Sublinear-Time Learning and Inference for High-Dimensional Models

Enxu Yan

May, 2018
CMU-ML-18-103

Machine Learning Department
School of Computer Science
Carnegie Mellon University
Pittsburgh, PA 15213

Thesis Committee:
Pradeep Ravikumar (Chair)
Geoffrey J. Gordon
Barnabas Poczos
Cho-Jui Hsieh
Inderjit S. Dhillon

Submitted in partial fulfillment of the requirements for the degree of Doctor of Philosophy.

This research was supported by the National Science Foundation award number IIS1661755.

Keywords: Sparsity, Optimization, Primal and Dual methods, Extreme Classification, Deep Neural Network, Structured Prediction, MAP Inference, Latent-Feature Models, Mixed Regression.

For my family.

Abstract

Across domains, the scale of data and complexity of models have both been increasing greatly in the recent years. For many models of interest, tractable learning and inference without access to expensive computational resources have become challenging. In this thesis, we approach efficient learning and inference through the leverage of sparse structures inherent in the learning objective, which allows us to develop algorithms sublinear in the size of parameters without compromising the accuracy of models. In particular, we address the following three questions for each problem of interest: (a) how to formulate model estimation as an optimization problem with tractable sparse structure, (b) how to efficiently, i.e. in sublinear time, search, maintain, and utilize the sparse structures during training and inference, (c) how to guarantee fast convergence of our optimization algorithm despite its greedy nature? By answering these questions, we develop state-of-the-art algorithms in varied domains. Specifically, in the extreme classification domain, we utilizes primal and dual sparse structures to develop greedy algorithms of complexity sublinear in the number of classes, which obtain state-of-the-art accuracies on several benchmark data sets with one or two orders of magnitude speedup over existing algorithms. We also apply the primal-dual-sparse theory to develop a state-of-the-art trimming algorithm for Deep Neural Networks, which sparsifies neuron connections of a DNN with a task-dependent theoretical guarantee, which results in models of smaller storage cost and faster inference speed. When it comes to structured prediction problems (i.e. graphical models) with inter-dependent outputs, we propose decomposition methods that exploit sparse messages to decompose a structured learning problem of large output domains into factorwise learning modules amenable to sublinear-time optimization methods, leading to practically much faster alternatives to existing learning algorithms. The decomposition technique is especially effective when combined with search data structures, such as those for Maximum Inner-Product Search (MIPS), to improve the learning efficiency jointly. Last but not the least, we design novel convex estimators for a latent-variable model by reparameterizing it as a solution of sparse support in an exponentially high-dimensional space, and approximate it with a greedy algorithm, which yields the first polynomial-time approximation method for the Latent-Feature Models and Generalized Mixed Regression without restrictive data assumptions.

Acknowledgments

I would like to sincerely express my gratitude to my advisor, Pradeep Ravikumar, for the continuous support of my Ph.D study and research, for giving me rooms to explore diverse research directions, and providing guidance when we face challenges. I could not have imagined having a better advisor and mentor for my Ph.D study. I would also like to thank the rest of my thesis and proposal committee: Geoffrey Gordon, Barnabas Poczos, Inderjit Dhillon, and Cho-Jui Hsieh for their insightful comments and encouragement during my thesis dissertation. I also thank my intimate labmates and collaborators, Kai Zhong, Xiangru Huang, Lingfei Wu, Arun Sai, Adarsh Prasad, Wei Dai, Xin Lin, Jiong Zhang, Qi Lei, Chao-Yuan Wu, Ruohan Zhang for the stimulating discussions and days and nights we were working together before deadlines in the last four years. In addition, I would like to thank all the mentors, professors and students that I have ever co-worked with during my internship and remote collaborations: Shou-De Lin, Dmitry Malioutov, Abhishek Kumar, Satyen Kale, Felix Yu, Sanjiv Kumar, Pushmeet Kohli, Abdelrahman Mohamed, Michael Rosenblum, Han Liu, Rui Yan, Qixing Huang, Ting-Wei Lin, Shan-Wei Lin, Wei-Cheng Li, Yu-Cheng Lu, Sung-En Chang, and many others. Last but not the least, I would like to thank my family: my wife's continuous support during my best and also worst time, without which I could not possibly get through these years, and my parents' tolerance for my leaving home country for the career development.

Contents

List of Figures

List of Tables

Chapter 1

Introduction

Along with the dramatic improvement of modern Artificial Intelligence systems in the recent years, the model complexity has also grown to another level, where a single model could easily involve millions of parameters and billions of operations. At the same time, the amount of data for training is also growing with the unlimited resources on the internet, crowdsourcing technology, and also data augmentation techniques. This puts significant challenges to the modern Machine Learning system as the computational cost of training is growing with both the model complexity and the amount of data. For many tasks of interest, it has become increasingly impossible in practice to train a model of state-of-the-art performance without access to expensive computational resources such as dozens of GPU/CPUs. Sometimes even performing inference is computationally expensive, especially on devices such as mobile phones or other edge devices—they could easily consume too much energy or take too much time and space.

1.1 Thesis Statement

In this thesis, we consider a number of situations for which the size of model parameters could easily go beyond millions:

- **Extreme Classification**: consider a simple linear model for document tagging using *Bag-of-Words* features. For practical applications with both vocabulary size and the number of tags being hundreds of thousands, a simple linear model would have $(10^5)^2$ parameters, which is very hard to store in a computer's physical memory, and the training would be even more expensive.
- **Structured Prediction**: for problem involving an output domain of size up to thousands, modeling the interaction between outputs could easily introduce millions of parameters. One example is the n-gram language model used in speech recognition and machine translation, which often results in hard inference problems that can only be approximated via Beam Search in practice.
- **Deep Architecture**: a recent trend of modeling is to introduce multiple layers of feature extraction before predicting the outputs, which introduces huge amount of parameters and expensive operations such as 2D convolutions. In practice, an accurate DNN often involves billions of operations for the prediction of a single instance.

- **Latent-Variable Model**: The latent-variable model (LVM) itself does not necessarily involve large number of parameters. However, to design robust estimators for LVM, in this thesis we define a novel convex estimator with exponential number of variables, which therefore requires algorithms that scale sublinearly with the number of variables.

In this thesis, we approach the tractability of large-scale estimation problems by leveraging structures inherent in the learning objective, which allows us to: (i) design sublinear-time algorithm for existing objective without sacrificing quality of the solution, and (ii) design new objective (i.e. new estimator) that has potentially exponential or even infinite number of variables without sacrificing tractability of the problem. For the first part, we use *Extreme Classification*, *Structured Prediction* and *Deep Neural Network* as main examples to demonstrate how to exploit sparse structures of the objective to design sublinear-time algorithms when the problem has huge number of variables. For the second part, we show that, under the sparse optimization scheme, we can formulate discrete latent-variable models such as *Latent-Feature Allocation* and *Mixture of Regression* as high-dimensional sparse estimation problems, which allow us to develop novel methods with theoretical guarantees without strong distributional assumption on the data.

1.2 Contributions and Outline

In the following, we introduce a couple of domains where our proposed approaches have been shown successful, and discuss the contributions of this thesis within each domain.

Extreme Classification Extreme Classification encompasses multiclass and multilabel problems with huge number of classes or labels. Problems of this kind are prevalent in real-world applications such as text, image or video annotation, where one aims to learn a predictor that tags a data point with the most relevant labels out of a huge collection. In such settings, standard approaches such as one-versus-all and one-versus-one become intractable both in training and prediction phase due to the computations involving large number of model parameters [23]. There are a couple of approaches proposed in the literature to address this issues: One of the most popular approach to reduce complexity is to impose structural relations among labels to reduce the model complexity such as low rank [15, 51, 125], tree-structure [17, 18, 77] and clusters [66]. However, in practice such structural assumptions do not often hold and thus enforcing such structures often significantly decreases the model performance. Another popular approach to speed up learning in Extreme Classification is to sample a small number of candidate negative classes as contrasts to the positive labels and evaluate the loss function (and its gradient) approximately [35, 47, 68], which however often results in slower convergence and thus does not lead to an overall speedup. In this thesis, unlike existing structural approaches, we investigate *sparse structures inherent in the learning objective*, which allows us to qualitatively reduce the computational cost without sacrificing model's accuracy. On several benchmark datasets for extreme classification, our approach has demonstrated the state-of-the-art accuracy while being one or two orders of magnitude faster than the competing methods in both single-core environments and massively parallelizable environments.

Structured Prediction with Large Output Domain Structured prediction has wide applications such as Natural Language Processing (NLP), Computer Vision, and Bioinformatics, where one is interested in outputs of strong interdependence. Although many dependency structures yield intractable inference problems, approximation techniques such as belief propagation and convex relaxations [56, 64, 80] have been developed. However, solving those approximate objective (LP, QP, SDP, etc.) is still computationally expensive for factor graphs of large output domain, which results in prohibitive training time when embedded into a learning algorithm relying on the inference oracles [48, 59]. For instance, many applications in NLP such as Machine Translation [29], Speech Recognition [106], and Semantic Parsing [21] have output domains as large as the size of vocabulary in the target language, giving an expensive cost of inference. In this thesis, we propose a new approach to learning structured predictor which, instead of employing inference as a subroutine in a learning algorithm, takes the *learning of each factor* as subroutines and passes messages between factors to guarantee convergence to the same solution. This reduces the complexity of learning w.r.t. the size of output domain from quadratic (or more) to sublinear due the sublinear size of messages given by the number of active classes in the (multiclass) learning subproblem posed by each factor, where the spirit is to reduce the entropy of the belief on each factor so one can pass *sparse messages* among factors. We demonstrate the effectiveness of the decomposition technique in both learning phase and prediction phase (with MAP inference), where the proposed greedy decomposition algorithm enjoys an order of magnitude faster convergence than the competing optimization algorithms on problems with thousands of output candidates.

