Conference on Machine Learning*, pp. 1040–1048. PMLR, 2013.

Chan, S.-O., Diakonikolas, I., Servedio, R. A., and Sun, X. Efficient density estimation via piecewise polynomial approximation. In *Proceedings of the forty-sixth annual ACM symposium on Theory of computing*, pp. 604–613. ACM, 2014.

Dasgupta, S. Learning mixtures of gaussians. In *Foundations of Computer Science*, pp. 634–644, 1999.

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

Diakonikolas, I., Kane, D. M., and Stewart, A. List-decodable robust mean estimation and learning mixtures of spherical gaussians. In *Proceedings of the 50th Annual ACM SIGACT Symposium on Theory of Computing*, pp. 1047–1060. ACM, 2018.

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

Feldman, J., O'Donnell, R., and Servedio, R. A. Learning mixtures of product distributions over discrete domains. *SIAM Journal on Computing*, 2008.

Ghosh, A. and Kannan, R. Alternating minimization converges super-linearly for mixed linear regression. In *International Conference on Artificial Intelligence and Statistics*, pp. 1093–1103. PMLR, 2020.

Ghosh, A., Pananjady, A., Guntuboyina, A., and Ramchandran, K. Max-affine regression: Provable, tractable, and near-optimal statistical estimation. *arXiv preprint arXiv:1906.09255*, 2019.

Ghosh, A., Chung, J., Yin, D., and Ramchandran, K. An efficient framework for clustered federated learning. *arXiv preprint arXiv:2006.04088*, 2020.

Hardt, M. and Price, E. Tight bounds for learning a mixture of two gaussians. In *Symposium on Theory of Computing*, 2015.

Hopkins, S. B. and Li, J. Mixture models, robustness, and sum of squares proofs. In *Symposium on Theory of Computing*, 2018.

Hsu, D., Kakade, S. M., and Zhang, T. An analysis of random design linear regression. *arXiv preprint arXiv:1106.2363*, 2011.

Jain, P., Netrapalli, P., and Sanghavi, S. Low-rank matrix completion using alternating minimization. In *Proceedings of the forty-fifth annual ACM symposium on Theory of computing*, pp. 665–674, 2013.

Kalai, A. T., Moitra, A., and Valiant, G. Efficiently learning mixtures of two gaussians. In *Proceedings of the forty-second ACM symposium on Theory of computing*, pp. 553–562. ACM, 2010.

Kannan, A., Kurach, K., Ravi, S., Kaufmann, T., Tomkins, A., Miklos, B., Corrado, G., Lukacs, L., Ganea, M., Young, P., et al. Smart reply: Automated response suggestion for email. In *Proceedings of the 22nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, pp. 955–964, 2016.

Karmalkar, S., Klivans, A., and Kothari, P. List-decodable linear regression. *Advances in neural information processing systems*, 2019.

Klusowski, J. M., Yang, D., and Brinda, W. Estimating the coefficients of a mixture of two linear regressions by expectation maximization. *IEEE Transactions on Information Theory*, 65(6):3515–3524, 2019.

Kothari, P. K., Steinhardt, J., and Steurer, D. Robust moment estimation and improved clustering via sum of squares. In *Proceedings of the 50th Annual ACM SIGACT Symposium on Theory of Computing*, pp. 1035–1046. ACM, 2018.

Krishnamurthy, A., Mazumdar, A., McGregor, A., and Pal, S. Sample complexity of learning mixture of sparse linear regressions. In *Advances in Neural Information Processing Systems (NeurIPS)*, 2019.

Kwon, J. and Caramanis, C. Global convergence of em algorithm for mixtures of two component linear regression. *arXiv preprint arXiv:1810.05752*, 2018.

Li, Y. and Liang, Y. Learning mixtures of linear regressions with nearly optimal complexity. In *Conference On Learning Theory*, pp. 1125–1144. PMLR, 2018.

Lu, J., Hoi, S., and Wang, J. Second order online collaborative filtering. In *Asian Conference on Machine Learning*, pp. 325–340. PMLR, 2013.

Mazumdar, A. and Pal, S. Recovery of sparse signals from a mixture of linear samples. In *International Conference on Machine Learning (ICML)*, 2020.

Mazumdar, A. and Pal, S. On learning mixture models with sparse parameters. *arXiv preprint arXiv:2202.11940*, 2022.

Mohri, M., Rostamizadeh, A., and Talwalkar, A. *Foundations of machine learning*. MIT press, 2018.

Moitra, A. and Valiant, G. Settling the polynomial learnability of mixtures of gaussians. In *Foundations of Computer Science*, 2010.

Netrapalli, P., Jain, P., and Sanghavi, S. Phase retrieval using alternating minimization. *IEEE Transactions on Signal Processing*, 63(18):4814–4826, 2015.

Pal, S., Mazumdar, A., and Gandikota, V. Support recovery of sparse signals from a mixture of linear measurements. *Advances in Neural Information Processing Systems*, 34, 2021.

Resnick, P. and Varian, H. R. Recommender systems. *Communications of the ACM*, 40(3):56–58, 1997.

Shen, Y. and Sanghavi, S. Iterative least trimmed squares for mixed linear regression. *arXiv preprint arXiv:1902.03653*, 2019.

