Conditional Linear Regression

Diego Calderon UIUC
Brendan Juba Washington U. St. Louis
Sirui Li MIT
Zongyi Li Caltech
Lisa Ruan Harvard

Abstract

Work in machine learning and statistics commonly focuses on building models that capture the vast majority of data, possibly ignoring a segment of the population as outliers. However, there may not exist a good, simple model for the distribution, so we seek to find a small subset where there exists such a model. We give a computationally efficient algorithm with theoretical analysis for the conditional linear regression task, which is the joint task of identifying a significant portion of the data distribution, described by a k-DNF, along with a linear predictor on that portion with a small loss. In contrast to work in robust statistics on small subsets, our loss bounds do not feature a dependence on the density of the portion we fit, and compared to previous work on conditional linear regression, our algorithm's running time scales polynomially with the sparsity of the linear predictor. We also demonstrate empirically that our algorithm can leverage this advantage to obtain a k-DNF with a better linear predictor in practice.

1 Introduction

Linear regression is a standard tool of statistical analysis. While the standard linear regression task seeks to model the majority of the data, we consider problems where a linear predictor can only be accurate for a small portion of the data distribution. We will consider cases in which the linear predictor is accurate on a conditional distribution described by some simple condition. Note that neither the condition nor the linear rule is known in advance. To illustrate our problem, consider a set of patient data from a hospital that includes multiple continuous factors, such as rate of smoking, radiation exposure, etc. Assume we want to predict risk of developing lung cancer. There may be no linear rules that can model, e.g., the risk of developing lung cancer for the majority of the data. However, there may be some linear model that fits a specific subset of the data well, such as adult city-dwellers. If such a model exists, we aim to find it together with the description of the corresponding condition. Our focus is on identifying the portions of the population for which such simply structured models succeed in making accurate predictions, even when such models do not exist for most of the population. This problem was introduced by Juba (2017), who gave an algorithm for conditional linear regression under the ℓ_∞ loss where the predictor factors are sparse (i.e., its time and data requirements are exponential in the number of regression factors), and an algorithm for the general case that only identifies a condition describing a small fraction of the optimal condition. The former, sparse algorithm was extended to general ℓ_p losses by Hainline et al. (2019).

Our contribution We give an algorithm for the ℓ_2 loss that, under some mild regularity assumptions,

1. Only uses polynomial time and data in the dimension and number of factors.
2. Recovers a condition that covers as much of the distribution as the optimal condition.
3. Approximates an optimal t-term k-DNF with a $\tilde{\mathcal{O}}(t \log\log n)$-factor blow-up of the loss.
4. Is faster, especially for non-sparse problems, compared to the previous algorithm by Hainline et al.

Our algorithm builds on the *list-learning algorithms* due to Charikar et al. (2017). That work aimed to learn about arbitrary small subsets of the data by producing a list of parameter values containing estimates of the parameters for any small subset. Charikar et al. could only address tasks such as mean estimation, and as they discuss, could not obtain sufficient accuracy to use their framework for linear regression—their algorithm returns uninformative estimates of the regression parameters. Our algorithm, like theirs, it-

Proceedings of the 23rdInternational Conference on Artificial Intelligence and Statistics (AISTATS) 2020, Palermo, Italy. PMLR: Volume 108. Copyright 2020 by the author(s).

eratively computes local estimates for the regression parameters with consideration of their neighbors, and then (re)clusters the terms using their corresponding parameters. Our primary innovation lies in using a fixed family of subsets ("terms") as the basic units of data as opposed to individual points, which we leverage to obtain adequate estimates of the regression parameters. Specially, our improvements upon the work of Charikar et al. (2017) are:

1. We modify Charikar et al's algorithm and analysis to *operate on sets instead of points, by introducing weights and modifying the definition of neighbors.*
2. In our algorithm, *we show that it does not lose good sets in any iteration*, in contrast to Charikar et al's algorithm which indeed may lose a small fraction of good points. Consequently, whereas the original algorithm may need to terminate with a inaccurate estimate of the parameters, *we can potentially reach any accuracy.*
3. We introduce a covering algorithm in the end to *extract a small k-DNF condition.*

We stress that Charikar et al. (and subsequent works such as (Diakonikolas et al., 2018)) cannot obtain a linear predictor with loss that scales with the loss of the best linear predictor on the data on account of a difference in the formulation: Charikar et al. and Diakonikolas et al. consider arbitrary subsets of the data, whereas we only consider subsets described by k-DNFs. Diakonikolas et al. show that even for the simpler problem of mean estimation, one can only guarantee loss that scales polylogarithmically with the density of the set for which we estimate the mean. Some very recent works (Karmalkar et al., 2019; Raghavendra and Yau, 2020) solve a list-learning linear regression task, that could be used to solve our problem, but these algorithms have a running time that is exponential in a high-order polynomial of $1/\mu$, where μ is the size (fraction) of target subset in the whole data set. Even for $\mu = 1/2$, these algorithms are essentially infeasible to run in practice. By contrast, our algorithms have a running time that only depends polynomially on $1/\mu$, and we demonstrate their feasibility on some standard benchmark data sets.

Related Work Our problem is similar in spirit to work in *robust statistics* (Huber, 1981; Rousseeuw and Leroy, 1987), with the key distinction that apart from the list-learning works discussed above, robust statistics assumes that the outliers comprise a minority of the data. By contrast here, the majority of the data may be "outliers." Another setting in this vein is *learning with rejection* in which the predictor has the option to "abstain" from making a prediction. In most cases, the strategy for deciding when to abstain is based on some measure of "confidence" of the prediction—for example, this is how El-Yaniv and Wiener (2012) addressed a linear regression task. The difference is that these methods do not generally produce a "nice" description of the region on which they will make a prediction. On the other hand, Cortes et al. (2016) considered a version of the task in which the prediction region is constrained to come from a fixed family of nice rules, like us. The difference is that Cortes et al. do not seek to achieve given rates of coverage or error, but rather posit that abstaining from prediction has a known, fixed cost relative to the cost of an error, and seek to minimize this overall cost. Finally, algorithms such as RANSAC (Fischler and Bolles, 1981) similarly find a dense linear relation among a subset of the points when one exists, but these algorithms scale exponentially with the dimension, and like learning with rejection, do not obtain a rule characterizing which points will satisfy the linear relationship. There are many other works that fit the data using multiple linear rules, such as linear mixed models (McCulloch and Searle, 2001; Jiang, 2007), segmented regression, regression trees (Quinlan, 1992), cluster-wise linear regression (Park et al., 2017), or piecewise linear regression. These methods are similar in that the portion of the data fit by an individual linear rule may be small. The distinction is generally that they seek to model the entire data distribution with linear rules, i.e., they cluster the data and minimize the total loss over all clusters. By contrast, we only seek a small fraction (say, 10%) of the data where a good linear rule exists. We also stress that unlike some of these methods, we find a formula describing the portion of the distribution we fit.