Compression of Deep Neural Network State-of-the-art Deep Neural Network (DNN) typically involves millions of parameters. For example, the VGG-16 network from the winning team of ILSVRC-2014 contains more than one hundred millions of parameters and the inference of one single image on VGG-16 takes tens of billions of operations, which prohibits its use on edge devices such as mobile phones or in real-time applications. A recent thread of researches has thus focused on compressing the DNN and one of the key steps of compression is to trim the connections between neurons, which reduces the number of non-zero parameters and thus the model size [1, 34, 36, 37, 69]. However, there has been a huge gap between theory and practice on this topic. In particular, the trimming algorithms of practical success have been relying on heuristics [34, 36, 37] subject to certain failure cases, while the performance of existing theoretically-motivated approach has been less competitive to the heuristics-based approaches [1]. In this thesis, we revisit the simple idea of trimming DNNs through ℓ_1 regularization and exhibit two surprising results: (i) our analysis suggests that, for any stationary point under the ℓ_1 regularization, the number of non-zero parameters at each layer of a DNN should not be more than the number of penalized prediction logits—an upper bound typically several orders of magnitude smaller than the total number of DNN parameters; (ii) it is critical to employ an ℓ_1-friendly optimization solver targeting for *high precision*, instead of SGD, in order to find the *stationary point of sparse support*—in our experiments a Proximal Quasi-Newton method can compress the fully-connected layers of VGG-16 (accounting for more than 90% of parameters) by *4 orders of magnitudes*, a ratio much better than the current state of the art.

Estimation of Latent-Variable Models Tractable estimation for models with binary latent variables such as *Latent-Feature Model (LFM)* (also called *Indian Buffet Process (IBP)*) and mixed regression (MR) have been difficult [32, 67, 84, 97, 124, 127]. It is in part due to the combinatorial nature of the binary incidence vectors. Indeed, with N samples, and K latent components, the number of possible binary matrices consisting of the N binary feature incidence vectors is 2^{NK}, and the log-likelihood of such models is not a concave function of its parameters. In practice, one often employs local search methods similar to *Expectation Maximization* [11], Markov Chain Monte Carlo (MCMC) [24] or variational methods [25] to find a feasible solution. However, none of these approaches provide guarantees on the quality of solution within finite time. The use of *Spectral Sethods* bypasses the problem of non-concave log-likelihood by estimating the *moments* derived from the model [12, 84, 97], which however requires a high-order sample complexity and also the knowledge of data distribution and thus is impractical. In this thesis, we propose a novel approach to the estimation of binary latent-variable models by formulating it as a high-dimensional sparse optimization problem and greedily searching binary incidence vectors through solving a MAX-CUT problem. Unlike Spectral method, our approach does not rely on any restrictive assumption on the distribution of data but provide polynomial runtime and approximation guarantees.

Chapter 2

Learning and Inference with Sparse Structures

In this chapter, we consider the problem of extreme classification and introduce our first methodology for developing algorithms of sublinear cost w.r.t. the model size (i.e. size of output domain) by exploiting a type of structure, termed Primal-Dual sparsity (PD-Sparse), according to our thread of works in [113, 118]. We will first develop a sequential optimization algorithm in section 2.1, and then give a twist in the obective function to make the PD-Sparse technique massively parallelizable in an environment of hundreds or thousands of cores in section 2.2.

2.1 Extreme Classification: Primal & Dual Sparsity

Extreme Classification encompasses multiclass and multilabel problems with huge number of classes or labels. Problems of this kind are prevalent in real-world applications such as text, image or video annotation, where one aims to learn a predictor that tags a data point with the most relevant labels out of a huge collection. In the multiclass setting, we are given the fact that only one label is correct, while in the multilabel setting, multiple labels are allowed.

In the Extreme Classification setting, standard approaches such as one-versus-all and one-versus-one become intractable both in training and prediction phase due to computations involving large number of model parameters [23]. Recently several approaches have been proposed to exploit structural relations between labels for reducing training and prediction time. A natural approach is to find an embedding so the model parameters of each label can be projected to a low-dimensional space, reducing the cost in training and prediction [15, 51, 125]. However, in real applications, the data may not be low-rank and the low-rank approaches may result in lower accuracy. Furthermore, for high-dimensional data with a sparse feature matrix, the model learned from low-rank approach can project a sparse feature vector into a dense vector that results in even higher prediction cost than a simple linear classifier.

Another recent thread of research has investigated tree-based methods that partition labels into tree-structured groups, so in both training and prediction phases, one can follow tree branches to avoid accessing irrelevant models [17, 18, 77]. However, finding a balanced tree structure that partitions labels effectively in the feature space is a difficult problem in itself. While many

heuristics have been proposed for finding a good tree structure, in practice, one needs to ensemble several trees to achieve performance comparable to standard classifiers.

In this paper, instead of making a structural assumption on the label relationships, we assume that for each instance, there are only a few correct labels and the feature space is rich enough for one to distinguish between labels. This assumption is much weaker than other structural assumptions, and under such an assumption, we show that a simple margin-maximizing loss yields an extremely sparse dual solution in the setting of extreme classification. Furthermore, the loss, when combined with ℓ_1 penalty, gives a sparse solution both in the primal and in the dual for any ℓ_1 regularization parameter $\lambda > 0$.

We thus propose a Fully-Corrective Block Coordinate Frank-Wolfe algorithm to solve the primal-dual sparse problem given by margin-maximizing loss with ℓ_1-ℓ_2 penalties. Let D be the problem dimension, N be the number of samples, and K be the number of classes. In case $DK \gg N$, the proposed algorithm has complexity sublinear to the number of variables by exploiting sparsity in the primal to search active variables in the dual. In case $DK \lesssim N$, we propose a stochastic approximation method to further speed up the search step in the Frank-Wolfe algorithm. In our experiments on both multiclass and multilabel problems, the proposed method achieves significantly higher accuracy than existing approaches of Extreme Classification with competitive training and prediction time.

2.1.1 Problem Setup

Our formulation is based on the Empirical Risk Minimization (ERM) framework. Given a collection of training instances $\mathcal{D} = \{(\boldsymbol{x}_i, \boldsymbol{y}_i)\}_{i=1}^N$ where $\boldsymbol{x}_i \in \mathbb{R}^D$ is D-dimensional (possibly sparse) feature vector of i-th instance and $\boldsymbol{y}_i \in \{0,1\}^K$ is label indicator vector with $y_{ik} = 1$ if k is a correct label for the i-th instance and $y_{ik} = 0$ otherwise. We will use $\mathcal{P}(\boldsymbol{y}) = \{k \in [K] \mid y_k = 1\}$ to denote positive label indexes, while using $\mathcal{N}(\boldsymbol{y}) = \{k \in [K] \mid y_k = 0\}$ to denote the negative label indexes. In this work, we assume the number of labels K is extremely large but the number of positive labels $nnz(\boldsymbol{y})$ is small and not growing linearly with K. For example, in multiclass classification problem, we have $nnz(\boldsymbol{y}) = 1$, and the assumption is also satisfied typically in mulitlabel problems. Denote $X := (\boldsymbol{x}_i^T)_{i=1}^N$ as the $N \times D$ design matrix and $Y := (\boldsymbol{y}_i^T)_{i=1}^N$ as the N by K label matrix, our goal is to learn a classifier $h : \mathbb{R}^D \to [K]$

$$h(\boldsymbol{x}) := \underset{k}{argmax} \, \langle \boldsymbol{w}_k, \boldsymbol{x} \rangle, \tag{2.1}$$

parameterized by a $D \times K$ matrix $W = (\boldsymbol{w}_k)_{k=1}^K$.

Loss with Dual Sparsity In this paper, we consider the *separation ranking loss* [20] that penalizes the prediction on an instance $\boldsymbol{x}$ by the highest response from the set of negative labels minus the lowest response from the set of positive labels

$$L(\boldsymbol{z}, \boldsymbol{y}) = \max_{k_n \in \mathcal{N}(\boldsymbol{y})} \max_{k_p \in \mathcal{P}(\boldsymbol{y})} \left(1 + z_{k_n} - z_{k_p}\right)_+ \tag{2.2}$$

where an instance has zero loss if all positive labels $k_p \in \mathcal{P}_i$ have higher responses than that of negative labels $k_n \in \mathcal{N}_i$ plus a margin. In the multiclass setting, let $p(\boldsymbol{y})$ be the unique positive

label. The loss (2.2) becomes the well-known multiclass SVM loss

$$L(\boldsymbol{z}, \boldsymbol{y}) = \max_{k \in [K] \setminus \{p(\boldsymbol{y})\}} \left(1 + z_k - z_{p(\boldsymbol{y})}\right)_+ \tag{2.3}$$

proposed in [19] and widely-used in linear classification package such as LIBLINEAR [28]. The basic philosophy of loss (2.2) is that, for each instance, there are only few labels with high responses, so one can boost prediction accuracy by learning how to distinguish between those confusing labels. Note the assumption is reasonable in Extreme Classification setting where K is large and only few of them are supposed to give high response. On the other hand, this does not give much computational advantage in practice, since, to identify labels of high response for each instances, one still needs to evaluate (2.1) for $\forall n \in [N], \forall k \in [K]$, resulting in an $O(nnz(X)K)$ complexity that is of the same order to the one-vs-all approach. [52] proposed an approach in the multiclass setting that tries to identify active variables corresponding to labels of high responses in the dual formulation of the ℓ_2-regularized instance

$$\frac{1}{2} \sum_{k=1}^{K} \|\boldsymbol{w}_k\|^2 + C \sum_{i=1}^{N} L(W^T \boldsymbol{x}_i, y_i), \tag{2.4}$$

where $W = [\boldsymbol{w}_1, \boldsymbol{w}_2, ..., \boldsymbol{w}_K]$. Let $\boldsymbol{\alpha}_k := (\alpha_{ik})_{i \in [N]}$ and $\boldsymbol{\alpha}^i := (\alpha_{ik})_{k \in [K]}$. The dual problem is of the form

$$\begin{aligned} \min_{\boldsymbol{\alpha}} \quad & \frac{1}{2} \sum_{k=1}^{K} \|\boldsymbol{w}_k(\boldsymbol{\alpha})\|^2 + \sum_{i=1}^{N} \boldsymbol{e}^T \boldsymbol{\alpha}^i \\ s.t. \quad & \boldsymbol{\alpha}^i \in \Delta_i^K, \ \forall i \in [N] \end{aligned} \tag{2.5}$$

where

$$\boldsymbol{w}_k(\boldsymbol{\alpha}_k) = \sum_{i=1}^{N} \alpha_{ik} \boldsymbol{x}_i = X^T \boldsymbol{\alpha}_k, \tag{2.6}$$

$\boldsymbol{e}_i = \boldsymbol{1} - \boldsymbol{y}_i$, and $\Delta_i^K = \{\boldsymbol{\alpha} \mid \sum_{k=1}^{K} \alpha_k = 0, \ \alpha_{p(\boldsymbol{y}_i)} \leq C, \ \alpha_k \leq 0, \ \forall k \neq p(\boldsymbol{y}_i)\}$ is a shifted simplex of K corners. In particular, the optimal solution $\boldsymbol{\alpha}$ of (2.5) satisfies: for $k \neq p(\boldsymbol{y}_i)$, $\alpha_{ik}^* \neq 0$ if and only if label k has highest response $z_{ik} = \langle \boldsymbol{w}_k, \boldsymbol{x}_i \rangle$ that attains the maximum of (2.3). Therefore, to identify active variables that correspond to the confusing labels, [52] proposes a *shrinking* heuristic that "shrinks" a dual variable whenever its descent direction towards the boundary. The shrunken variables are then excluded from the optimization, which in practice reduces training time by orders of magnitude. While the shrinking heuristic is quite successful for problem of medium number of class K. For problem of K more than 10^4 labels, the technique becomes impractical since even computing gradient for each of the $N \times K$ variables once requires days of time and hundreds of gigabytes of memory (as shown in our experiments).