Städler, N., Bühlmann, P., and Van De Geer, S. l1-penalization for mixture regression models. *Test*, 19(2): 209–256, 2010.

Vassilvitskii, S. and Arthur, D. k-means++: The advantages of careful seeding. In *Proceedings of the eighteenth annual ACM-SIAM symposium on Discrete algorithms*, pp. 1027–1035, 2006.

Vershynin, R. *High-dimensional probability: An introduction with applications in data science*, volume 47. Cambridge university press, 2018.

Viele, K. and Tong, B. Modeling with mixtures of linear regressions. *Statistics and Computing*, 12(4):315–330, 2002.

Yi, X., Caramanis, C., and Sanghavi, S. Alternating minimization for mixed linear regression. In *International Conference on Machine Learning*, pp. 613–621. PMLR, 2014.

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

Yin, D., Pedarsani, R., Chen, Y., and Ramchandran, K. Learning mixtures of sparse linear regressions using sparse graph codes. *IEEE Transactions on Information Theory*, 65(3):1430–1451, 2019.

Appendix

A. Proof of Theorem 4.2

Without loss of generality, let us focus on $j = 1$, i.e., θ_1^+. We have

$$\|\theta_1^+ - \theta_1^*\| = \|\theta_1 - \theta_1^* - \frac{\gamma}{n} \sum_{i \in S_1} \nabla F_i(\theta_1)\|$$

$$= \|(\theta_1 - \theta_1^*) - \frac{\gamma}{n} \sum_{i \in S_1} (\nabla F_i(\theta_1) - \nabla F_i(\theta_1^*)) - \frac{\gamma}{n} \sum_{i \in S_1} \nabla F_i(\theta_1^*)\|$$

$$\leq \underbrace{\|(\theta_1 - \theta_1^*) - \frac{\gamma}{n} \sum_{i \in S_1} (\nabla F_i(\theta_1) - \nabla F_i(\theta_1^*))\|}_{T_1} + \underbrace{\frac{\gamma}{n} \|\sum_{i \in S_1} \nabla F_i(\theta_1^*)\|}_{T_2}.$$

Let us first consider T_1. We have

$$T_1 = \|(I - \frac{2\gamma}{n} \sum_{i \in S_1} x_i x_i^\top)(\theta_1 - \theta_1^*)\|.$$

We first observe

$$\sigma_{\min}(\frac{1}{n} \sum_{i \in S_1} x_i x_i^\top) \geq \sigma_{\min}(\frac{1}{n} \sum_{i \in S_1 \cap S_1^*} x_i x_i^\top)$$

In order to upper bound T_1, we need to control the size of $S_1 \cap S_1^*$. Using Lemma C.2, we argue that provided the initialization condition is satisfied, $|S_1 \cap S_1^*| \geq \frac{1}{2}|S_1^*| \geq c p_1 n$ with probability at least $1 - \exp(-c_1 n)$, where p_1 is the fraction of observations, for which θ_1^* is a better predictor. Also Lemma A.2 shows that conditioned on $i \in S_1 \cap S_1^*$, the random variables x_i is still sub-Gaussian with constant parameter. Hence, using (Vershynin, 2018), we have

$$\sigma_{\min}(\frac{1}{n} \sum_{i \in S_1} x_i x_i^\top) \geq c_1 p_1 n$$

with probability at least $1 - \exp(-d)$, and as a result,

$$T_1 \leq (1 - c\gamma p_1)\|\theta_1 - \theta_1^*\|,$$

with probability at least $1 - \exp(-c_1 n) - \exp(-d) \geq 1 - c_1 \exp(-c_2 d)$.

Let us now consider the term T_2. We have

$$T_2 = \frac{\gamma}{n} \|\sum_{i \in S_1} \nabla F_i(\theta_1^*)\|$$

$$\leq \frac{\gamma}{n} \sum_{i \in S_1} \|\nabla F_i(\theta_1^*)\|$$

$$= \frac{\gamma}{n} \sum_{i \in S_1 \cap S_1^*} \|\nabla F_i(\theta_1^*)\| + \frac{\gamma}{n} \sum_{j=2}^{k} \sum_{i \in S_1 \cap S_j^*} \|\nabla F_i(\theta_1^*)\|$$

For $i \in S_1^*$, the term $\|\nabla F_i(\theta_1^*)\| \leq \mu$. With this, we have

$$T_2 \leq \frac{\gamma}{n}|S_1 \cap S_1^*|\mu + \frac{\gamma}{n} \sum_{j=2}^{k} |S_1 \cap S_j^*| \max_i \|\nabla F_i(\theta_1^*)\|$$

$$\leq \frac{\gamma}{n}|S_1^*|\mu\|\theta_1^*\| + \frac{c_1 \gamma}{n}(k-1)\exp\left(-c_2 \frac{\Delta_1^2}{\rho^2} d\right) n \max_i \|\nabla F_i(\theta_1^*)\|,$$

where $\rho = (\max_{j \in [k]} \|\theta_j^*\|)^2$ and $\Delta_1 = \Delta - 2\lambda$. Here, we use Lemma C.2 in conjunction with sub-Gaussian concentration. Finally, we have

$$T_2 \le \gamma p_1 \mu + C_1 k \gamma \exp\left(-c_2 \frac{\Delta_1^2}{\rho^2} d\right) (\max_i \|x_i\|^2 \|\theta_1^*\| + \max_i \|x_i\| \|y_i\|)$$

$$\le \gamma p_1 \mu \|\theta_1^*\| + c_1 k \gamma \exp\left(-c_2 \frac{\Delta_1^2}{\rho^2} d\right) \|\theta_1^*\|,$$

where we use that $\|x_i\| \le 1$ for all i, and $|y_i| \le 1$ for all i.