2 Problem Formulation and Results

We suppose we have data consisting of N examples, where each example has three kinds of attributes: a vector of n Boolean attributes $\mathbf{x}$, a vector of d real-valued attributes $\mathbf{y}$, and a real-valued attribute z that we wish to predict. For example, in our cancer prediction setting, we have an example $(\mathbf{x}, \mathbf{y}, z)^{(i)}$ for each ith patient in which:

- $\mathbf{x}^{(i)}$ is a vector of Boolean demographic properties that describe patient i (e.g., adult, city dweller)
- $\mathbf{y}^{(i)}$ is vector of continuous risk factors for patient i, (e.g., rate of smoking, radiation exposure)
- $z^{(i)}$ might be the probability of developing a certain kind of cancer in the next ten years.

When there is no confusion, we will use $\mathbf{x}^{(i)}$ to denote the whole point $(\mathbf{x}, \mathbf{y}, z)^{(i)}$. Note that we can obtain Boolean attributes $\mathbf{x}$ from continuous attributes $\mathbf{y}$ by using binning and splits, similar to decision trees. For

example, we can define a new Boolean attribute like $x =$ "whether $y \leq a$ " for some quantile of the data a, which we use in the experiments. Similarly, Boolean variables can also be used as regression factors.

We wish to find a linear rule $\langle \mathbf{w}, \mathbf{y} \rangle$ to predict z. We would typically achieve this by minimizing the loss $\|\langle \mathbf{w}, \mathbf{y} \rangle - z\|_2$ averaged over the data. But, it's common that there doesn't exist a good linear rule for the whole data set. We propose to find a subset of the data (i.e. a condition $\mathbf{c}$), such that there exists a good fit on $\mathbf{c}$. Of course, if we just pick any subset that fits some linear rule well, this is unlikely to be predictive. Instead, we will pick out a subset described by some simple conditions which will use the Boolean $\mathbf{x}$ attributes. Following the previous work (Juba, 2017), we will only consider conditions represented by k-DNF (Disjunctive normal form) formulas, which are an "or" of *terms*, where each term is an "and" of at most k attributes (which we permit to be negated). Thus: $\mathbf{c} = t_1 \vee \ldots \vee t_s$ for some s, where $t_i = \ell_{i,1} \wedge \ldots \wedge \ell_{i,k}$, and where $\ell_{i,j}$ is either $x_{i,j}$ or $\neg x_{i,j}$, for some attribute $x_{i,j}$. And, when we say a t-term DNF, we mean $s \leq t$. For example, a term could be "patients who are adult and female"; a DNF could be "patients who are either (adult and female) or (not smokers and male)". We focus on k-DNF conditions since the use of other natural representations results in an intractable problem (Juba, 2017). So, we want to find such a k-DNF condition that there exists an accurate linear predictor in its conditional distribution, and that is true somewhat often. We will demand that data satisfies it with probability at least μ. Since the task is thus to choose an appropriate set of terms (defining a k-DNF), we can view the terms as m atomic sets of data.[1] A table of notation is included in the supplemental material.

Definition 2.1 (Conditional Linear Regression) *Given data* $\{(x_1^{(i)}, \ldots, x_n^{(i)}, y_1^{(i)}, \ldots, y_d^{(i)}, z^{(i)})\}_{i=1}^N$ *drawn i.i.d. from a distribution D, the task is to find a k-DNF condition $\mathbf{c}$ and parameters $\mathbf{w} = (w_1, \ldots, w_d)$, such that the loss $\|\langle \mathbf{w}, \mathbf{y} \rangle - z\|$ is bounded on $D|\mathbf{c}$ and* $\Pr[\mathbf{c}(\mathbf{x}) = 1] \geq \mu$.

We will describe an algorithm that finds a pair $(\hat{\mathbf{c}}, \hat{\mathbf{w}})$ that is close to the optimal solution $(\mathbf{c}^*, \mathbf{w}^*)$, given that the data distribution on $\mathbf{c}^*$ is sufficiently nice in the following sense. We assume first that $\mathbf{w}^*$ gives a predictor with subgaussian residuals on $\mathbf{c}^*$. Second, that our loss function is Lipschitz. Third, that how much the covariances of the desired conditional distribution on $\mathbf{c}^*$ varies across its significant terms is bounded in a spectral sense, captured by the parameter S_{β_0} (defined later). If the distribution on this subset is independent of which term of $\mathbf{c}^*$ is satisfied, then S_{β_0} is 0. If the distribution across these terms is very different from one another, it will be harder for our algorithm to identify that they should be fit by a common linear rule. We will need that this quantity is sufficiently small relative to the degree of (strong) convexity κ of the loss function. (Note that we will see at least that our algorithm succeeds on some standard benchmark data sets in practice.) Generally, for the squared error loss on a bounded space, the convexity coefficient can be viewed as a constant, when the bound $\|\mathbf{y}\|_2 \leq B$ is fixed. Our main theorem is:

Theorem 2.2 *Suppose D is a joint distribution over $\mathbf{x} \in \{0,1\}^n, \mathbf{y} \in \mathcal{B} \subset \mathbb{R}^d$ and $z \in \mathbb{R}$.* **If** *there exists a (ideal) t-term condition $\mathbf{c}^*$ out of m possible terms and parameters $\mathbf{w}^* \in \mathcal{H} \subset \mathbb{R}^d$, where $\mathcal{B}$ has l_2 radius B and $\mathcal{H}$ has l_2 radius r, such that:*

1. $\mathbb{E}_D[(\langle \mathbf{w}^*, \mathbf{y} \rangle - z)^2 | \mathbf{c}^*(\mathbf{x}) = 1] \leq \epsilon$.
2. $Pr[\mathbf{c}^*(\mathbf{x}) = 1] \geq \mu$,

and we have the regularities:

1. *the error $(\langle \mathbf{w}^*, \mathbf{y} \rangle - z)$ follows a σ-subgaussian distribution on $D|\mathbf{c}^*$*
2. *the loss function $f(\mathbf{w}, (\mathbf{y}, z)) = (\langle \mathbf{w}, \mathbf{y} \rangle - z)^2$ is L-lipschitz and κ-strongly convex, where $\kappa \geq \Omega\big(tS_{\frac{\gamma}{t}0} \log \frac{1}{\mu} / \sqrt{\mu}\big)$,*

then *for any $\delta, \gamma \in (0,1)$, using $N = \mathcal{O}(\frac{B^6 d^3 \sigma^2 L^2 t^2}{\mu \gamma^4} \log(m/\delta))$ examples, we can find a $\mathcal{O}(t \log \mu N)$-term condition $\hat{\mathbf{c}}$ and parameters $\hat{\mathbf{w}}$ in polynomial time, such that with probability $(1-\delta)$:*

1. $\mathbb{E}_D[(\langle \hat{\mathbf{w}}, \mathbf{y} \rangle - z)^2 | \hat{\mathbf{c}}(\mathbf{x}) = 1] \leq \mathcal{O}(t \log(\mu N)(\epsilon + \gamma))$
2. $Pr[\hat{\mathbf{c}}(\mathbf{x}) = 1] \geq (1-\gamma)\mu$.