Primal and Dual-Sparse Formulation One important observation that motivates this work is the intriguing property of ERM with dual-sparse loss (2.2) and ℓ_1 penalty

$$\lambda \sum_{k=1}^{K} \|\boldsymbol{w}_k\|_1 + \sum_{i=1}^{N} L(W^T \boldsymbol{x}_i, \boldsymbol{y}_i). \tag{2.7}$$

in the setting of Extreme Classification. Consider the optimal solution W^* of (2.7), which satisfies

$$\lambda \boldsymbol{\rho}_k^* + \sum_{i=1}^{N} \alpha_{ik}^* \boldsymbol{x}_i = \lambda \boldsymbol{\rho}_k^* + X^T \boldsymbol{\alpha}_k = 0, \ \forall k \in [K] \tag{2.8}$$

for some subgradients $\boldsymbol{\rho}_k^* \in \partial \|\boldsymbol{w}_k^*\|_1$ and $\boldsymbol{\alpha}^{i*} \in \partial_{\boldsymbol{z}} L(\boldsymbol{z}_i, \boldsymbol{y}_i)$ with $\boldsymbol{z}_i = W^{*T} \boldsymbol{x}_i$. Recall that the subgradients $\boldsymbol{\alpha}^i$ of loss (2.2) have $\alpha_{ik^*} \neq 0$ for some $k^* \neq \bar{k}$ only if k^* is the confusing label that satisfies

$$k^* \in arg \max_{k \neq \bar{k}} \langle \boldsymbol{w}_k, \boldsymbol{x}_i \rangle.$$

This means we have $nnz(\boldsymbol{\alpha}^i) \ll K$ and $nnz(A) \ll NK$ as long as there are few labels with higher responses than the others, which is satisfied in most of Extreme Classification problems. On the other hand, the subgradient $\boldsymbol{\rho}_k$ of ℓ_1-norm satisfies

$$\rho_{jk} = \begin{cases} 1, & w_{jk}^* > 0 \\ -1, & w_{jk}^* < 0 \\ \nu, & \nu \in [-1, 1], w_{jk}^* = 0, \end{cases} \tag{2.9}$$

which means the set of non-zero primal variables $\mathcal{B}_k^* = \{j \mid w_{jk}^* \neq 0\}$ at optimal satisfies

$$\lambda \ sign([\boldsymbol{w}_k^*]_{\mathcal{B}_k^*}) \mathbf{1}_{\mathcal{B}_k^*} = \left[X^T \boldsymbol{\alpha}_k^*\right]_{\mathcal{B}_k^*}, \tag{2.10}$$

which is a linear system of $|\mathcal{B}_k^*|$ equality constraints and $nnz(\alpha_k)$ variables. However, for general design matrix X that draws from any continuous probability distribution [95], the above cannot be satisfied unless

$$nnz(\boldsymbol{w}_k^*) = |\mathcal{B}_k^*| \leq nnz(\boldsymbol{\alpha}_k^*), \forall k \in [K] \tag{2.11}$$

and (2.11) further implies

$$nnz(W^*) \leq nnz(A^*) \tag{2.12}$$

by summation over K, where A^* an $N \times K$ matrix of stacked $(\boldsymbol{\alpha}_k^*)_{k=1}^K$. This means in Extreme Classification problem, not only non-zero dual variables but also primal variables are sparse at optimal. Note this result holds for any ℓ_1 parameter $\lambda > 0$, that means it does not gain primal sparsity via sacrificing the expressive power of the predictor. Instead, it implies there exists a naturally sparse optimal solution W^* under the loss (2.2), which can be found through imposing a very small ℓ_1 penalty. The result is actually a simple extension to the fact that the number of non-zero weights at optimal under ℓ_1 penalty is less or equal to the number of samples [95]. We summarize the result as following Corollary.

Corollary 1 (Primal and Dual Sparsity)**.** *The optimal primal and dual solution* (W^*, A^*) *of ERM problem* (2.7) *with loss* (2.2) *satisfies*

$$nnz(W^*) \leq nnz(A^*)$$

for any $\lambda > 0$ *if the design matrix* X *is drawn from a continuous probability distribution.*

Dual Optimization via Elastic Net Although (2.7) has superior sparsity, both the primal and dual optimization problems for (2.7) are non-smooth and non-separable w.r.t. coordinates, where greedy coordinate-wise optimization could be non-convergent [1]. However, from the duality between strong convexity and dual smoothness [50, 63], this issue can be resolved simply via adding an additional strongly convex term in the primal. In particular, by adding an ℓ_2 regularizer to (2.7), the *Elastic-Net*-regularized problem

$$\sum_{k=1}^{K} \frac{1}{2}\|\boldsymbol{w}_k\|^2 + \lambda\|\boldsymbol{w}_k\|_1 + C\sum_{i=1}^{N} L(W^T\boldsymbol{x}_i, \boldsymbol{y}_i) \tag{2.13}$$

has dual form

$$\begin{aligned} \min_{\boldsymbol{\alpha}} \quad & G(\boldsymbol{\alpha}) := \frac{1}{2}\sum_{k=1}^{K}\|\boldsymbol{w}_k(\boldsymbol{\alpha}_k)\|^2 + \sum_{i=1}^{N}\boldsymbol{e}_i^T\boldsymbol{\alpha}^i \\ s.t. \quad & \boldsymbol{\alpha}^i \in \mathcal{C}_i,\ \forall i \in [N] \end{aligned} \tag{2.14}$$

that entangles variable of different samples $\boldsymbol{\alpha}^i$, $\boldsymbol{\alpha}^{i'}$ only through a smooth term $\sum_{k=1}^{K}\|\boldsymbol{w}_k(\boldsymbol{\alpha}_k)\|^2/2$, where

$$\boldsymbol{w}_k(\boldsymbol{\alpha}_k) := \mathbf{prox}_{\lambda\|.\|_1}(X^T\boldsymbol{\alpha}_k). \tag{2.15}$$

and

$$\mathcal{C}_i := \left\{ \boldsymbol{\alpha} \middle| \begin{array}{l} \sum_{k\in\mathcal{N}_i}(-\alpha_k) = \sum_{k\in\mathcal{P}_i}\alpha_k \in [0, C], \\ 0 \le \alpha_k, \forall k \in \mathcal{P}_i,\ \alpha_k \le 0, \forall k \in \mathcal{N}_i \end{array} \right\}. \tag{2.16}$$

The proximal operator of ℓ_1-norm $\mathbf{prox}_{\lambda\|.\|_1}(\boldsymbol{v})$ performs soft-thresholding to each single element v_j as

$$\mathbf{prox}_{\lambda|.|}(v_j) := \begin{cases} 0, & |v_j| \le \lambda \\ v_j - \lambda, & v_j > \lambda \\ v_j + \lambda, & v_j < \lambda \end{cases}$$

The dual problem (2.14) has very similar form to that from purely ℓ_2 regularized problem (2.5), with difference on the definition of $\boldsymbol{w}_k$ (2.15), where the ℓ_1-ℓ_2-regularized problem has $\boldsymbol{w}_k(\boldsymbol{\alpha}_k)$ being a sparse vector obtained from applying soft-thresholding operator to $X^T\boldsymbol{\alpha}_k$. This, however, leads to the key to our efficiency gain. In particular, the objective allows efficient search of active dual variables via sparsity in the primal, while allows efficient maintenance of nonzero primal variables through an active-set strategy in the dual.

Note the Elastic-Net-regularized problem could not satisfy corollary 1. However, empirically, it has been observed to produce solution of sparsity close to that from ℓ_1 regularizer, while the solution from Elastic-Net is often of higher prediction accuracy [128]. In our experiments, we have observed extremely sparse primal solution from (2.13) which not only help in the training phase but also results in faster prediction that is competitive to the logarithmic-time prediction given by tree-based approach [18].

[1] The coordinate descent method has global convergence only on problem where the non-smooth terms are separable w.r.t. the coordinates.

Algorithm 1 Fully-Corrective BCFW

[0:] Initialize $\boldsymbol{\alpha}^0 = \mathbf{0}$, $\mathcal{A}^0 = \emptyset$.
for $t = 1...T$ **do**
  [1:] Draw a sample index $i \in [N]$ uniformly at random.
  [2:] Find most-violating label $k^* \in \mathcal{N}_i$ via (2.20).
  [3:] $\mathcal{A}_i^{t+\frac{1}{2}} = \mathcal{A}_i^t \cup \{k^*\}$.
  [4:] Solving subproblem (2.21) w.r.t. active set $\mathcal{A}_i^{t+\frac{1}{2}}$.
  [5:] $\mathcal{A}_i^{t+1} = \mathcal{A}_i^{t+\frac{1}{2}} \setminus \{k \mid \alpha_{ik} = 0, k \notin \mathcal{P}_i\}$.
  [6:] Maintain $\boldsymbol{w}(\boldsymbol{\alpha})$, $\boldsymbol{v}(\boldsymbol{\alpha})$ via (2.22).
end for.

2.1.2 Algorithm

The objective (2.14) comprises a smooth function subject to constraints $\mathcal{C}_1, .., \mathcal{C}_N$ separable w.r.t. blocks of variables $\boldsymbol{\alpha}_1, \boldsymbol{\alpha}_2, ..., \boldsymbol{\alpha}_N$. A fast convergent algorithm thus minimizes (2.14) one block at a time. In this section, we propose a Fully-Corrective Block-Coordinate Frank-Wolfe (BCFW) for the dual problem (2.14) that explicitly taking advantage of the primal and dual sparsity.