Combining T_1 and T_2, we have

$$\|\theta_1^+ - \theta_1^*\| \le (1 - c\gamma p_1) \|\theta_1 - \theta_1^*\| + \gamma p_1 \mu \|\theta_1^*\| + c_1 k \gamma \exp\left(-c_2 \frac{\Delta_1^2}{\rho^2} d\right) \|\theta_1^*\|$$

with high probability. Choosing $\gamma = \frac{1}{2cp_1}$, we have

$$\|\theta_1^+ - \theta_1^*\| \le \frac{1}{2} \|\theta_1 - \theta_1^*\| + c_1 \mu \|\theta_1^*\| + c_2 k \gamma \exp\left(-c_3 \frac{\Delta^2}{\rho^2} d\right) \|\theta_1^*\|,$$

with probability at least $1 - C_1 \exp(-C_2 d)$, which proves the theorem.

A.0.1. GOOD INITIALIZATION

We stick to analyzing θ_1^+. In the following lemma, we only consider θ_2. In general, for $\{\theta_3, ..., \theta_k\}$, the total probability of error, using union bound will be $(k-1)$ times the probability of error obtained in the following lemma.

Lemma A.1. *We have*

$$\mathbb{P}\left(F_i(\theta_1) > F_i(\theta_2) | F_i(\theta_1^*) < F_i(\theta_2^*)\right) \le c_1 k \exp\left(-c_2 \frac{\Delta_1^2}{\rho^2} d\right),$$

where $\rho = (\max_{j \in [k]} \|\theta_j^*\|)^2$ *and* $\Delta_1 = \Delta - 2\lambda$.

Proof. Let us first consider the event

$$\{F_i(\theta_1^*) < F_i(\theta_2^*)\} = \left\{(y_i - \langle x_i, \theta_1^* \rangle)^2 < (y_i - \langle x_i, \theta_2^* \rangle)^2\right\}$$

We are interested in the event $\{F_i(\theta_1) < F_i(\theta_2)\}$ since the sets S_1 and S_2 are defined based on this event. In particular we are interested in $\mathbb{P}(F_i(\theta_1) < F_i(\theta_2) | \{F_i(\theta_1^*) < F_i(\theta_2^*)\})$. To that end, we first compute the complement event, given by

$$\mathbb{P}(F_i(\theta_1) > F_i(\theta_2) | \{F_i(\theta_1^*) < F_i(\theta_2^*)\}),$$

Conditioned on $\{F_i(\theta_1^*) < F_i(\theta_2^*)\}$, we compute the probability of the event

$$\{(y_i - \langle x_i, \theta_1 \rangle)^2 > (y_i - \langle x_i, \theta_2 \rangle)^2\}$$
$$= \{(y_i - \langle x_i, \theta_1^* \rangle)^2 + \langle x_i, \theta_1 - \theta_1^* \rangle^2 - 2\langle x_i, \theta_1 - \theta_1^* \rangle (y_i - \langle x_i, \theta_1^* \rangle)$$
$$> (y_i - \langle x_i, \theta_2^* \rangle)^2 + \langle x_i, \theta_2 - \theta_2^* \rangle^2 - 2\langle x_i, \theta_2 - \theta_2^* \rangle (y_i - \langle x_i, \theta_2^* \rangle)\}$$
$$\subseteq \{\langle x_i, \theta_1 - \theta_1^* \rangle^2 - 2\langle x_i, \theta_1 - \theta_1^* \rangle (y_i - \langle x_i, \theta_1^* \rangle)$$
$$> \langle x_i, \theta_2 - \theta_2^* \rangle^2 - 2\langle x_i, \theta_2 - \theta_2^* \rangle (y_i - \langle x_i, \theta_2^* \rangle)\},$$

where we use the fact that $(y_i - \langle x_i, \theta_1^* \rangle)^2 < (y_i - \langle x_i, \theta_2^* \rangle)^2$. Continuing, we have

$$\{(y_i - \langle x_i, \theta_1 \rangle)^2 > (y_i - \langle x_i, \theta_2 \rangle)^2\}$$
$$\subseteq \{\langle x_i, \theta_1 - \theta_1^* \rangle^2 - 2\langle x_i, \theta_1 - \theta_1^* \rangle (y_i - \langle x_i, \theta_1^* \rangle) > t\} \bigcup \{\langle x_i, \theta_2 - \theta_2^* \rangle^2 - 2\langle x_i, \theta_2 - \theta_2^* \rangle (y_i - \langle x_i, \theta_2^* \rangle) \le t\},$$

for any t. In particular, we select $t = |\langle x_i, \theta_1 - \theta_1^* \rangle||\langle x_i, \theta_2 - \theta_2^* \rangle|$. We have