The Lipschitz condition follows from the bound on y and w, $L \leq rB^2$, and the number of good terms t is always at most the total number of possible terms $m \leq n^k$, so $\mathcal{O}(t \log(\mu N)\epsilon) \leq \tilde{\mathcal{O}}(n^k \epsilon)$ (suppressing the other parameters). Also, we will be able to enforce that our loss function is at least κ-strongly convex by adding a regularization term, $\frac{\kappa}{2}\|\mathbf{w}\|_2^2$, at some possible cost to solution quality.

We note that some dependence on the number of terms/Boolean attributes in the error seems unavoidable for polynomial-time algorithms. Juba (2017) showed that algorithms for conditional linear regression (w.r.t. ℓ_p losses) yield algorithms for agnostic learning of the family of rules used for conditions with respect to a particular model that treats false-positives and false-negatives differently. The current state of the art for learning this model achieves roughly a $O(n^{k/2})$ blow-up for general k-DNFs (Zhang et al., 2017), or a

[1] Indeed, we will not use the structure of the terms and we could work instead with an arbitrary family of m such atomic subsets of the data, as long as we can collect enough data relative to the number of subsets.

$O(t \log\log n)$ blow-up for t-term k-DNFs (Juba et al., 2018). Indeed, we adapt the approach of Juba et al. (2018) to obtain the analogous guarantee here. (We note that for the standard error model, the state of the art is essentially a $O(n^{k/3})$ blow-up (Awasthi et al., 2010).) Note that for interpretabilty, cases where the DNF is small are of the most interest.

3 Algorithm

Our algorithm works primarily on terms: we consider the terms $\{t_j\}_{j=1}^m$ to be atomic sets of data, whose weights $|t_j|$ are the number of points (or probability mass) satisfying the terms. The ideal condition $\{x : \mathbf{c}^*(\mathbf{x}) = 1\}$ is denoted by I_{good}; we also use I_{good} to denote the collection of terms of the DNF $\mathbf{c}^*$: $\{t_i : t_i \text{ is a term of } \mathbf{c}^*\}$, so the number of terms in I_{good} is t. From the perspective of Charikar et al. (2017), we treat I_{good} as our "good data," with the other points being arbitrary bad data. The algorithm computes regression parameters $\mathbf{w}_j$ for each term t_j, and clusters the $\mathbf{w}_j$. The parameters are iteratively recomputed in a space centered on each of the clusters, improving the quality of our estimates. Eventually, our algorithm suggests a list of candidate parameters $\hat{\mathbf{w}}$, with one of them approximating $\mathbf{w}^*$. Using the residuals of each $\hat{\mathbf{w}}$ as labels, we learn a corresponding $\hat{\mathbf{c}}$ and evaluate its quality. Towards realizing this strategy, we need to compute approximations to the regression parameters that are not too impacted by the presence of terms outside the desired DNF. A table of notation, some illustrations, and all proofs are included in the **supplemental material**.

3.1 Reformulation in term-wise losses

Our algorithms will assume the terms are disjoint and that we have an adequate number of examples to estimate the loss on each term. We will ensure these properties by introducing duplicate points and deleting terms that are satisfied by too few examples.

Given N data points and m terms $t_1, \ldots, t_m$, if we view terms as sets, our analysis will require these terms to be disjoint. A simple method is to duplicate the points for each term they are contained in. After duplication, the terms are disjoint, and there will be at most Nm points. We denote the resulting number of points by N'. The size of I_{good}, changing from $|\bigcup_{I_{good}} t_i|$ to $\sum_{I_{good}} |t_i|$, may also blow up with a factor ranging from 1 to t. Note that the proportion of good points N_{good}/N decreases by at most a factor of $1/m$ since $N'_{good}/N' \geq N_{good}/mN$. This double counting process may skew the empirical distribution of I_{good} by up to a factor of t. For convenience, we will use the same notation N, I_{good}, and μ for both before and after duplication when there is no confusion.

The approach of Charikar et al. (2017) can only guarantee satisfactory estimates of the parameters for sufficiently large subsets of the data. Intuitively, this is not a significant limitation as if a term has very small size, it will not contribute much to our empirical estimates. Indeed, with high probability, the small terms (terms with size $< \beta\mu N$ for $\beta \leq \gamma/t$) only comprise a γ fraction of I_{good}. Based on this motivation, if a term has size less than $\beta\mu N$, then we just delete it at the beginning. As before, we use t and m for the number of terms when there is no confusion.

Given N data points and m disjoint sets (terms) $t_1, \ldots, t_m$ with sizes (weights) $|t_1|, \ldots, |t_m|$, we can define a loss function for each point in the space of parameters. For each ith point, define $f^{(i)} : \mathcal{H} \to \mathbb{R}$ by $f^{(i)}(\mathbf{w}) = (z^{(i)} - \langle \mathbf{w}, \mathbf{y}^{(i)} \rangle)^2$, which is a function of $\mathbf{w}$: $z^{(i)}$ and $\mathbf{y}^{(i)}$ are fixed. Similarly, we define a loss function for each of the terms t_j, $f_j : \mathcal{H} \to \mathbb{R}$, as the average loss over these data points $\{\mathbf{x}^{(i)} = (\mathbf{x}^{(i)}, \mathbf{y}^{(i)}, z^{(i)})\}$ in the term t_j (beware we abuse the notation to let $\mathbf{x}^{(i)}$ denote the ith point $(\mathbf{x}, \mathbf{y}, z)^{(i)}$). Note these loss functions are stochastic, depending on the sample from the distribution $\mathbf{x} \sim D$.

The expected loss for a fixed term t_j is: $\mathbb{E}[f_j(\mathbf{w})] = \mathbb{E}_{\mathbf{x}^{(i)}}[(z^{(i)} - \langle \mathbf{w}, \mathbf{y}^{(i)} \rangle)^2 | t_j]$. Similarly, for I_{good}, we define the loss function $f_{I_{good}}(\mathbf{w}) = \frac{1}{|I_{good}|} \sum_{\mathbf{x}^{(i)} \in I_{good}} f^{(i)}(\mathbf{w})$. Let $\bar{f}$ denote the expected loss function for points averaged over I_{good}, $\bar{f} = \mathbb{E}[f_{I_{good}}]$. Then the optimal $\mathbf{w}^*$ is defined as $\mathbf{w}^* := \arg\min_{\mathbf{w}} \bar{f}(\mathbf{w})$. Our ultimate goal is to find $\hat{\mathbf{w}}$ that minimizes $\bar{f}$, but the difficulty is that $\bar{f}$ is unknown (since I_{good} is unknown). To overcome this barrier, instead of directly minimizing $\bar{f}$, we try to find parameters $\hat{\mathbf{w}}$ such that $\bar{f}(\hat{\mathbf{w}}) - \bar{f}(\mathbf{w}^*)$ is small. Once we get a close approximation $\hat{\mathbf{w}}$, we can use the covering algorithm of Juba et al. (2018) to find a good corresponding condition $\hat{\mathbf{c}}$.