Note for the similar dual problem (2.5) resulted from L2-regularization, a BCFW method that searches the greedy coordinate α_{ik}^* at each iterate is not better than a Block Coordinate Descent (BCD) algorithm that performs updates on the whole block of variable $\boldsymbol{\alpha}^i$ [28, 52], since the greedy search requires evaluation of gradient w.r.t. each coordinate, which results in the same cost to minimizing the whole block of variables, given the minimization can be done via a simplex projection.

On the other hand, our dual objective (2.14) has gradient of i-th block equals

$$\nabla_{\boldsymbol{\alpha}^i} G(\boldsymbol{\alpha}) = W^T \boldsymbol{x}_i - \boldsymbol{e}_i. \tag{2.17}$$

If a primal-sparse W can be maintained via (2.15), the gradient costs $O(nnz(x_i)nnz(w_j))$ (and $O(nnz(W))$ for dense x_i). In contrast, the update of the whole block of variable $\boldsymbol{\alpha}^i$ would require maintaining relation (2.15) for $\boldsymbol{w}_1 ... \boldsymbol{w}_K$, which cannot exploit sparsity of $\boldsymbol{w}_k$ and would require $O(nnz(x_i)K)$ (and $O(DK)$ for dense x_i). So in the Extreme Classification setting, the cost of updating an coordinate is orders of magnitude larger than cost of evaluating its gradient.

Fully-Corrective Block-Coordinate Frank Wolfe (FC-BCFW) As a result, we employ a BCFW strategy where the updates of variables are restricted to an active set of labels $\mathcal{A}_i^t$ for each sample i. In each iteration, the BCFW method draws a block of variables $\boldsymbol{\alpha}^i$ uniformly from $\{\boldsymbol{\alpha}^i\}_{i=1}^N$, and finds greedy direction based on a local linear approximation

$$\boldsymbol{\alpha}_{FW}^{it} := \underset{\boldsymbol{\alpha}^i \in \mathcal{C}_i}{argmin} \langle \nabla_{\boldsymbol{\alpha}^i} G(\boldsymbol{\alpha}^t), \boldsymbol{\alpha}^i \rangle. \tag{2.18}$$

For $\mathcal{C}_i$ of structure (2.16), (2.18) is equivalent to finding the most violating pair of positive, negative labels:

$$(k_n^*, k_p^*) := \underset{k_n \in \mathcal{N}_i, k_p \in \mathcal{P}_i}{argmin} \langle \nabla_{\boldsymbol{\alpha}^i} G(\boldsymbol{\alpha}^t), (\boldsymbol{\delta}_{k_p} - \boldsymbol{\delta}_{k_n}) \rangle, \tag{2.19}$$

where $\boldsymbol{\delta}_k$ is $K \times 1$ indicator vector for k-th variable. However, since we are considering problem where $|\mathcal{P}_i|$ is small, we can keep all positive labels in the active set $\mathcal{A}_i$. Then to guarantee that the FW direction (2.18) is considered in the active set, we only need to find the most-violating negative label:

$$\begin{aligned} k_n^* &:= \underset{k_n \in \mathcal{N}_i}{argmax} \ \langle \nabla_{\boldsymbol{\alpha}^i} G(\boldsymbol{\alpha}^t), \boldsymbol{\delta}_{k_n} \rangle, \\ &= \underset{k_n \in \mathcal{N}_i}{argmax} \ \langle \boldsymbol{w}_k^t, \boldsymbol{x}_i \rangle - 1 \end{aligned} \tag{2.20}$$

which costs $O(nnz(\boldsymbol{x}_i)nnz(\boldsymbol{w}_{\bar{j}}))$, where $\bar{j}$ is the feature j of most non-zero labels in W, among all nonzero features $\boldsymbol{x}_i$.

After adding k_n^* to the active set, we minimize objective (2.14) w.r.t. the active set and fix $\alpha_{ik} = 0$ for $\forall k \notin \mathcal{A}_i$ by solving the following block subproblem

$$\min_{\boldsymbol{\alpha}_{\mathcal{A}_i} \in \mathcal{C}_i} \langle \nabla_{\boldsymbol{\alpha}_{\mathcal{A}_i}} G, \boldsymbol{\alpha}_{\mathcal{A}_i} - \boldsymbol{\alpha}_{\mathcal{A}_i}^t \rangle + \frac{Q_i}{2} \|\boldsymbol{\alpha}_{\mathcal{A}_i} - \boldsymbol{\alpha}_{\mathcal{A}_i}^t\|^2 \tag{2.21}$$

where $Q_i = \|\boldsymbol{x}_i\|^2$ and $\mathcal{A}_i = \mathcal{A}_i^t \cup \{k_n^*\}$. Note, when $|\mathcal{P}_i| = 1$, the subproblem (2.21) can be solved by a simple projection to simplex of complexity $O(|\mathcal{A}_i| \log |\mathcal{A}_i|)$. For $|\mathcal{P}_i| > 1$, we derive a similar procedure that generalizes projection of simplex to that for the constraint $\mathcal{C}_i$ of the same complexity.

After solving the subproblem (2.21) w.r.t. the active set $\mathcal{A}_i$, we update $\boldsymbol{w}_k(\boldsymbol{\alpha}_k^t)$ to $\boldsymbol{w}_k(\boldsymbol{\alpha}_k^{t+1})$ by maintaining an additional vector $\boldsymbol{v}_k^t$ such that

$$\boldsymbol{v}_k^t = X^T \boldsymbol{\alpha}_k^t, \quad \boldsymbol{w}_k^t = \mathbf{prox}_{\lambda \|.\|}(\boldsymbol{v}_k^t). \tag{2.22}$$

where maintaining the first relation costs $O(nnz(\boldsymbol{x}_i)|\mathcal{A}_i|)$ and maintaining the second requires the same cost by checking only values changed by the first step.

The overall space requirement of Algorithm 1 for storing non-zero dual variables $\{\boldsymbol{\alpha}^i\}_{i=1}^N$ is bounded by $|\mathcal{A}| \ll NK$, while the storage for maintaining primal variable is dominated by the space for $\{\boldsymbol{v}_k\}_{k=1}^K$, which in the worst case, requires $O(DK)$. However, by the definition of $\boldsymbol{v}_k$ (2.22), the number of non-zero elements in $\{\boldsymbol{v}_k\}_{k=1}^K$ is bounded by $O(nnz(X) \max_i(|\mathcal{A}_i|))$, with $\max_i(|\mathcal{A}_i|)$ bounded by the number of BCFW passes. This means the space requirement of the algorithm is only t times of the data size $nnz(X)$ for running t iterations. In practice, $|\mathcal{A}_i|$ converges to the number of active labels of sample i and does not increase after certain number of iterations.

The overall complexity for each iterate of FC-BCFW is $O(nnz(\boldsymbol{x}_i)nnz(\boldsymbol{w}_{\bar{j}}^t) + nnz(\boldsymbol{x}_i)|\mathcal{A}_i^t|)$. In case data matrix is dense the cost for one pass of FC-BCFW over all variables can be written as $O(Nnnz(W) + Dnnz(A))$, where A is the N by K matrix reshape of $\boldsymbol{\alpha}$. Let

$$k_W := nnz(W)/D \ , \ k_A := nnz(A)/N$$

be the average number of active labels per feature and per sample respectively. We have

$$O(Nnnz(W) + Dnnz(A)) = O(NDk_W + NDk_A).$$

Note k_A is bounded by the number of BCFW passes, and it is generally small when label has diverse responses on each instance. On the other hand, suppose the Elastic-Net penalty leads

to sparsity similar to that of ℓ_1-regularized problem (which we observed empirically). We have $nnz(W) \lesssim nnz(A)$ and thus $k_W \lesssim \frac{N}{D} k_A$, which means k_W is small if $D \approx N$. On the other hand, for problem of small dimension, the bound becomes useless as the $\frac{N}{D} k_A$ can be even larger than K. In such case, the search (2.20) becomes the bottleneck.

Theorem 1 (Convergence of FC-BCFW)**.** *Let $G(\boldsymbol{\alpha})$ be the dual objective* (2.14)*. The iterates $\{\boldsymbol{\alpha}^t\}_{t=1}^{\infty}$ given by the Fully-Corrective Block-Coordinate Frank-Wolfe (Algorithm 1) has*

$$G(\boldsymbol{\alpha}^t) - G^* \leq \frac{2(QR^2 + \Delta G^0)}{t/N + 2}\ ,\ t \geq 0 \tag{2.23}$$

where $Q = \sum_{i=1}^{N} Q_i$, $\Delta G^0 := G(\boldsymbol{\alpha}^0) - G^*$ and $R = 2C$ is the diameter of the domain (2.16)*.*

Note our objective $G(\boldsymbol{\alpha}^t)$ is N times of the objective defined by *average loss* in for example [59, 85], so one would divide both sides of (2.23) by N to compare the rates.

2.1.3 Experiments

In this section we compare our proposed Primal-Dual Sparse(PD-Sparse) method with existing approaches to multiclass and multilabel problems. In all experiments, we set C=1 for all methods based on Empirical Risk Minimization, and choose $\nu = 3$ and $\lambda \in \{0.01, 0.1, 1, 10\}$ that gives best accuracy on a heldout data set for our method. To prevent over-fitting, we compute accuracy on a heldout data set to determine number of iterations used in all the iterative solvers. The compared algorithms are listed as follows.

- LibLinear [28] one-versus-all logistic regression (1vsA-Logi).
- LibLinear one-vs-all SVM (1vsA-SVM).
- LibLinear multiclass SVM (Multi-SVM).
- LibLinear one-vs-all l_1-regularized logistic regression solver (1vsA-L1-Logi).
- Vowpal-Wabbit (VW): A public fast learning system proposed in [18] for Extreme multiclass classification. We use the online trees (Tree) options provided by their solver.
- FastXML: An Extreme multilabel classification method [77] that organizes models with tree structure. We use solver provided by the author with default parameters.
- LEML: A low-rank Empirical-Risk-Minimization solver from [125]. We use solver provided by the authors with best rank parameter chosen from $\{50, 100, 250, 500, 1000\}$.
- SLEEC: A method based on Sparse Local Embeddings for Extreme multilabel classification [8]. We use solver provided by the author with default parameters.

Among these solvers, LibLinear multiclass SVM, Vowpal-Wabbit are only for multiclass problems. All other solvers can be used on both multiclass and multilabel data sets. Note FastXML, LEML and SLEEC are designed for multilabel problems but also applicable to multiclass problems.