$$\langle x_i, \theta_1 - \theta_1^* \rangle^2 - 2\langle x_i, \theta_1 - \theta_1^* \rangle(y_i - \langle x_i, \theta_1^* \rangle)$$
$$= \langle x_i, \theta_1 - \theta_1^* \rangle^2 + 2\langle x_i, \theta_1^* - \theta_1 \rangle(y_i - \langle x_i, \theta_1^* \rangle)$$
$$\leq |\langle x_i, \theta_1 - \theta_1^* \rangle|^2 + 2|\langle x_i, \theta_1^* - \theta_1 \rangle||y_i - \langle x_i, \theta_1^* \rangle|$$
$$= |\langle x_i, \theta_1 - \theta_1^* \rangle|^2 + 2|\langle x_i, \theta_1 - \theta_1^* \rangle||y_i - \langle x_i, \theta_1^* \rangle|$$

and

$$\langle x_i, \theta_2 - \theta_2^* \rangle^2 - 2\langle x_i, \theta_2 - \theta_2^* \rangle(y_i - \langle x_i, \theta_2^* \rangle)$$
$$\geq |\langle x_i, \theta_2 - \theta_2^* \rangle|^2 - 2|\langle x_i, \theta_2 - \theta_2^* \rangle||y_i - \langle x_i, \theta_2^* \rangle|$$

With this we have,

$$\mathbb{P}\big((y_i - \langle x_i, \theta_1 \rangle)^2 > (y_i - \langle x_i, \theta_2 \rangle)^2 | \mathcal{E}_i\big)$$
$$\leq \underbrace{\mathbb{P}(|\langle x_i, \theta_1 - \theta_1^* \rangle| > |\langle x_i, \theta_2 - \theta_2^* \rangle| - 2|y_i - \langle x_i, \theta_1^* \rangle|)}_{T_1}$$
$$+ \underbrace{\mathbb{P}(|\langle x_i, \theta_2 - \theta_2^* \rangle| < |\langle x_i, \theta_1 - \theta_1^* \rangle| + 2|y_i - \langle x_i, \theta_1^* \rangle|)}_{T_2}.$$

We first analyze T_1. The analysis for the T_2 term follows similarly. We have

$$T_1 = \mathbb{P}(|\langle x_i, \theta_1 - \theta_1^* \rangle| > |\langle x_i, \theta_2 - \theta_2^* \rangle| - 2|y_i - \langle x_i, \theta_1^* \rangle|) \tag{6}$$
$$\leq \mathbb{P}(|\langle x_i, \theta_2 - \theta_2^* \rangle| > \Delta) + \mathbb{P}(|\langle x_i, \theta_1 - \theta_1^* \rangle| > \Delta - 2\lambda), \tag{7}$$

where Δ is the separation as defined in Section 2 and λ is a small parameter defined in.. With this we have

$$T_1 = \mathbb{P}\big(y_i - \langle x_i, \theta_1 \rangle)^2 > (y_i - \langle x_i, \theta_2 \rangle)^2 | \mathcal{E}_i\big)$$
$$= \mathbb{P}\left(\langle x_i, \frac{\theta_2 - \theta_2^*}{\|\theta_2 - \theta_2^*\|} \rangle > \frac{\Delta}{\|\theta_2 - \theta_2^*\|}\right) + \mathbb{P}\left(\langle x_i, \frac{\theta_1 - \theta_1^*}{\|\theta_1 - \theta_1^*\|} \rangle > \frac{\Delta - 2\lambda}{\|\theta_1 - \theta_1^*\|}\right)$$

Now, note that $\theta_2 - \theta_2^*$ and $\theta_1 - \theta_1^*$ depend on x_i; since we are not allowed to have fresh samples at every iteration.

To avoid such dependence we compute an uniform bound using one-step discretization (covering argument). Suppose $u_1, u_2, ..., u_n$ be an ϵ cover of $\mathbb{S}^{d-1}$ in ℓ_2 norm. (Vershynin, 2018) shows that $M \leq (1 + \frac{2}{\epsilon})^d$. Also, suppose u_l is the nearest point of the ϵ cover to $\frac{\theta_2 - \theta_2^*}{\|\theta_2 - \theta_2^*\|}$ and u_r is the nearest point of the ϵ cover to $\frac{\theta_1 - \theta_1^*}{\|\theta_1 - \theta_1^*\|}$. We have

$$= \mathbb{P}\left(\langle x_i, \frac{\theta_2 - \theta_2^*}{\|\theta_2 - \theta_2^*\|} \rangle > \frac{\Delta}{\|\theta_2 - \theta_2^*\|}\right) + \mathbb{P}\left(\langle x_i, \frac{\theta_1 - \theta_1^*}{\|\theta_1 - \theta_1^*\|} \rangle > \frac{\Delta - 2\lambda}{\|\theta_1 - \theta_1^*\|}\right)$$
$$= \mathbb{P}\left(\langle x_i, u_l \rangle + \langle x_i, \frac{\theta_2 - \theta_2^*}{\|\theta_2 - \theta_2^*\|} - u_l \rangle > \frac{\Delta}{\|\theta_2 - \theta_2^*\|}\right)$$
$$+ \mathbb{P}\left(\langle x_i, u_r \rangle + \langle x_i, \frac{\theta_1 - \theta_1^*}{\|\theta_1 - \theta_1^*\|} - u_r \rangle > \frac{\Delta - 2\lambda}{\|\theta_1 - \theta_1^*\|}\right)$$
$$= \mathbb{P}\left(\langle x_i, u_l \rangle > \frac{\Delta}{\|\theta_2 - \theta_2^*\|} - \epsilon\right) + \mathbb{P}\left(\langle x_i, u_r \rangle > \frac{\Delta - 2\lambda}{\|\theta_1 - \theta_1^*\|} - \epsilon\right)$$
$$\leq \mathbb{P}\left(\langle x_i, u_l \rangle > \frac{\Delta \sqrt{d}}{\tilde{c}\|\theta_2^*\|} - \epsilon\right) + \mathbb{P}\left(\langle x_i, u_r \rangle > \frac{(\Delta - 2\lambda)\sqrt{d}}{\tilde{c}_1\|\theta_1^*\|} - \epsilon\right)$$