Definition 3.1 (Reformulation of the problem) *Given D a distribution over points $\{\mathbf{x}^{(i)} := (\mathbf{x}^{(i)}, \mathbf{y}^{(i)}, z^{(i)})\}_{(i)=1}^N$, and $\{t_j\}_{j=1}^m$ predefined disjoint subsets (terms), let I_{good} be the (unknown) target collection corresponding to $\mathbf{c}^* = \bigcup_{t_j \in \mathbf{c}^*} t_j$ with probability mass $Pr[\mathbf{x} \in I_{good}] \geq \mu$, and $\bar{f}$ be the regression loss over I_{good}. If there exists a linear predictor $\mathbf{w}^*$ such that $\bar{f}(\mathbf{w}^*) \leq \epsilon$. Then we want to find $\hat{\mathbf{w}}$ that approximates $\mathbf{w}^*$ such that $\bar{f}(\hat{\mathbf{w}}) \leq \epsilon + error$.*

3.2 Main Optimization Algorithm

The main algorithm is an alternating-minimization algorithm: we assign "local" regression parameters $\mathbf{w}_i$ for each term t_i, and use a semi-definite program

(SDP) to minimize the total loss $\sum |t_i| f_i(\mathbf{w}_i)$ with regularization to force these parameters to be close to each other. Following each iteration, we use SDP (2) to remove outliers, by decreasing the weight factors c_i for those terms without enough neighbors. Intuitively, if there is a good linear rule $\mathbf{w}^*$ on I_{good}, then for each term $t_i \subset I_{good}$, $f_i(\mathbf{w}^*)$ should be small. Therefore, we can find a small ellipse Y bounding all parameters $\{w_i\}$ for the terms in I_{good}. We shift the parameters so that the ellipse is centered at 0. SDP (1) finds such an ellipse while minimizing the weighted total loss.

Algorithm 1: Soft regression algorithm

Input: terms $t_{1:m}$. **Output:** parameters $\hat{\mathbf{w}}_{1:m}$ and a matrix $\hat{Y}$

Initialize $c_{1:m} \leftarrow (1, \ldots, 1)$, $\lambda \leftarrow \sqrt{8\mu} N t S / r$

repeat

Let $\hat{\mathbf{w}}_{1:m}, \hat{Y}$ be the solution to the SDP:

$$\begin{aligned} &\underset{\mathbf{w}_1,\ldots,\mathbf{w}_m,Y}{\text{minimize}} \ \textstyle\sum_{i=1}^m c_i |t_i| f_i(\mathbf{w}_i) + \lambda \mathrm{tr}(Y) \\ &s.t. \quad \mathbf{w}_i \mathbf{w}_i^\top \preceq Y \text{ for } i = 1, \ldots, m. \end{aligned} \tag{1}$$

if $\mathrm{tr}(\hat{Y}) > 6r^2/\mu$ **then**

for $i = 1$ **to** m **do**

Let $\tilde{\mathbf{w}}_i$ be the solution to the SDP:

$$\begin{aligned} &\underset{\tilde{\mathbf{w}}_i, a_{i1},\ldots,a_{im}}{\text{minimize}} \ f_i(\tilde{\mathbf{w}}_i) \quad s.t. \\ &\mathbf{I}: \tilde{\mathbf{w}}_i = \textstyle\sum_{j=1}^m a_{ij} \hat{\mathbf{w}}_j, \quad \mathbf{II}: \sum_{j=1}^m a_{ij} = 1, \\ &\mathbf{III}: 0 \le a_{ij} \le \tfrac{2}{\mu N} |t_j| \quad \forall j \end{aligned} \tag{2}$$

$z_i \leftarrow f_i(\tilde{\mathbf{w}}_i) - f_i(\hat{\mathbf{w}}_i)$

end for

$z_{max} \leftarrow \max\{ \ z_i \mid c_i \neq 0\}$, **for** $i = 1$ **to** n **do**: $c_i' \leftarrow c_i . \frac{z_{max} - z_i}{z_{max}}$.

end if

until $\mathrm{tr}(\hat{Y}) \le \frac{6r^2}{\mu}$

Return $\hat{\mathbf{w}}_{1:m}, \hat{Y}$

We solve SDP (2) for each term t_i to find its best μN neighbor points and compute the "average" parameter $\tilde{\mathbf{w}}_i$ over the neighborhood. $\tilde{\mathbf{w}}_i$ is a linear combination of its neighbors' parameters: $\tilde{\mathbf{w}}_i = \sum_{j=1}^m a_{ij} \hat{\mathbf{w}}_j$, minimizing the term's loss $f_i(\tilde{\mathbf{w}}_i)$. Intuitively, if a term is a good term, i.e. $t_i \subset I_{good}$, then its parameter $\hat{\mathbf{w}}_i$ should be close to the average of parameters of all terms in I_{good}, $\mathbf{w}_i \approx \sum_{I_{good}} \frac{|t_j|}{|I_{good}|} \mathbf{w}_j$. In the SDP for t_i, we define coefficients a_{ij} to play the role of $\frac{|t_j|}{|I_{good}|}$. These coefficients $\{a_{ij}\}$ are required to sum to 1, i.e. $\sum_{j=1}^m a_{ij} = 1$, and each should not be larger than $\frac{|t_j|}{|I_{good}|} \sim 2\frac{|t_j|}{\mu N}$. At a high level, the SDP computes the best neighbors for t_i by assigning $\{a_{ij}\}$, so that the average parameter $\tilde{\mathbf{w}}_i$ over the neighbors minimizes f_i. If a term is bad, it is hard to find such good neighbors, so if the loss $f_i(\tilde{\mathbf{w}}_i)$ is much larger than the original loss, then we consider the term to be an outlier, and down-weight its weight factor c_i.

3.3 A Linear Bound on the Loss

Based on a spectral norm analysis, we will show the bound will shrink linearly with the radius as long as we have enough data. First, to estimate the losses by their inputs, we introduce the gradient ∇f. By the convexity of f, we have $(f(\mathbf{w}) - f(\mathbf{w}^*)) \le \langle \nabla f(\mathbf{w}), \mathbf{w} - \mathbf{w}^* \rangle$. Note that $\|\mathbf{w} - \mathbf{w}^*\|$ is bounded by $2r$, where $r := \max \|\mathbf{w}\|_2$. To bound the gradients, we use the spectral norm:

$$S := \max_{\mathbf{w} \in \mathcal{H}} \frac{1}{\sqrt{t}} \| \left[\nabla f_j(\mathbf{w}) - \nabla \bar{f}(\mathbf{w}) \right]_{j \in I_{good}} \|_{op}$$

It is the 2-norm of the matrix whose rows are gradients of loss functions of terms in I_{good}: $(\nabla f_i(\mathbf{w}) - \nabla \bar{f}(\mathbf{w}))_{i \in I_{good}}$. S measures the difference between the gradient of loss functions of terms in I_{good}: $\nabla f_i(\mathbf{w})$ and gradient of average loss on I_{good}: $\nabla \bar{f}(\mathbf{w})$. At a high level, this bound tells us how bad these loss functions could be. We will show S shrinks as the radius of parameters r decreases.