Our experiments are conducted on 9 public data sets. Among them, *LSHTC1*, *Dmoz*, *imagenet*, *aloi.bin* and *sector* are multiclass and *LSHTC-wiki*, *EUR-Lex*, *RCV1-regions*, *bibtex* are multilabel. *ImageNet* uses bag-of-word features downloaded directly from ImageNet [2]. *EUR-Lex*

[2] `http://image-net.org/`

Table 2.1: Results on multiclass data sets. The numbers shown are (i) *training time*, (ii) *model size*, (iii) *prediction time* and (iv) *testing accuracy (in %)*, respectively, where N= number of train samples, K = number of classes, D = number of features. the best results among all solvers are marked. Multi-SVM is not applicable to Dmoz due to $> 200G$ memory requirement.

Data	FastXML	LEML	1vsA-Logi	1vsA-SVM	Multi-SVM	1vsA-L1-Logi	PD-Sparse	VW(Tree)	SLEEC
LSHTC1	2131s	78950s	$\approx$ 6d	23744s	6411s	$\approx$ 14d	**952s**	1193s	10793s
N=83805	308M	7.7G	$\approx$ 57G	11G	4.5G	$\approx$ **57M**	92M	744M	1.38G
D=347255	6.33s	189s	N/A	50.3s	49.0s	N/A	**6.20s**	6.84s	155s
K=12294	21.66	16.52	N/A	**23.22**	22.4	N/A	22.66*	10.56	12.8
Dmoz	6900s	97331s	$\approx$ 27d	136545s	N/A	$\approx$ 565d	**2068.14s**	7103s	113200s
N=345068	1.5G	3.6G	$\approx$ 96G	19G	N/A	$\approx$ 406M	**40M**	1.8G	3.23G
D=833484	57.1s	1298s	N/A	429.7s	N/A	N/A	**6.74s**	28s	3292s
K=11947	38.4	31.28	N/A	36.8	N/A	N/A	**39.58**	21.27	32.49
imgNet	28440s	107490s	380971s	28640s	14510s	472611s	**4958s**	6492s	520570s
N=1261404	914M	13M	14M	23M	24M	**1.8M**	3.6M	35M	2.76G
D=1000	139s	554s	315.3s	136.8s	203.4s	390.5s	329.5s	**37.7s**	45372s
K=1000	6.48	7.21	8.56	**15.25**	10.3	10.07	12.7	5.37	8.5
aloi.bin	2410s	62440s	42390s	9468s	449s	31770s	773.8s	**334.3s**	12200s
N=100000	992M	5.4G	15G	5.9G	612M	18M	**7.1M**	106M	1.96G
D=636911	10.99s	38.83s	16.61s	22.83s	12.42s	13.24s	**1.42s**	1.59s	191s
K=1000	95.5	88.16	96.34	96.63	**96.66**	95.71	96.33	89.47	92.55
sector	100.77s	556.31s	107.12s	19.46s	**11.46s**	102.31s	14.12s	327.34s	164.3s
N=8658	7.0M	48M	129M	62M	57M	**580K**	1.6M	17M	223.5M
D=55197	0.25s	**0.069s**	0.114s	0.156s	0.169s	0.13s	0.09s	0.16s	1.59s
K=105	84.9	94.07	90.8	94.79	95.11	93.13	**95.3***	82.1	87.62

and *bibtex* are from Mulan multilabel data collections. [3] *LSHTC1*, *Dmoz* and *LSHTC-wiki* are from LSHTC2 competition described in [76]. *RCV1-regions*, *aloi.bin* and *sector* are from LIBSVM data collection [4], where *aloi.bin* uses Random Binning features [79, 115] approximating effect of RBF Laplacian kernel.

The statistics of data sets and results are shown in Table 2.6 and 2.5. We include statistics of test and heldout data set in Appendix B. Note many one-vs-all solvers require running for a huge amount of time. We run a distributed version and use training time and models of at least 100 classes to estimate the expected total running time and model size.

As showed in the table, solvers rely on structural assumptions such as FastXML (tree), VW (tree), LEML (low-rank) and SLEEC (piecewise-low-rank) could obtain accuracy significantly worse than standard one-vs-all methods on multiclass data sets. Standard multiclass solvers however suffer from complexity growing linearly with K. On the other hand, by exploiting primal and dual sparsity inherent in Extreme Classification problem, PD-Sparse has training time, prediction time and model size growing sublinearly with K while keeping a competitive accuracy. As showed in Table 2.3, the average number of active dual variables for each sample is much smaller than the number of classes.

[3] `mulan.sourceforge.net/datasets-mlc.html`

[4] `www.csie.ntu.edu.tw/~cjlin/libsvmtools/datasets/multilabel.html`

Table 2.2: Results on Multilabel data sets. The numbers shown are (i) *training time*, (ii) *model size*, (iii) *prediction time* and (iv) *test top-1 accuracy (in %)*, respectively, where N= number of training samples, K = number of classes, D = number of features.

Data	FastXML	LEML	1vsA-Logi	1vsA-SVM	1vsA-L1-Logi	PD-Sparse	SLEEC
LSHTC-wiki	**104442s**	217190s	>10y	>96d	>10y	124867s	2224000s
N=2355436	8.9G	10.4G	≈ 426G	≈870G	≈ **358M**	685M	12.6G
D=2085167	164.8s	2896s	N/A	N/A	N/A	**15.56s**	8906s
K=320338	78.28	28.46	N/A	N/A	N/A	**89.3***	73.44
EUR-Lex	**317s**	7471s	22551s	3227s	32531s	434.9s	2443s
N=15643	324.5M	78M	257M	118M	14M	**8.0M**	80.8M
D=5000	**0.996s**	42.24s	7.93s	7.23s	1.39s	1.089s	4.89s
K=3956	67.3	67.82	**77.3**	64.5	73.8	76.3	74.2
RCV1-regions	94.06s	2247s	79.27s	14.73s	84.74s	**8.82s**	1129s
N=20835	14.61M	205M	129M	39M	**504K**	1.7M	204M
D=47237	0.824s	2.515s	0.486s	0.392s	0.174s	**0.115s**	15.8s
K=225	93.28	96.28	90.96	95.98	94.7	**96.54**	91
bibtex	18.35s	157.9s	8.944s	**3.24s**	13.97s	5.044s	298s
N=5991	27M	8.6M	3.7M	3.3M	412K	**68K**	26.7M
D=1837	0.09s	0.2215s	0.0383s	0.079s	0.0238s	**0.0059s**	0.94s
K=159	64.14	64.01	62.65	58.46	61.16	64.55	**65.09**

Table 2.3: Average number of active dual and primal variables (k_A, k_W respectively) when parameter λ maximizes heldout accuracy.

Data sets	k_A	k_W
EUR-Lex (K=3956)	20.73	45.24
LSHTC-wiki (K=320338)	18.24	20.95
LSHTC (K=12294)	7.15	4.88
aloi.bin (K=1000)	3.24	0.31
bibtex (K=159)	18.17	1.94
Dmoz (K=11947)	5.87	0.116

2.2 Parallelizable Primal-Dual Sparse Method

As we pointed out in the previous section, when the number of classes become large, the collection of model parameters is under-determined, so that a simple structural constraint of sparsity might be practically useful. For the specific max-margin loss, we have shown that there exists a naturally sparse solution to the Extreme Classification problem with sub-linear number of non-zeros in both primal and dual variables, which can be exploited to develop an estimation algorithm with sub-linear dependency on the number of classes. Since primal-dual sparsity is naturally satisfied in the Extreme Classification setting, the estimation algorithm often leads to higher accuracy on problems with larger number of classes. However, the max-margin loss employed in the previous section has several disadvantages: (i) it is not separable w.r.t. classes and thus requires optimizing parameters of all classes together, which can lead to a high memory consumption that prohibits its application to larger problems, (ii) the non-separability w.r.t. classes prevents a simple parallelization scheme that the one-versus-all methods enjoy, (iii) the max-margin loss focuses on the margin between the most confusing classes, which as a criterion is sensitive to *mis-labeling*, such as missing positive labels.

A recent work [6] shows that an one-versus-all based approach with weight truncation (for

model compression) can reduce the training time substantially via parallelization, albeit the approach has no theoretical guarantee on the resulting model quality. This leads to the question: can we develop a method that enjoys both the *parallelizability*, *small memory footprint* of the one-versus-all technique and the sub-linear complexity of *primal-dual sparse* method?

In this section, we propose a *greedy algorithm* that enjoys both the low runtime complexity inherent in the primal-dual sparse approach *and* one-versus-all's simple parallelization training with small memory footprint. In particular:

- We show that the loss optimized by the common one-versus-all technique also enjoys a similar *primal-dual sparse* structure presented in [118] when a class-wise bias is added.
- Then we propose a *greedy active-set algorithm* that optimizes parameters of each class separately *without communication* and thus enjoys both parallelizability and primal-dual sparsity.
- We then extend the analysis we did in last section in two ways: (i) bounding the number of non-zero dual variables in terms of the *number of positive samples* instead of the *number of confusing samples* that could be growing linearly with the total number of samples for the separable loss considered; (ii) bounding not only for the optimal solution but also for any descent iterates during the optimization, which leads to a more realistic analysis for the sub-linear complexity of the algorithm.
- In our experiments on several benchmark data sets with hundreds of thousands of classes, the new approach achieves accuracy competitive with state-of-the-art. On a cluster of 100 cores, our method is orders-of-magnitude faster than the existing parallel one-versus-all methods and the sequential PD-Sparse algorithm.

2.2.1 Formulation

A significant disadvantage of the loss (2.2) is that it is not separable w.r.t. the class parameters $\boldsymbol{w}_1, ..., \boldsymbol{w}_K$, and therefore it requires training all parameters W together. This incurs a much larger memory consumption than the *one-versus-all* approach even in the presence of sparsity. This prohibits *PD-Sparse* from using a simple parallelization scheme that assigns the training of different classes to different cores—a scheme that could enjoy nearly linear speedup to even a thousand of cores in [6].

To achieve parallelizability and space efficiency, we consider the following *class-separable hinge loss*

$$L(\boldsymbol{z}, \boldsymbol{y}) := \sum_{k=1}^{K} \ell(z_k, y_k) = \sum_{k=1}^{K} \max(1 - y_k z_k, 0) \tag{2.24}$$

One-versus-all method can be interpreted as minimizing (2.24) since

$$\min_{W \in \mathbb{R}^{D \times K}} \sum_{i=1}^{N} \sum_{k=1}^{K} \ell(\boldsymbol{w}_k^T x_i, y_{ik}) = \sum_{k=1}^{K} \left(\sum_{i=1}^{N} \ell(\boldsymbol{w}_k^T x_i, y_{ik}) \right) \tag{2.25}$$

The goal of this section is to show that (2.25) also permits a *primal-dual sparse* structure similar to [118] when a *bias* term is added to the parameters of each class. We first note that showing

(2.24) has dual sparsity is more difficult, since unlike (2.2), it penalizes each class separately, so there could potentially be many more active labels for each sample as we discuss below. We thus take two different approaches to show the desired dual sparsity structure.