where we use the fact that $\|x_i\| \leq 1$, the definition of an ϵ cover and the initialization condition.

We now use the fact that, x_i is centered sub-Gaussian (with constant parameter) random variables. Note that since

$$\mathbb{P}(F_i(\theta_1^*) < F_i(\theta_2^*) = p_1 \geq \min\{p_1, ..., p_k\} > \bar{c},$$

where $\bar{c}$ is a constant, we invoke (Yi et al., 2016, Lemma 15) to obtain

$$\mathbb{P}\left(\langle x_i, u_l\rangle > \frac{\Delta\sqrt{d}}{\tilde{c}\|\theta_2^*\|} - \epsilon\right) \leq \exp\left(-c(\frac{\Delta\sqrt{d}}{\tilde{c}\|\theta_2^*\|} - \epsilon)^2\right),$$

and

$$\mathbb{P}\left(\langle x_i, u_r\rangle > \frac{(\Delta - 2\lambda)\sqrt{d}}{\tilde{c}_1\|\theta_1^*\|} - \epsilon\right) \leq \exp\left(-c(\frac{(\Delta - 2\lambda)\sqrt{d}}{\tilde{c}\|\theta_1^*\|} - \epsilon)^2\right),$$

We now take union bound over $(1+\frac{2}{\epsilon})^d$ entries, and finally substituting ϵ as a sufficiently small constant, we have

$$\mathbb{P}\left(\langle x_i, \frac{\theta_2 - \theta_2^*}{\|\theta_2 - \theta_2^*\|}\rangle > \frac{\Delta}{\|\theta_2 - \theta_2^*\|}\right) \leq c_1 \exp\left(-c_2 \frac{\Delta^2}{\|\theta_2^*\|^2} d\right),$$

and

$$\mathbb{P}\left(\langle x_i, \frac{\theta_1 - \theta_1^*}{\|\theta_1 - \theta_1^*\|}\rangle > \frac{\Delta - 2\lambda}{\|\theta_1 - \theta_1^*\|}\right) \leq c_1 \exp\left(-c_2 \frac{(\Delta - 2\lambda)^2}{\|\theta_1^*\|^2} d\right).$$

Note that similar analysis holds for the term T_2, since we use the symmetric sub-Gaussian concentration. Hence, we finally obtain,

$$\mathbb{P}\left(F_i(\theta_1) > F_i(\theta_2) | F_i(\theta_1^*) < F_i(\theta_2^*)\right) \leq C \exp\left(-c \frac{(\Delta - 2\lambda)^2}{(\max_{j\in[k]}\|\theta_j^*\|)^2} d\right),$$

and

$$\mathbb{P}\left(F_i(\theta_1) < F_i(\theta_2) | F_i(\theta_1^*) < F_i(\theta_2^*)\right) \geq 1 - C \exp\left(-c \frac{(\Delta - 2\lambda)^2}{(\max_{j\in[k]}\|\theta_j^*\|)^2} d\right),$$

which concludes the proof. □

Lemma A.2. *Conditioned on $i \in S_1 \cap S_1^*$, the distribution of x_i is sub-Gaussian with a constant parameter.*

Proof. Let us first consider the following probability

$$\begin{aligned}
&\mathbb{P}(i \in S_1 \cap S_1^*) \\
&= \mathbb{P}(F_i(\theta_1) < F_i(\theta_2), F_i(\theta_1^*) < F_i(\theta_2^*)) \\
&= \mathbb{P}(F_i(\theta_1) < F_i(\theta_2) | F_i(\theta_1^*) < F_i(\theta_2^*)) \mathbb{P}(F_i(\theta_1^*) < F_i(\theta_2^*)) \\
&\geq p_1 \left[1 - C \exp\left(-c \frac{(\Delta - 2\lambda)^2}{(\max_{j\in[k]}\|\theta_j^*\|)^2} d\right)\right] \\
&\geq \tau
\end{aligned}$$

where τ is a constant. Here we use Assumption 4.1 and the fact that $\min\{p_1, p_2\} > \bar{c}$ where $\bar{c}$ is a constant. Since $\mathbb{P}(i \in S_1 \cap S_1^*)$ is greater than a constant, using (Yi et al., 2016, Lemma 15), we conclude that x_i is sub-Gaussian with constant parameter. □

B. Proofs from Section 3

We now provide the proof of Theorem 3.1.