For linear regression, $f_j(\mathbf{w}) := \frac{1}{|t_j|} \sum_{i \in t_j} f^{(i)}(\mathbf{w})$, and $\nabla f_j(\mathbf{w}) = \frac{1}{|t_j|} \sum_{i \in t_j} \nabla f^{(i)}(\mathbf{w})$, where for each point $\nabla f^{(i)}(\mathbf{w}) = 2(\mathbf{w}^\top \mathbf{y}^{(i)} - z^{(i)}) \mathbf{y}^{(i)}$. If we assume $z^{(i)} = \mathbf{w}^{*\top} \mathbf{y}^{(i)} + \epsilon^{(i)}$, and the residual $\epsilon^{(i)}$ (a subgaussian, e.g., from $\mathcal{N}(0, \sigma_\epsilon^2)$) is independent of $\mathbf{y}^{(i)}$, then we can write $(\nabla f_j(\mathbf{w}) - \nabla \bar{f}(\mathbf{w}))$ as

$$2(\mathbf{w}^\top - \mathbf{w}^{*\top}) \Big(\sum_{i \in t_j} \frac{\mathbf{y}^{(i)} \mathbf{y}^{(i)\top}}{|t_j|} - \mathbb{E}[\mathbf{y}\mathbf{y}^\top] \Big) + \sum_{i \in t_j} \frac{\epsilon^{(i)} \mathbf{y}^{(i)}}{|t_j|}$$

where the first part is going to shrink as $(\mathbf{w}^\top - \mathbf{w}^{*\top})$ decreases and the second part approaches zero as we draw more data, $\frac{1}{|t_j|} \sum_{t_j} \epsilon^{(i)} \mathbf{y}^{(i)} \to 0$. More concretely, we define

$$S_0 := \| \Big[\frac{1}{|t_j|} \sum_{t_j} \mathbf{y}^{(i)} \mathbf{y}^{(i)\top} - \mathbb{E}[\mathbf{y}\mathbf{y}^\top] \Big]_{I_{good}} \|_{op}.$$

Note, S_0 is fixed given the data, and thus remains constant across iterations. Furthermore, S_0 concentrates around $\| \left[\mathbb{E}[\mathbf{y}^{(i)} \mathbf{y}^{(i)\top} | t_j] - \mathbb{E}[\mathbf{y}\mathbf{y}^\top] \right]_{I_{good}} \|_{op}$ and can thus be bounded. The bound on S we can guarantee will decrease when we take more points. We obtain

Lemma 3.2 *For $N = \mathcal{O}(\sigma_\epsilon^2 \log(m/\delta)/\beta\mu r^2))$ points, with probability $1 - \delta$ the spectral norm of the gradients S is bounded by a linear function of the radius $r := \max_{\mathbf{w}} \|\mathbf{w}\|_2$, i.e., $S = \mathcal{O}(rS_0)$.*

3.4 List-regression Algorithm

We now introduce the recentering algorithm. Following Charikar et al. (2017), we initially use Algorithm 1 to assign a parameter $\hat{\mathbf{w}}_i$ for each term. In each iteration, we check if the termination condition $r \le \frac{1}{2} r_{final}$ is met. If not, we repeatedly find Padded Decompositions $\mathcal{P}_h$ (Fakcharoenphol et al., 2003) to cluster the terms by their parameters $\hat{\mathbf{w}}_i$ and reuse Algorithm 1 on each cluster to get estimates $\bar{\mathbf{w}}_i(h)$. Finally we pick a central $\bar{\mathbf{w}}_i(h_0)$ and update $\hat{\mathbf{w}}_i$. In each iteration the $\hat{\mathbf{w}}$ of good terms will get closer, so we can decrease the radius of the ellipse containing I_{good} by half. Eventually, the algorithm will be able to constrain the parameters for all of the good terms in a very small ellipse. The algorithm then outputs a list of candidate parameters, with one approximating $\mathbf{w}^*$.

Algorithm 2: List-regression algorithm

Input: m terms, target radius r_{final}. **Output:** candidate solutions $\{\mathbf{u}_1, ..., \mathbf{u}_s\}$ and $\hat{\mathbf{w}}_{1:m}$.
Initialize $r^{(1)} \leftarrow r$, $\hat{\mathbf{w}}^{(1)}_{1:m} \leftarrow$ Algorithm 1 with center 0 and radius r.
for $\ell = 1, 2, \ldots$ **do**
  $\mathcal{W} \leftarrow \{\hat{\mathbf{w}}_i^{(\ell)} |\ \hat{\mathbf{w}}_i^{(\ell)}$is assigned$\}$
  if $r^{(\ell)} < \frac{1}{2} r_{final}$ **then**
    Greedily find maximal sets $\{\mathbf{u}_1, ..., \mathbf{u}_s\}$ s.t.
      I: $|B(\mathbf{u}_j; 2r_{\text{final}}) \cap \mathcal{W}| \ge (1-\beta)\mu N,\ \forall j$.
      II: $\|\mathbf{u}_j - \mathbf{u}_{j'}\|_2 > 4r_{\text{final}}, \quad \forall j \ne j'$.
    Return $\mathcal{U} = \{\mathbf{u}_1, ..., \mathbf{u}_s\}, \hat{\mathbf{w}}^{(\ell)}_{1:m}$.
  end if
  for $h = 1$ **to** $112 \log(\ell(\ell+1)/\delta)$ **do**
    $\bar{\mathbf{w}}_{1:m}(h) \leftarrow$ unassigned
    Let $\mathcal{P}_h$ be a $(\rho, 2r^{(\ell)}, \frac{7}{8})$-padded decomposition of $\mathcal{W}$ with $\rho = \mathcal{O}(r^{(\ell)} \log(\frac{2}{\mu}))$.
    for $T \in \mathcal{P}_h$ **do**
      Let $B(u, \rho)$ be a ball containing T. Run Algorithm 1 on $\mathcal{H} \cap B(u, \rho)$, with radius $r = \rho$ and center shifted to u. For each $\hat{\mathbf{w}}_i \in T$ assign $\bar{\mathbf{w}}_i(h)$ as the outputs of Algorithm 1.
    end for
  end for
  for $i = 1$ **to** m **do**
    Find a h_0 such that $\|\bar{\mathbf{w}}_i(h_0) - \bar{\mathbf{w}}_i(h)\|_2 \le \frac{1}{3} r^{(\ell)}$ for at least $\frac{1}{2}$ of the h's.
    $\hat{\mathbf{w}}_i^{(\ell+1)} \leftarrow \bar{\mathbf{w}}_i(h_0)$ (or "unassigned" if no such h_0 exists)
  end for
  $r^{(\ell+1)} \leftarrow \frac{1}{2} r^{(\ell)}$
end for

The analysis uses a "local" spectral norm bound for any β fraction of points. We get good estimates of $\mathbf{w}$ for any sufficiently large subset larger than β. A key observation is that in contrast to Charikar et al., we do not "lose" points from our clusters across iterations since our terms are all large enough that they are preserved. This enables a potentially arbitrarily-close approximation of $\mathbf{w}^*$ given enough data.