Dual Sparsity: Active Labels We first employ an approach similar to [118], and try to establish dual sparsity in terms of the number of active labels of each sample i. In our case, the set of active labels are characterized as

$$\mathcal{C}_i := \{k \mid y_{ik}\langle \boldsymbol{w}_k, \boldsymbol{x}_i\rangle \leq 1\},$$

that is, labels of either wrong predictions or confidence scores ≤ 1 for a particular sample i. This is related to the set of support vectors of each class $\mathcal{S}_k := \{i \mid y_{ik}\langle \boldsymbol{w}_k, \boldsymbol{x}_i\rangle \leq 1\}$. Denote $k_a := \frac{1}{N}\sum_{i=1}^{N}|\mathcal{C}_i|$ and $n_a := \frac{1}{K}\sum_{k=1}^{K}|\mathcal{S}_k|$ as the average number of active labels and support vectors. We have

$$Nk_a = Kn_a.$$

Note when the number of active labels k_a is constant, we have $n_a = Nk_a/K$ decreases with K, which results in the *dual sparsity* when $K \gg k_a$.

The following theorem then shows that the dual sparsity also implies a primal-sparse solution when $n_a \ll D$.

Theorem 2. *Let $\lambda > 0$ be an arbitrarily small constant and*

$$\boldsymbol{w}^* \in \underset{\boldsymbol{w}\in\mathbb{R}^D}{argmin}\ \ \lambda\|\boldsymbol{w}\|_1 + \sum_{i=1}^{N}\ell(\boldsymbol{w}^T\boldsymbol{x}_i, y_i). \tag{2.26}$$

Then for $\{\boldsymbol{x}_i\}_{i=1}^N$ drawn from a continuous distribution *we have*

$$d_k := nnz(\boldsymbol{w}^*) \leq n_a. \tag{2.27}$$

where $n_a := \{i \mid y_i\langle \boldsymbol{w}^*, \boldsymbol{x}_i\rangle \leq 1\}$ is the number of support vectors.

Proof. Let X be an $N \times D$ feature matrix with rows $\{\boldsymbol{x}_i\}_{i=1}^N$. Any optimal solution of (2.26) satisfies

$$\lambda\boldsymbol{\rho}^* + \sum_{i=1}^{N}\alpha_i^*\boldsymbol{x}_i = \lambda\boldsymbol{\rho}^* + X^T\boldsymbol{\alpha}^* = 0 \tag{2.28}$$

for some $\boldsymbol{\rho}^* \in \partial(\|\boldsymbol{w}^*\|_1)$ and $\alpha_i^* \in \partial_{z_i}\ell(z_i, y_i)$ with $z_i := \langle \boldsymbol{w}^*, \boldsymbol{x}_i\rangle + b$. Then since hinge loss (and square hinge loss) has $\alpha_i^* \neq 0$ only when $y_i z_i^* \leq 1$, there are at most n_a non-zeros α_i^* in the linear system (2.28). On the other hand, the subgradient $\boldsymbol{\rho}^*$ of $\|\boldsymbol{w}^*\|_1$ satisfies

$$\boldsymbol{\rho}^* = \begin{cases} 1\ , & w_j^* > 0 \\ -1\ , & w_j^* < 0 \\ \nu\ , & \nu \in [-1, 1], w_j^* = 0. \end{cases}$$

Let $\mathcal{B} := \{j \mid w_j^* \neq 0\}$ be the indexes with non-zeros weights. Then consider equations given by the non-zeros α_is and $|\mathcal{B}|$ rows in (2.28)

$$-\lambda\ sign([\boldsymbol{w}^*]_{\mathcal{B}}) = X^T\boldsymbol{\alpha}^*.$$

It has $nnz(\boldsymbol{w}^*)$ equations and n_a variables, which, for a feature matrix at *general position*, can be satisfied only if

$$nnz(\boldsymbol{w}^*) = |\mathcal{B}| \leq n_a.$$

A feature matrix X is always at *general position* if its rows $\{\boldsymbol{x}\}_{i=1}^N$ are drawn from a continuous distribution [95]. □

Note the Theorem 2 implies that when the number of active labels k_a is constant, both the number of non-zeros in the primal variables d_k and dual variables n_a are proportional to $\frac{Nk_a}{K}$. However, there are several drawbacks with Theorem 2. First, the loss (2.24) does not *force* k_a to be small as the max-margin loss (2.2) does, so in our case k_a could actually increase linearly with K. Second, Theorem 2 only analyzes the optimal solution $(\boldsymbol{\alpha}^*, \boldsymbol{w}^*)$, which cannot however guarantee the sparsity of $\boldsymbol{w}$ during the intermediate iterates of the training algorithm. Finally, the result (2.27) holds only for a purely ℓ_1-regularized objective, which is known to yield a non-smooth dual objective that can be hard to optimize in a coordinate-wise fashion. In next section, we thus employ a different approach to resolve these caveats.

Dual Sparsity: Positive Examples In this section, we take a more realistic approach which bounds the ℓ_1-norm of primal and dual variables by the *number of positive examples* of each class, which decreases as a function of K even under the separable loss (2.24). Denote n_p as the average number of positive samples per class, and k_p as the average number of positive labels per sample. We have

$$n_p := \left(\frac{k_p}{K}\right) N \ll N. \tag{2.29}$$

when K is large. Note unlike the number of active labels k_a, the number of positive labels k_p is a constant that does not grow with K in most of Extreme Classification problems. Therefore, the number of positive samples per class n_p is decreasing with K since $n_p = Nk_p/K$.

Now consider a model that incorporates the bias term, where we have an additional feature $x_{i,0} = 1$ for all samples, and an additional weight $w_{k,0}$ for all classes. For ease of optimization in the dual, we consider the following ℓ_1-ℓ_2 (Elastic-Net) regularized objective :

$$\min_{\boldsymbol{w}_k \in \mathbb{R}^D} \quad F(\boldsymbol{w}_k) := \lambda \sum_{j=1}^{D} |w_{jk}| + \frac{1}{2}\|\boldsymbol{w}_k\|^2 + \sum_{i=1}^{N} \ell(\boldsymbol{w}_k^T \boldsymbol{x}_i, y_{ik}), \tag{2.30}$$

which has a dual objective of the form

$$\begin{aligned} \min_{\boldsymbol{\alpha}_k \in \mathbb{R}^N} \quad & G(\boldsymbol{\alpha}_k) := \frac{1}{2}\|\boldsymbol{w}(\boldsymbol{\alpha}_k)\|^2 - \sum_{i=1}^{N} \alpha_{ik} \\ s.t. \quad & \boldsymbol{w}(\boldsymbol{\alpha}_k) = \mathbf{prox}_\lambda(\hat{X}^T \boldsymbol{\alpha}_k), \\ & 0 \leq \alpha_{ik} \leq 1. \end{aligned} \tag{2.31}$$

where $\hat{X}$ is $N \times D$ matrix with rows $\{y_{ik}\boldsymbol{x}_i\}_{i=1}^N$, and $\mathbf{prox}_\lambda(.)$ is the proximal operator of the function $\lambda \sum_{j=1}^D |w_j|$. Note we do not penalize bias term in the ℓ_1 regularization, so $w_0 = [\hat{X}^T \boldsymbol{\alpha}_k]_0$. Then the following two theorems bound the ℓ_1-norm of primal and dual parameters respectively.

Theorem 3 (ℓ_1-norm of primal variables)**.** *Let $\hat{\boldsymbol{w}} = (\hat{w}_0 = -1, \hat{\boldsymbol{w}}_{-0} = \mathbf{0})$ be a trival solution. For any $\boldsymbol{w}_k$ with $F(\boldsymbol{w}_k) \leq F(\hat{\boldsymbol{w}})$, we have*

$$\|\boldsymbol{w}_k\|_1 \leq \frac{2n_p^k}{\lambda}. \tag{2.32}$$

where n_p^k is the number of positive samples of k-th class.

Proof. $\hat{\boldsymbol{w}}$ satisfies

$$F(\hat{\boldsymbol{w}}) = \frac{1}{2} + n_p^k \leq 2n_p^k$$

since $\ell(-1, y_{ik}) = 0$ for $y_{ik} = -1$ and $\ell(0, y_{ik}) = 1$ for $y_{ik} = 1$. Then for any $\boldsymbol{w}_k$ of better objective than $\hat{\boldsymbol{w}}$,

$$\lambda\|\boldsymbol{w}_k\|_1 \leq F(\boldsymbol{w}_k) \leq F(\hat{\boldsymbol{w}}) \leq 2n_p^k$$

which yields the result. □

Theorem 4 (ℓ_1-norm of dual variables)**.** *Any optimal solution $\boldsymbol{\alpha}_k$ of* (2.31) *satisfies*

$$\|\boldsymbol{\alpha}_k^*\|_1 \leq 4n_p^k.$$

Proof. Let $\boldsymbol{w}$, $\boldsymbol{\alpha}$ be the primal and dual variables for a particular class. By strong duality, the optimal dual objective (in its maximization form) equals to the optimal primal objective

$$\max_{\boldsymbol{\alpha}} -G(\boldsymbol{\alpha}) = \min_{w} F(\boldsymbol{w})$$

and therefore, any optimal solution $(\boldsymbol{w}^*, \boldsymbol{\alpha}^*)$ satisfies

$$-\frac{1}{2}\|\boldsymbol{w}^*\|^2 + \sum_i \alpha_i^* = \frac{1}{2}\|\boldsymbol{w}^*\|^2 + \lambda\sum_{j=1}^{D}|w_j^*| + \sum_{i=1}^{N}\ell(\langle \boldsymbol{w}^*, \boldsymbol{x}_i\rangle, y_{ik}),$$

which yields the bound

$$\begin{aligned}\|\boldsymbol{\alpha}^*\|_1 &= \|\boldsymbol{w}^*\|^2 + \lambda\sum_{j=1}^{D}|w_j^*| + \sum_{i=1}^{N}\ell(\langle \boldsymbol{w}^*, \boldsymbol{x}_i\rangle, y_{ik}) \\ &\leq 2\left(\frac{1}{2}\|\boldsymbol{w}^*\|^2 + \lambda\sum_{j=1}^{D}|w_j^*| + \sum_{i=1}^{N}\ell(\langle \boldsymbol{w}^*, \boldsymbol{x}_i\rangle, y_{ik})\right) \leq 4n_p^k\end{aligned}$$

where the last inequality is due to the existence of $\hat{\boldsymbol{w}} = (\hat{w}_0 = -1, \hat{\boldsymbol{w}}_{-0} = 0)$ with primal objective $F(\hat{\boldsymbol{w}}) \leq 2n_p^k$. □

Theorem 3 and 4 successfully bound the ℓ_1-norm of primal, dual variables by $n_p = O(N/K)$. Although a small ℓ_1-norm does not imply small number of non-zeros in general, we show in the next section that the *complexity* of our algorithm is determined by the ℓ_1 norm of primal and dual variables. In particular, in Section 2.2.2, we propose a random sparsification procedure that finds a sparse approximation to $\boldsymbol{w}_k$ in order to perform efficient greedy search of active coordinates. The procedure is guaranteed to find a sparse solution with number of non-zeros proportional to $\|\boldsymbol{w}_k\|_1^2$. Further, we give an iteration complexity, and thus a bound on the active size, proportional to $\|\boldsymbol{\alpha}^*\|_1^2$, where $\boldsymbol{\alpha}^*$ is an optimal solution of (2.31).