Proof of Theorem 3.1. We begin by the following chain,

$$\frac{1}{n}\mathbb{E}_{\boldsymbol{\sigma}}\left[\sup_{\bar{h}\in\bar{\mathcal{H}}}\sum_{i=1}^n \sigma_i \mathcal{L}(y_i, \bar{h}(x))\right] = \frac{1}{n}\mathbb{E}_{\sigma_1,\cdots,\sigma_{n-1}}\left[\mathbb{E}_{\sigma_n}\left[\sup_{\bar{h}\in\bar{\mathcal{H}}} u_{n-1}(\bar{h}) + \sigma_n \mathcal{L}(y_n, \bar{h}(x_n))\right]\right],$$

where $u_n(\bar{h}) = \sum_{i=1}^{n-1} \sigma_i \mathcal{L}(y_i, \bar{h}(x_i))$. By definition of the supremum, for any $\epsilon > 0$, there exist $\bar{h}_a$ and $\bar{h}_b$ such that,

$$u_{n-1}(\bar{h}_a) + \mathcal{L}(y_n, \bar{h}_a(x_n)) \geq (1-\epsilon)\left[\sup_{\bar{h}} u_{n-1}(\bar{h}) + \mathcal{L}(y_n, \bar{h}(x_n))\right]$$

$$u_{n-1}(\bar{h}_b) - \mathcal{L}(y_n, \bar{h}_b(x_n)) \geq (1-\epsilon)\left[\sup_{\bar{h}} u_{n-1}(\bar{h}) - \mathcal{L}(y_n, \bar{h}(x_n))\right]$$

By expanding the definition of the expectation we can then prove that,

$$(1-\epsilon)\mathbb{E}_{\sigma_n}\left[\sup_{\bar{h}} u_{n-1}(\bar{h}) + \sigma_n \mathcal{L}(y_n, \bar{h}(x_n))\right] \leq \frac{1}{2}\left(u_{n-1}(\bar{h}_a) + \mathcal{L}(y_n, \bar{h}_a(x_n))\right) + \frac{1}{2}\left(u_{n-1}(\bar{h}_b) - \mathcal{L}(y_n, \bar{h}_b(x_n))\right).$$

Let $s_l = \text{sign}(\bar{h}_a(x_n)_l - \bar{h}_b(x_n)_l)$. Then we have the following chain,

$$(1-\epsilon)\mathbb{E}_{\sigma_n}\left[\sup_{\bar{h}} u_{n-1}(\bar{h}) + \sigma_n \mathcal{L}(y_n, \bar{h}(x_n))\right]$$

$$\leq \frac{1}{2}\left(u_{n-1}(\bar{h}_a) + u_{n-1}(\bar{h}_b) + \mathcal{L}(y_n, \bar{h}_a(x_n)) - \mathcal{L}(y_n, \bar{h}_b(x_n))\right)$$

$$= \frac{1}{2}\left(u_{n-1}(\bar{h}_a) + u_{n-1}(\bar{h}_b) + \min_j \ell(y_n, \bar{h}_a(x_n)_j) - \min_l \ell(y_n, \bar{h}_b(x_n)_l)\right)$$

$$\overset{(a)}{\leq} \frac{1}{2}\left(u_{n-1}(\bar{h}_a) + u_{n-1}(\bar{h}_b) + \ell(y_n, \bar{h}_a(x_n)_{l^*}) - \ell(y_n, \bar{h}_b(x_n)_{l^*})\right)$$

$$\leq \frac{1}{2}\left(u_{n-1}(\bar{h}_a) + u_{n-1}(\bar{h}_b) + \max_l(\ell(y_n, \bar{h}_a(x_n)_l) - \ell(y_n, \bar{h}_b(x_n)_l))\right)$$

$$\leq \frac{1}{2}\left(u_{n-1}(\bar{h}_a) + u_{n-1}(\bar{h}_b) + \sum_l(\ell(y_n, \bar{h}_a(x_n)_l) - \ell(y_n, \bar{h}_b(x_n)_l))\right)$$

$$\leq \frac{1}{2}\left(u_{n-1}(\bar{h}_a) + u_{n-1}(\bar{h}_b) + \sum_l \mu s_l(\bar{h}_a(x_n)_l - \bar{h}_b(x_n)_l)\right)$$

$$= \sum_{l=1}^{k} \frac{1}{2}\left(u_{n-1}(\bar{h}_a)/k + \mu s_l \bar{h}_a(x_n)_l\right) + \frac{1}{2}\left(u_{n-1}(\bar{h}_b)/k - \mu s_l \bar{h}_b(x_n)_l\right)$$

$$\leq \sum_l \mathbb{E}_{\sigma_{n,l}}\left[\sup_{\bar{h} \in \mathcal{H}} \frac{u_{n-1}(\bar{h})}{k} + \sigma_{n,l} \bar{h}(x_n)_l\right]$$

$$= k\mathbb{E}_{\sigma_n}\left[\sup_{\bar{h} \in \mathcal{H}} \frac{u_{n-1}(\bar{h})}{k} + \mu \sigma_n h_j(x_n)\right].$$

where $\sigma_{n,l}$ are Rademacher R.Vs. The result follows by induction and symmetry between the mixture coordinates of $\bar{h}(\cdot)$. In (a), l^* is the coordinate at which the first minima is attained. $\square$

C. Proofs from Section 4.3

We first start by proving that for any of the optimal subsets of the the original samples if we further sub-sample a smaller partition, then the regression solution in both the sets are not too far from each other. The result follows from an application of random design regression analysis in (Hsu et al., 2011).