For $\beta < 1$, we define a local spectral norm bound S_β on arbitrary subsets T in I_{good}, such that T takes up at least a β fraction of I_{good} ($N_T \ge \beta N$). Denote the number of points in T by N_T and the number of terms by m_T. We define

$$S_\beta := \max_{\substack{w \in \mathcal{H},\ T \subset I_{good} \\ s.t.\ N_T \ge \beta N}} \frac{1}{\sqrt{m_T}} \|[\nabla f_j(\mathbf{w}) - \nabla \bar{f}(\mathbf{w})]_{j \in T}\|_{op}.$$

Similar to the analysis of S, $S_\beta = \mathcal{O}(r S_{\beta_0})$, where if we abuse notation to let a DNF $\mathbf{t}$ also denote a set of terms (thus, $|\mathbf{t}|$ denotes the number of terms in $\mathbf{t}$),

$$S_{\beta_0} := \max_{\substack{\mathbf{t} \subseteq \mathbf{c}\ s.t. \\ \Pr[\mathbf{t}(\mathbf{x}) | \mathbf{c}] \ge \beta}} \frac{1}{\sqrt{|\mathbf{t}|}} \left\| \left[\mathbb{E}[\mathbf{y}\mathbf{y}^\top | t_j] - \mathbb{E}[\mathbf{y}\mathbf{y}^\top | \mathbf{c}] \right]_{t_j \in \mathbf{t}} \right\|_{op}.$$

$S_{\beta 0}$ describes the inherent distribution of the data on the terms. When there exists a solution where the terms have similar covariances, then $S_{\beta 0}$ will be small, e.g., if the data distribution is (nearly) independent of which of the terms are satisfied. Unfortunately, although S_β converges to $S_{\beta 0}$ as N increases, $S_{\beta 0}$ is an intrinsic property of the distribution that does not decrease with N. We denote the value of S_β in the ℓth iteration by $S_\beta^{(\ell)}$, where $S_\beta^{(\ell)} = \mathcal{O}(r^{(\ell)} S_{\beta_0})$.

As we deleted all terms of size smaller than $\beta\mu N$, all the remaining terms have at least $\beta\mu$ probability-weight (or $\beta\mu N$ empirical size). Then, we can show that every term will satisfy $\|\hat{\mathbf{w}}_i - \hat{\mathbf{w}}_{avg}\|_2^2 \le \frac{10}{\kappa}(\sqrt{\text{tr}(\hat{Y})} + r) t S_\beta$. Using this observation we can show that for each iteration, the $\hat{\mathbf{w}}_i$ for terms in I_{good} are within $\mathcal{O}(\frac{r^{(\ell)} t S_\beta^{(\ell)}}{\kappa\sqrt{\mu}})$ of $\mathbf{w}^*$. Finally, we show the radius $r^{(\ell)}$ (used in the ℓth iteration) can then be decreased by half at each iteration. In conclusion we obtain:

Theorem 3.3 *Let any r_{final} and $\delta, \beta \le \frac{1}{2}$ be given. Suppose that the loss functions f_i are κ-strongly convex and $S_{\beta_0} \le \mathcal{O}(\frac{\kappa\sqrt{\mu}}{t \log(1/\mu)})$ for all $i \in I_{good}$. For $N = \mathcal{O}(\sigma_\epsilon^2 \log(m/\delta)/\beta\mu r_{final}^2))$ points, let $\mathcal{U}, \hat{\mathbf{w}}_{1:m}$ be the output of Algorithm 2. Then with probability at least $1-\delta$, $\mathcal{U}$ has size at most $\lfloor \frac{1}{(1-\beta)\mu} \rfloor$, and $\min_{\mathbf{u} \in \mathcal{U}} \|\mathbf{u} - \mathbf{w}^*\|_2 \le \mathcal{O}(r_{final})$. Moreover, $\|\hat{\mathbf{w}}_i - \mathbf{w}^*\|_2 \le \mathcal{O}(r_{final})$ for every term $i \in I_{good}$.*

3.5 Obtaining a k-DNF Condition

Once we get outputs $\{\mathbf{u}_1, ..., \mathbf{u}_s\}$ from Algorithm 2, we switch from the parameter space $\{\mathbf{w}\}$ back to the

Boolean data space $\{\mathbf{x}\}$, to search for corresponding conditions $\mathbf{c}$ for each candidate parameter $\mathbf{u}$. If we find a pair $(\mathbf{u}, \mathbf{c})$ such that $\mathbf{c}$ contains enough points and the loss $f_{\mathbf{c}}(\mathbf{u})$ is small, we return this pair as the final solution. Suppose $\mathbf{u}$ is the candidate that $\|\mathbf{u} - \mathbf{w}^*\| < \mathcal{O}(r_{final}) =: \gamma$, then $|\bar{f}(\mathbf{u}) - \bar{f}(\mathbf{w}^*)| \leq \gamma L = \mathcal{O}(\gamma)$ for some Lipschitz constant L. Recalling $\bar{f}$ is nonnegative, if $\bar{f}(\mathbf{w}^*) \leq \epsilon$, then $\bar{f}(\mathbf{u}) \leq \gamma + \epsilon$.

We now address the effect of our duplication of points. We added a copy of a point for each term it satisfied. On I_{good}, which contains t terms, each point has at most t copies. We thus obtain:

Lemma 3.4 *Let* $\mathbf{u}$ *be such that* $\|\mathbf{u} - \mathbf{w}^*\| < \gamma$. *Then* $|\bar{f}(\mathbf{u})| \leq t(\gamma + \epsilon)$.

We have obtained a parameter vector $\mathbf{u}$ such that the loss for each term $f_i(\mathbf{u})$ is close to $f_i(\mathbf{w}^*)$. We can now use a greedy set-cover algorithm to find the corresponding conditions $\mathbf{c}$, following the approach of Juba et al. (2018). At a high level, given regression parameters $\mathbf{u}$, we compute the loss $f(\mathbf{u})$ for each point, and then use the covering algorithm to find a collection of terms that cover enough points while minimizing the loss. Specifically, the algorithm greedily chooses terms t_j satisfying $\sum_{i \in t_j} f^{(i)}(\mathbf{u}) \leq (1+\gamma)\mu\epsilon N$ to maximize the number of additional points $(\mathbf{x}, \mathbf{y}, z)^{(i)}$ with $t_j(\mathbf{x}^{(i)}) = 1$ that did not satisfy previously chosen terms. It continues choosing terms this way until at least $(1 - \gamma/2)\mu N$ examples satisfy the collection of chosen terms.