2.2.2 Algorithm

In this section, we propose a greedy algorithm that alternates between the search of potential support vectors and the optimization over a small set of active samples. Note we have an objective (2.30) for each class k independent of other classes, so in the following we simply use $\boldsymbol{\alpha}$ to denote $\boldsymbol{\alpha}_k$ and $\boldsymbol{w}$ to denote $\boldsymbol{w}_k$ for a particular class k being solved. The algorithm will be performed for each class independently *without communication*, and thus it is inherently easy to parallelize.

A Greedy Active-Set Method Instead of searching for the *confusing labels* of each sample as in [118], we propose a greedy algorithm that optimizes (2.31) by looking for *active samples* of each class. Note the domain of our objective (2.31) has box constraints $0 \leq \alpha_i \leq 1$ instead of simplex constraints as in [118], so a Frank-Wolfe-based algorithm used in [118] does not lead to sparse iterates. Here we propose a *greedy active-set coordinate descent* algorithm that alternates between the greedy search of novel active variables and the optimization over an active set.

The objective (2.31) has gradient of the form

$$\nabla G(\boldsymbol{\alpha}) = \hat{X}^T \boldsymbol{w}(\boldsymbol{\alpha}) - 1, \tag{2.33}$$

which can be evaluated in time $O\left(nnz(\boldsymbol{w})\bar{n}\right)$ if the vector $\boldsymbol{w}(\boldsymbol{\alpha})$ is maintained whenever any dual coordinate α_i is changed, where $\bar{n}$ is an upper bound on the number of non-zero in each column of X. On the other hand, when $\boldsymbol{\alpha}$ is changed by $\Delta\boldsymbol{\alpha}$, the maintenance of

$$\boldsymbol{w}(\boldsymbol{\alpha} + \Delta\boldsymbol{\alpha}) = \mathbf{prox}_{\lambda\|.\|_1}(\hat{X}^T\boldsymbol{\alpha} + \hat{X}^T\Delta\boldsymbol{\alpha}) \tag{2.34}$$

requires a cost of $O(nnz(\Delta\boldsymbol{\alpha})\bar{d})$ where $\bar{d}$ is an upper bound on the number of non-zeros for each row of $\hat{X}$. Note one can exploit the sparsity of both $\hat{X}$ and $\boldsymbol{w}(\boldsymbol{\alpha})$ simultaneously when computing the gradients of all coordinates together as

$$\nabla G(\boldsymbol{\alpha}) = \sum_{j=0}^{D} w_j \hat{X}_{:,j}. \tag{2.35}$$

Therefore, an efficient algorithm exploits (2.35) to compute gradients of all coordinate simultaneously while updates only a small number of them to ensure a small $nnz(\Delta\alpha)$. This suggests a greedy strategy that optimizes only coordinates α_i leading to the most progress. The resulting algorithm is summarized in Algorithm 2, where we perform an inner minimization over active set via Randomized Dual Coordinate Descent (Algorithm 3) [43]. The cost of one epoch of Algorithm 3 over the active set is $O(|\mathcal{A}|\bar{d})$. Therefore, each iteration of Algorithm 2 has an overall cost of $O(nnz(\boldsymbol{w})\bar{n} + |\mathcal{A}|\bar{d})$. Note this is a cost sublinear to the size of data matrix $nnz(X)$ if $nnz(\boldsymbol{w}) \ll D$ and $nnz(\boldsymbol{\alpha}) \ll N$. To the end, we give bounds on $nnz(\boldsymbol{w})$ and $|\mathcal{A}|$ that decreases with number of classes K.

Sublinear-Time Search via Random Sparsification Given an ℓ_1 norm bounded by (2.32), we show that the maximum negative gradient found in Step 2 of Algorithm 2 can be approximated with δ precision when replacing $\boldsymbol{w}(\boldsymbol{\alpha})$ with its random sparsified version $\tilde{\boldsymbol{w}}$ obtained from Algorithm 4, where the number of non-zeros in $\tilde{\boldsymbol{w}}$ is bounded by the square of ℓ_1-norm as stated in the following theorem.

Algorithm 2 Greedy Active-Set Algorithm

0. $\boldsymbol{\alpha} = 0$, $\boldsymbol{b} = \mathbf{1}$, $\mathcal{A} = \{i | y_{ik} = 1\}$ and $H_{ii} = \|\boldsymbol{x}_i\|^2, i \in [N]$.
for t=1.....T **do**
1. Compute $\nabla G(\boldsymbol{\alpha})$ via random sparsification (Algorithm 4).
2. $\mathcal{A} \leftarrow$ Pick κ variables $\notin \mathcal{A}$ of largest $-\nabla_{\alpha_i} G(\boldsymbol{\alpha})$.
3. Minimize (2.31) w.r.t. coordinates in $\mathcal{A}$ via Algorithm 3.
4. Eliminate $\{i \mid \alpha_i = 0 \ \& \ y_{ik} \neq 1\}$ from $\mathcal{A}$.
end for

Algorithm 3 Coordinate Descent for Acitve Subproblem

for s=1.....S **do**
1. Draw $i \in \mathcal{A}$ uniformly at random.
2. Compute $\nabla_i G(\boldsymbol{\alpha}) = y_i \langle \boldsymbol{w}, \boldsymbol{x}_i \rangle - 1$.
3. $\Delta\alpha_i \leftarrow \min(\max(\alpha_i - \nabla_i G(\boldsymbol{\alpha})/H_{ii}, 0), U) - \alpha_i$
4. Maintain (2.34) with update $\Delta\alpha_i$.
end for

Theorem 5. *Running the Random Sparsification procededure 4 for $R = \lceil \frac{2\|\boldsymbol{w}\|_1^2}{\delta^2} \rceil$ iterations gives a $\tilde{\boldsymbol{w}}$ satisfying*

$$nnz(\tilde{\boldsymbol{w}}) \leq \left(\frac{4n_p^2}{\lambda^2} \right) \frac{1}{\delta^2}, \tag{2.36}$$

with

$$\mathcal{E}[\min_i \ y_i \langle \tilde{\boldsymbol{w}}, \boldsymbol{x}_i \rangle] - \min_i \ y_i \langle \boldsymbol{w}, \boldsymbol{x}_i \rangle \leq \delta, \tag{2.37}$$

Proof. Since the function $f(\boldsymbol{z}) = \min_i \ z_i$ is 1-Lipschitz-continuous, we have

$$\min_i \ y_i \langle \tilde{\boldsymbol{w}}, \boldsymbol{x}_i \rangle - \min_i \ y_i \langle \boldsymbol{w}, \boldsymbol{x}_i \rangle \leq |\langle \tilde{\boldsymbol{w}}, \boldsymbol{x}_i \rangle - \langle \boldsymbol{w}, \boldsymbol{x}_i \rangle|.$$

Taking expectation over $\tilde{\boldsymbol{w}}$ on both sides, we have

$$\begin{aligned} \mathcal{E}[\min_i \ y_i \langle \tilde{\boldsymbol{w}}, \boldsymbol{x}_i \rangle] - \min_i \ y_i \langle \boldsymbol{w}, \boldsymbol{x}_i \rangle &\leq \mathcal{E}[|\langle \tilde{\boldsymbol{w}}, \boldsymbol{x}_i \rangle - \langle \boldsymbol{w}, \boldsymbol{x}_i \rangle|] \\ &\leq \sqrt{\mathcal{E}[|\langle \tilde{\boldsymbol{w}}, \boldsymbol{x}_i \rangle - \langle \boldsymbol{w}, \boldsymbol{x}_i \rangle|^2]} \end{aligned} \tag{2.38}$$

from Jensen's inequality. Since $\mathcal{E}[\langle \tilde{\boldsymbol{w}}, \boldsymbol{x}_i \rangle] = \langle \boldsymbol{w}, \boldsymbol{x}_i \rangle$ by the construction of $\tilde{\boldsymbol{w}}$ in Algorithm 4, the RHS of (C.19) corresponds to the square root of variance

$$\mathcal{E}[\min_i \ \langle \tilde{\boldsymbol{w}}, \boldsymbol{x}_i \rangle] \leq \sqrt{Var[\langle \tilde{\boldsymbol{w}}, \boldsymbol{x}_i \rangle]} = \frac{\|\boldsymbol{w}\|_1}{\sqrt{R}}.$$

The conclusion follows by noticing that $nnz(\tilde{\boldsymbol{w}}) \leq R$ and $\|\boldsymbol{w}\|_1$ satisfies (2.32). □

Note (2.37) implies that the greedy coordinate $\hat{i}$ found by approximate search and the coordinate found by exact search $i^* = arg\min_i \ y_i \langle \boldsymbol{w}, \boldsymbol{x}_i \rangle - 1$ satisfy

$$\mathcal{E}[\nabla_{\hat{i}} G(\boldsymbol{\alpha})] \leq \nabla_{i^*} G(\boldsymbol{\alpha}) + 2\delta \tag{2.39}$$

Algorithm 4 Random Sparsification

INPUT: a vector $\boldsymbol{w} \in \mathbb{R}^D$.
0. $\tilde{\boldsymbol{w}}^0 = 0$.
for $r = 1...R$ **do**
1. Draw $j \in [D]$ with probability $|w_j|/\|\boldsymbol{w}\|_1$.
2. $\tilde{\boldsymbol{w}}^{(r)} \to \tilde{\boldsymbol{w}}^{(r-1)} + sign(w_j)\boldsymbol{e}_j$
end for
OUTPUT: $\tilde{\boldsymbol{w}} := \frac{\|\boldsymbol{w}\|_1}{R}\tilde{\boldsymbol{w}}^{(R)}$

Therefore, by replacing $\boldsymbol{w}$ with $\tilde{\boldsymbol{w}}$, we reduce the cost of gradient computation from the worst-case $O(nnz(\boldsymbol{w})\bar{n}) = O(D\bar{n})$ to $O(nnz(\tilde{\boldsymbol{w}})\bar{n})$ with a 2δ approximation error. In the next section, we will show that setting $\delta = O(N\hat{\epsilon})$ suffices for the global convergence of our Greedy Algorithm 2 to $\frac{1}{N}(G(\boldsymbol{\alpha}) - G^*) \leq \hat{\epsilon}$ for some $\hat{\epsilon} \in (0, 1)$. Therefore, we have

$$nnz(\tilde{\boldsymbol{w}}) = O(\frac{n_p^2}{\lambda^2 N^2 \hat{\epsilon}^2}) = O(\frac{k_p^2}{\lambda^2 K^2 \hat{\epsilon}^2}).$$

which could be much less than D in the Extreme Classification setting.