Lemma C.1. *For any $i \in [k]$, let $B = B_i^*$. Suppose P samples are chosen i.i.d uniformly at random from B. Let θ_B and θ_P be the least squares solution for B and P respectively. Then we have that,*

$$\|\theta_B - \theta_P\|_{\Sigma_B}^2 = O\left(\frac{R(\sqrt{d} + \sqrt{\log(1/\delta)})^2 B_d}{|P|\sqrt{\lambda_{\min}(\Sigma_B)}} + \frac{4(b+wR)^2(\sqrt{d} + \sqrt{\log(1/\delta)})^2}{|P|} + \frac{\gamma^2 d^2 \log^2(1/\delta)}{|P|^2}\right),$$

with probability at least $1-\delta$.

Proof. The samples in P form a random design regression problem where,

$$\mathbb{E}[xy] = \frac{1}{|B|}\sum_{i \in B} x_i y_i$$

$$\mathbb{E}[xx^T] = \frac{1}{|B|}\sum_{i \in B} x_i x_i^T =: \Sigma_B.$$

We will now verify the conditions of Theorem 2 in (Hsu et al., 2011). To avoid cumbersome notation let $\theta = \theta_B$. Consider the decomposition of the regression target and covariates,

$$y = \theta^T x + \text{bias}(x) + \eta(x)$$

where $\eta(x)$ is the noise. The terms are defined as,

$$\text{bias}(x) = E[y|x] - \theta^T x \text{ and } \eta(x) = y - E[y|x]$$

Condition 1: Note that the noise is bounded,

$$|\eta(x)| \leq |y| + |\theta^T x| + |\text{bias}(x)| \leq 2(b + wR).$$

Therefore the noise is sub-gaussian with parameter $\sigma_{noise} \leq 2(b + wR)$.

Condition 2: The bounded bias is just from our assumption directly. We have the following,

$$\left\| \Sigma_B^{-1} \text{bias}(x)x \right\| \leq \gamma\sqrt{d}. \tag{8}$$

Condition 3: For any u s.t $\|u\| = 1$, we have that

$$u^T \Sigma_B^{-1/2} x \leq \frac{R}{\sqrt{\lambda_{min}(\Sigma_B)}}. \tag{9}$$

Thus, $\rho_{(1,cov)} = \frac{R}{\sqrt{\lambda_{min}(\Sigma_B)}}$. Thus all the conditions are valid and we can apply Theorem 2 in (Hsu et al., 2011). Thus finally with probability $1 - \delta$ we get,

$$\|\theta_P - \theta_B\|_{\Sigma_B}^2 = O\left(\frac{\rho_{(1,cov)}(\sqrt{d} + \sqrt{\log(1/\delta)})^2 \mathbb{E}[\text{bias}(x)^2]}{|P|} + \frac{\sigma_{noise}^2(\sqrt{d} + \sqrt{\log(1/\delta)})^2}{|P|} + \frac{\gamma^2 d^2 \log^2(1/\delta)}{|P|^2} \right).$$

$\square$

Now we show that the above result also implies that the respective average errors are also close to each other.

Lemma C.2. *Consider the setting in Lemma C.1. Then we have the following bound,*

$$\mathcal{E}_{ls}(B, \theta_P) - \mathcal{E}_{ls}(B, \theta_B) \leq 2|B|(wR + b)\|\theta_B - \theta_P\|_{\Sigma_B}.$$

Proof. We should note that,

$$\sum_{i \in B}(y_i - \theta_P x_i)^2 - (y_i - \theta_B x_i)^2 \leq |(\theta_P - \theta_B).x_i||(2y_i - \theta_B.x_i - \theta_P.x_i)|$$

$$:= \gamma_i|(\theta_P - \theta_B).x_i| \leq \sqrt{\left(\sum \gamma_i^2\right)\left(\sum((\theta_P - \theta_B).x_i)^2\right)} \leq 2|B|(wR + b)\|\theta_B - \theta_P\|_{\Sigma_B}.$$
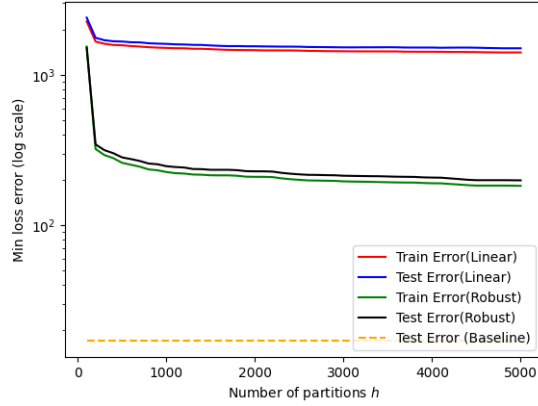
$\square$

Figure 1. Comparison of train error and test error (min-loss) of Algorithms 3 (with number of random partitions h) when we use a linear model/robust linear model to fit each part of partition.

Now we are at a position to prove our main theorem.