Lemma 3.5 *If there exists an optimal* k*-DNF* $\mathbf{c}^*$ *that is satisfied by a* μ*-fraction of the points with total loss* ϵ*, then, the weighted greedy set cover algorithm can find a* k*-DNF* $\hat{\mathbf{c}}$*, that is satisfied by a* $(1-\gamma)\mu$*-fraction of the points with total loss* $\mathcal{O}(t \log(\mu N)\epsilon)$

We can bound the generalization error of linear regression on each possible k-DNF using the Rademacher generalization bound for linear predictors (Kakade et al., 2009), and then take a union bound to prove the main theorem. In short, the process will blow up the complexity by $d^3 B^6 t/\gamma$, where d is the dimension of the feature space. Overall we achieve a $\mathcal{O}(t \log(\mu N)(\gamma + \epsilon))$ approximation as claimed in Theorem 2.2 with $N = \mathcal{O}(\frac{B^6 d^3 \sigma^2 L^2 t^2}{\mu \gamma^4} \log(m/\delta)))$ examples.

4 Experiments

We now present experiments showing that on real data sets, our algorithm can scale up to large, moderately high dimensional data (unlike the previous, sparse algorithms) and that we obtain loss that is consistently similar to or significantly smaller than that of the

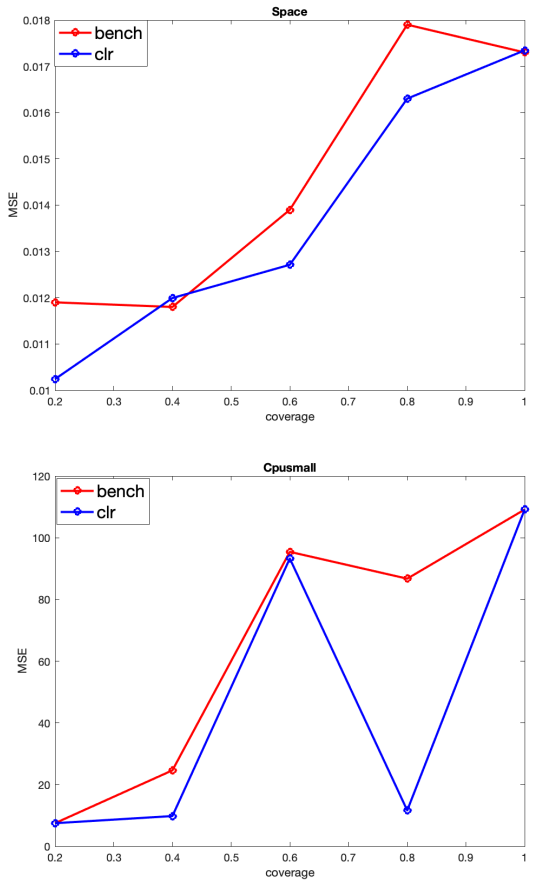

Figure 1: Linear regression on LIBSVM datasets
Top: Space data: $N = 1025, dim(x) = dim(y) = 6$.
Bot: Cpu data: $N = 2703, dim(x) = dim(y) = 12$.

sparse ℓ_2 regression algorithm (Hainline et al., 2019). Our prototype code is written in Matlab using the Yalmip library (Löfberg, 2004), with Mosek (mos) as our SDP solver. We first test our algorithm on two of the larger benchmark data sets from the LIBSVM repository (Chang and Lin, 2011), Space and Cpusmall, used previously by Hainline et al. (2019), and compared the loss achieved for several target fractions. These data sets contain only real-valued attributes. Following Hainline et al's strategy, we generate Boolean attributes using indicators for membership in the empirical 50%-quantile of each real attributes. We randomly selected 1/3 of the data to use for training and the other 2/3 for testing. Similar to Hainline et al., we use the algorithm to find a (list of) 2-DNFs on the training data, compute a linear rule on the corresponding subset on the training data, and then test the loss of the rule on the corresponding subset of the testing data. To cope with instability we observed in the SDP solver, we decreased the number of padded decompositions (in Algorithm 2)

from $112\log(l(l+1)/\delta)$ to 12, and instead repeated the algorithm 50 times to produce a list of candidates. We compute the linear rule and training loss for each candidate DNF, and then use the DNF with lowest training loss as our final output to test on the testing data. We repeated the experiments using different values of μ $(0.2, 0.4, 0.6, 0.8, 1.0)$ and set the parameters $S = \mu \cdot 10, \gamma = 0.1, r_{final} = S \cdot dim(y)$, where $dim(y)$ is the dimension of the real features. As shown in Figure 1, our error is lower than or comparable to that of the baseline algorithm in all cases. Remarkably, our running time is much better than the baseline. Their algorithm required a few days of cloud computing time, while our algorithm only required a few hours.

On Cpusmall, the loss at $\mu = 0.8$ is much smaller than at $\mu = 0.6$. Usually we expect the loss to decrease with μ: since our problem allows solutions on which the condition comprises 80% of the data when we are only seeking 60%, it is strictly easier to fit a smaller subset. This difference is not caused by the double-counting of points; the algorithm does not obtain such accurate estimates of the parameters on $\mu = 0.6$. The value of μ enters Algorithm 1 in several places, and the guarantees we can provide feature a $1/\sqrt{\mu}$ factor blow-up in the error (see supplemental material; notice, Theorem 2.2 requires a stronger condition for smaller μ). Indeed, in the soft regression/outlier detection procedure, we might expect that as the proportion of "signal" decreases, we'd get less accurate estimates. Thus, in practice, we advise running the algorithm several times using $\mu = (1-\Delta)^i$ for some number of iterations ($\Delta \in (0,1)$) down to a desired minimum, and taking the output with the smallest loss.

One may worry about overfitting for small μ. This is not an issue as long as we use enough data and runs. We observe that for $\mu = 0.2$, 10 out of the 50 runs do have much larger testing error than training error, which implies overfitting. But when we pick the candidate with minimum training error, overfitting doesn't impact our output. To prevent overfitting, we suggest using larger data sets and more trials so that even small subsets have lower variance. We also caution that our algorithm requires enough data to be stable. On some smaller data sets that have only about 100 examples, we found that the SDP solver could not solve instances in which the radius of the parameter space decreased below 1. Thus, the final parameters we obtained for this data provided poor estimates that were not competitive with the algorithm of Hainline et al.

To further demonstrate the scalability of our algorithm, we consider the Physicochemical Protein Structure data set (Rana, 2013) from the UCI repository, which consists of 45730 data points with 9 real features. (Additional synthetic data experiments illustrating the scalability can be found in the supplemental material.) Instead of just using 50%-quantiles as the Boolean attributes as before, we considered sets of quantiles: $\{50\%\}$, $\{25\%, 75\%\}$, or $\{25\%, 50\%, 75\%\}$, resulting in 9, 18, and 27 Boolean attributes, respectively. Since 20% of the data will give an adequately large testing set, we can use 80% of the data as our training set. Notice, since the running time of Hainline et al's algorithm has a linear dependence on the data set size in the dominant term, it is not feasible run on such a large data set. The algorithm is more stable on this larger data set: we observed each run gets similar results. Thus, unlike the previous experiments, we can just report the results of a single run. We also noticed that the max iteration of Algorithm 1 can be constrained to around 10 and the radius $r^{(1)}$ in Algorithm 2 can be initialized close to r_{final} to reduce the number of iterations, with similar quality.

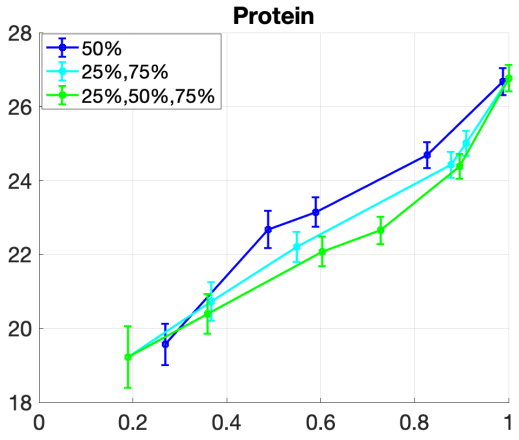

Figure 2: Scalability and effect of different quantiles. MSE vs. coverage for **Protein data:** $N = 36584, dim(y) = 9, dim(x) = \{9, 18, 27\}$

As shown in Figure 2, at coverage below 100% we again get a significantly reduced MSE compared to regression on the entire data set. As the coverage decreases the MSE also decreases, for all sets of quantiles. Using more quantiles can slightly improve the performance, at the cost of more Boolean attributes. Similar to Rosenfeld et al. (2015), we found the "extreme" quantiles were more useful, in that the MSE was not significantly reduced by the addition of the 50% quantile compared with just using 25% and 75%.

In conclusion, our algorithm is better than the previous algorithm for the larger benchmark data sets ($N \geq 1000$). Since it collapses the data points into loss matrices for terms, and the corresponding pre- and post-processing can be done in roughly linear time, it can scale up to very large data sets.

Acknowledgements

Brendan Juba was supported by an AFOSR Young Investigator Award and NSF award CCF-1718380; part of this work was performed while visiting the Simons Institute for Theory of Computing. Part of this work was performed as an REU at Washington University in St. Louis, when Diego Calderon was supported by WUSEF and Lisa Ruan was supported by the NSF Big Data Analytics REU Site, award IIS-1560191.

References

Mosek. `https://www.mosek.com/`.

P. Awasthi, A. Blum, and O. Sheffet. Improved guarantees for agnostic learning of disjunctions. In *Proc. 23rd COLT*, pages 359–367, 2010.

C.-C. Chang and C.-J. Lin. Libsvm: a library for support vector machines. *ACM Transactions on Intelligent Systems and Technology (TIST)*, 2(3):27, 2011.

M. Charikar, J. Steinhardt, and G. Valiant. Learning from untrusted data. In *Proc. 49th STOC*, pages 47–60, 2017. Full version arXiv:1611.02315v2 [cs.LG].

C. Cortes, G. DeSalvo, and M. Mohri. Learning with rejection. In *ALT 2016*, volume 9925 of *LNAI*, pages 67–82. 2016.

I. Diakonikolas, D. M. Kane, and A. Stewart. List-decodable robust mean estimation and learning mixtures of spherical Gaussians. In *Proc. 50th STOC*, pages 1047–1060, 2018.

R. El-Yaniv and Y. Wiener. Pointwise tracking the optimal regression function. In *Advances in Neural Information Processing Systems 25*, pages 2042–2050, 2012.

J. Fakcharoenphol, S. Rao, and K. Talwar. A tight bound on approximating arbitrary metrics by tree metrics. In *Proc. 35th STOC*, pages 448–455, 2003.

M. A. Fischler and R. C. Bolles. Random sample consensus: A paradigm for model fitting with applications to image analysis and automated cartography. *Communications of the ACM*, 24(6):381–395, 1981.

J. Hainline, B. Juba, H. S. Le, and D. P. Woodruff. Conditional sparse ℓ_p regression with optimal probability. In *Proc. 22nd AISTATS*, volume 89 of *PMLR*, pages 369–382, 2019.

P. J. Huber. *Robust Statistics*. John Wiley & Sons, New York, NY, 1981.

J. Jiang. *Linear and Generalized Linear Mixed Models and Their Applications*. Springer, Berlin, 2007.

B. Juba. Conditional sparse linear regression. In *Proc. 8th ITCS*, pages 45:1–45:14, 2017.

B. Juba, Z. Li, and E. Miller. Learning abduction under partial observability. In *Proc. 32nd AAAI*, pages 1888–1896, 2018.

S. M. Kakade, K. Sridharan, and A. Tewari. On the complexity of linear prediction: Risk bounds, margin bounds, and regularization. In *Advances in Neural Information Processing Systems 21*, pages 793–800, 2009.

S. Karmalkar, P. Kothari, and A. Klivans. List-decodable linear regression. In *Advances in Neural Information Processing Systems 32*, pages 7423–7432. 2019.

J. Löfberg. Yalmip : A toolbox for modeling and optimization in matlab. In *In Proceedings of the CACSD Conference*, Taipei, Taiwan, 2004.

C. E. McCulloch and S. R. Searle. *Generalized, Linear, and Mixed Models*. John Wiley & Sons, New York, NY, 2001.

Y. W. Park, Y. Jiang, D. Klabjan, and L. Williams. Algorithms for generalized cluster-wise linear regression. *INFORMS Journal on Computing*, 29(2):301–317, 2017.

J. R. Quinlan. Learning with continuous classes. In *5th Australian Joint Conference on Artificial Intelligence*, volume 92, pages 343–348. Singapore, 1992.

P. Raghavendra and M. Yau. List decodable learning via sum of squares. In *Proc. 31st SODA*, pages 161–180, 2020.

P. Rana. Physicochemical properties of protein tertiary structure data set, 2013.

A. Rosenfeld, D. G. Graham, R. Hamoudi, R. Butawan, V. Eneh, S. Kahn, H. Miah, M. Niranjan, and L. B. Lovat. MIAT: A novel attribute selection approach to better predict upper gastrointestinal cancer. In *Proc. IEEE International Conference on Data Science and Advanced Analytics (DSAA)*, pages 1–7, 2015.

P. J. Rousseeuw and A. M. Leroy. *Robust Regression and Outlier Detection*. John Wiley & Sons, New York, NY, 1987.

M. Zhang, T. Mathew, and B. Juba. An improved algorithm for learning to perform exception-tolerant abduction. In *Proc. 31st AAAI*, pages 1257–1265, 2017.