Proof of Theorem 4.7. Let $\{\tilde{P}_i\}$ be the partition of A that conforms to the partition $\{B_i^*\}$. Then by Hoeffding's bound we have w.p $1-\delta$

$$||\tilde{P}_i|-\alpha_i|A||\leq \sqrt{\alpha_i(1-\alpha_i)\log(k/\delta)|A|}.$$

Let us assume that n is such that $|A|\geq \frac{4(1-\alpha_i)}{\alpha_i}\log(k/\delta)$, for all i.

Then by Lemma C.2 we have that,

$$\sum_{i=1}^{k}\mathcal{E}_{ls}(B_i^*,\theta_{\tilde{P}_i})-\mathcal{E}_{ls}(B_i^*,\theta_{B_i^*})\leq 2(wR+b)\sum_{i=1}^{k}|B_i^*|\left\|\theta_{B_i^*}-\theta_{\tilde{P}_i}\right\|_{\Sigma_{B_i^*}}.$$

Thus w.p at least $1-2\delta$ we have,

$$\frac{1}{n}\sum_{i=1}^{k}\mathcal{E}_{ls}(B_i^*,\theta_{\tilde{P}_i})\leq \frac{1}{n}\sum_{i=1}^{k}\mathcal{E}_{ls}(B_i^*,\theta_{B_i^*})$$
$$+O\left(\sum_{i=1}^{k}\alpha_i(wR+b)\sqrt{\frac{R(\sqrt{d}+\sqrt{\log(k/\delta)})^2 B_d}{\alpha_i|A|\sqrt{\lambda_{\min}}}+\frac{(wR+B)^2(\sqrt{d}+\sqrt{\log(k/\delta)})^2}{\alpha_i|A|}}\right)$$

Now note that $\mathcal{E}(\{P_i^m\})$ in Algorithm 2 is less than the LHS in the above equation. Therefore, the loss achieved by the solution of Algorithm 2 can only be less than the RHS of the above equation. $\square$

The above result means that an ϵ additive approximation is obtained when,

$$|A|=O\left(\frac{1}{\epsilon^2}k^2\frac{\alpha}{\lambda_{\min}}\left(d+\log\frac{k}{\delta}\right)\right)$$

w.p at least $1-2\delta$.

σ	10	50	100	200
Train Error	15.65	96.44	1350.22	2408.80
Test Error	15.35	94.35	1341.29	2380.27

Table 4. Train/Test error (Min loss) of Algorithm 1 for different initializations after 40 iterations

Dataset	LR	Alg 1 (Mean, Var)	Alg 3 (Mean, Var)
A	22.57	(20.37,10.02)	(12.01,0.21)
B	18806	N/A	(5105, 51537)
C	15.34	N/A	(7.24,0.16)

Table 5. Mean (Mean) and Variance (Var) of min-loss on Training data generated from non-linear synthetic datasets. LR corresponds to a simple linear regression model and N/A implies that the algorithm did not converge in any implementation.

u	v	Train Data	Test Data
'1010'	'2116'	836	209
'752'	'1941'	1002	251
'752'	'2116'	827	207
'752'	'2909'	844	212
'1010'	'4725'	818	205

Table 6. Size of train and test data for users (u,v)

D. More experiments

Mixture of linear datasets: In this experiment, we create two datasets (X_1, y_1) (using `sklearn.datasets.makeregression` with 4 features, standard deviation of noise = 4.0 and bias = 100.0) and (X_2, y_2) (using `makeregression` module from `sklearn.datasets` with 4 features, standard deviation of noise = 4.0 and bias = 0.0) generated according to two distinct noisy linear models. Subsequently we combine the datasets and shuffle the datapoints to create a single one (X,y). We partition the dataset (X,y) intro training data (3200 datapoints) and test data (800 datapoints). We first design a baseline model as if an oracle provides us the information of which datapoint in (X,y) was generated according to which mode. In particular we fit two distinct linear regression models β_1,β_2 on the training data restricted to (X_1,y_1) and (X_2,y_2); subsequently, we predict on the test data using the two linear regression models and compute the min-loss. Using this model, the mean-squared training error is 15.256 and the mean min-loss error on the test data is 17.013. Next, we compare the performance (min-loss on the train and test data) of Algorithm 3 (averaged over 100 implementations and $|A| = 150$) which is depicted in Figure 1. In Figure 1, we empirically compare the two possibilities namely the linear model and the robust linear model on each part of the partition. It is empirically quite evident that Algorithm 3 when used with the robust regression model achieves a significantly better performance as compared to Algorithm 3 used with the linear model. We also implement the AM algorithm (see Algorithm 1) on this dataset. Here, we initialize $\theta_1^{(0)},\theta_1^{(1)}$ by $\beta_1 + \eta_1$ and $\beta_2 + \eta_2$ where η_1,η_2 are random vectors where each entry is generated independently according to a Gaussian with zero mean and variance σ^2. We vary σ and after 40 iterations, we compare the min-loss error (averaged over 15 implementations with $\gamma = 0.1$) on the train and test datasets. We provide these results in Table 4.

More on other experiments: We provide some missing figures and other details from the experiments here. Table 5 provides the training set min-losses for all the non-linear synthetic datasets. Table 6 provides the datasets statistics for the movie lens dataset that we use.