Convergence Analysis In this section, we give an iteration complexity of Algorithm 2 that depends on the ℓ_1 norm of the optimal solution $\boldsymbol{\alpha}^*$. For simplicity of the analysis, we assume that a normalized feature matrix with $\|\boldsymbol{x}_i\| \leq 1$.

Theorem 6. *Let $\boldsymbol{\alpha}^*$ be an optimal solution of* (2.31)*. The iterates $\{\boldsymbol{\alpha}^t\}_{t=1}^{\infty}$ given by Algorithm 2 with Random Sparsification tolerance $\delta \leq \frac{\epsilon}{4\|\boldsymbol{\alpha}^*\|1}$ has $\mathcal{E}[G(\boldsymbol{\alpha}^t)] - G(\boldsymbol{\alpha}^*) \leq \epsilon$ for any iterate*

$$t \geq \frac{4\|\boldsymbol{\alpha}^*\|_1^2}{\epsilon} + \frac{G(\mathbf{0}) - G^*}{\|\boldsymbol{\alpha}^*\|_1^2}.$$

Proof. Our dual objective (2.31) is of the form

$$g(\boldsymbol{\alpha}) + h(\boldsymbol{\alpha})$$

where $h(\boldsymbol{\alpha}) := \sum_{i=1}^N h_i(\alpha_i)$ and

$$h_i(\alpha) = \begin{cases} 0, & 0 \leq \alpha \leq 1 \\ \infty, & o.w. \end{cases}$$

and $g(\boldsymbol{\alpha}) = \frac{1}{2}\|\boldsymbol{w}(\boldsymbol{\alpha})\|^2 - \sum_{i=1}^N \alpha_i$ is smooth with Lipschitz-continuous gradient $\nabla g(\boldsymbol{\alpha})$. To see this, let $B_\alpha = \{j | w_j(\boldsymbol{\alpha}) \neq 0\}$. The generalized Hessian of $g(\boldsymbol{\alpha})$ is

$$\nabla^2 g(\boldsymbol{\alpha}) = X_{:,B_\alpha} X_{:,B_\alpha}^T$$

which has diagonal elements $\|\boldsymbol{x}_{i,B_\alpha}\|^2$ bounded by $\|\boldsymbol{x}_i\|^2 \leq 1$ and thus a spectral norm bounded by N. Then for any coordinate i, we have the following descent amount since the second derivative of the smooth part of objective is bounded by $\|\boldsymbol{x}_i\|^2 \leq 1$.

$$\min_\eta \ G(\boldsymbol{\alpha} + \eta \boldsymbol{e}_i) - G(\boldsymbol{\alpha}) \leq \min_\eta \ \nabla_i G * \eta + \frac{1}{2}\eta^2 + h_i(\alpha_i + \eta) \tag{2.40}$$

where $\boldsymbol{e}_i$ is an indicator vector. Note for $i \in \mathcal{A}$, the minimizer of RHS of (A.1) is 0 since the previous iteration already minimizes our objective w.r.t. the active set $\mathcal{A}$. And for $i \notin \mathcal{A}$, the minimizer of the RHS of (A.1) is $-[-\nabla_i G(\boldsymbol{\alpha})]_+^2/2$, which corresponds to the selection criteria at Step 2 of Algorithm 2. Therefore, at each iteration, the coordinate $\hat{i}$ and i^* found by the approximate and exact greedy search respectively satisfy

$$\begin{aligned}
&\mathcal{E}[G(\boldsymbol{\alpha}+\Delta\boldsymbol{\alpha})] - G(\boldsymbol{\alpha}) \leq \mathcal{E}[\min_{\eta}\ G(\boldsymbol{\alpha}+\eta\boldsymbol{e}_{\hat{i}})] - G(\boldsymbol{\alpha}) \\
&\leq \min_{\eta}\ \nabla_{i^*} G * \eta + \frac{1}{2}\eta^2 + h_i(\alpha_i+\eta) + 2\delta\eta \\
&\leq \min_{\boldsymbol{\eta}}\ \langle \nabla G, \boldsymbol{\eta}\rangle + \frac{1}{2}\|\boldsymbol{\eta}\|_1^2 + h(\boldsymbol{\alpha}+\boldsymbol{\eta}) + 2\delta\sum_i \eta_i
\end{aligned}$$

where the first inequality is because the update $\Delta\boldsymbol{\alpha}$ is obtained by minimizing objective w.r.t. a working set $\mathcal{A}^{r+1}$ containing $\hat{i}$, and the second, third inequalities follow from (2.39) and the fact that a linear objective subject to ℓ_1 ball has minimizer at the corner respectively. Then we use convexity to obtain a global estimate of descent amount relative to the suboptimality $G(\boldsymbol{\alpha}) - G(\boldsymbol{\alpha}^*)$:

$$\mathcal{E}[G(\boldsymbol{\alpha}+\Delta\boldsymbol{\alpha})] - G(\boldsymbol{\alpha}) \tag{2.41}$$

$$\leq \min_{\boldsymbol{\eta}}\ \langle \nabla G, \boldsymbol{\eta}\rangle + \frac{1}{2}\|\boldsymbol{\eta}\|_1^2 + h(\boldsymbol{\alpha}+\boldsymbol{\eta}) + 2\delta\sum_i \eta_i \tag{2.42}$$

$$\leq \min_{q\in[0,1]}\ q(\nabla G, \boldsymbol{\alpha}^* - \boldsymbol{\alpha}) + \frac{q^2}{2}\|\boldsymbol{\alpha}^*\|_1^2 + 2q\delta\|\boldsymbol{\alpha}^*\|_1 \tag{2.43}$$

$$\leq \min_{q\in[0,1]}\ -q(G(\boldsymbol{\alpha}) - G(\boldsymbol{\alpha}^*)) + \frac{q^2}{2}\|\boldsymbol{\alpha}^*\|_1^2 + 2q\delta\|\boldsymbol{\alpha}^*\|_1 \tag{2.44}$$

where the last ineqaulity is from convexity, and the second inequality is from a restriction of optimization space to $\boldsymbol{\eta} = q(\boldsymbol{\alpha}^* - \boldsymbol{\alpha})$ and the fact that for $i \in \mathcal{A}$ the minimizer of (C.20) has $\eta_i = 0$. Then choosing $\delta \leq \frac{G(\boldsymbol{\alpha})-G(\boldsymbol{\alpha}^*)}{4\|\boldsymbol{\alpha}^*\|_1}$ and minimizing the RHS w.r.t. q leads to

$$\mathcal{E}[G(\boldsymbol{\alpha}+\Delta\boldsymbol{\alpha})] - G(\boldsymbol{\alpha}) \leq -\frac{(G(\boldsymbol{\alpha}) - G(\boldsymbol{\alpha}^*))^2}{4\|\boldsymbol{\alpha}^*\|_1^2} \tag{2.45}$$

for iterates with $G(\boldsymbol{\alpha}) - G(\boldsymbol{\alpha}^*) \leq 2\|\boldsymbol{\alpha}^*\|_1^2$ and has

$$\mathcal{E}[G(\boldsymbol{\alpha}+\Delta\boldsymbol{\alpha})] - G(\boldsymbol{\alpha}) \leq -\|\boldsymbol{\alpha}^*\|_1^2/2 \tag{2.46}$$

for iterates with $G(\boldsymbol{\alpha}) - G(\boldsymbol{\alpha}^*) > 2\|\boldsymbol{\alpha}^*\|_1^2$. Note the constant descent amount (2.46) happens only in the beginning iterates when $G(\boldsymbol{\alpha}) - G(\boldsymbol{\alpha}^*)$ and can happen at most $2(G(\mathbf{0}) - G(\boldsymbol{\alpha}^*))/\|\boldsymbol{\alpha}^*\|_1^2$ times. Considering those iterates of case (2.45), we have recursive relation $\Delta G^{t+1} - \Delta G^t \leq -\frac{(\Delta G^t)^2}{4\|\boldsymbol{\alpha}^*\|_1^2}$ where $\Delta G^t := \mathcal{E}[G(\boldsymbol{\alpha}^t) \mid \boldsymbol{\alpha}^{t-1}] - G(\boldsymbol{\alpha}^*)$. The recursion leads to the conclusion by, for example, Theorem 1 of [81]. □

Theorem 6 is significant when combined with Theorem 4, which gives us an iteration complexity of

$$t = \frac{4\|\boldsymbol{\alpha}^*\|_1^2}{\epsilon} \leq \frac{64n_p^2}{\epsilon},$$

and also a bound on the active size $|\mathcal{A}| \leq \kappa t \leq \kappa \frac{64n_p^2}{\epsilon}$ that depends only on the number of positive examples. Considering the average case where $n_p = Nk_p/K$, for achieving $\frac{1}{N}(G(\boldsymbol{\alpha}) - G^*) \leq \hat{\epsilon} \in (0, 1)$, we have

$$|\mathcal{A}| = O(\frac{Nk_p^2}{K^2\hat{\epsilon}}).$$

Then the complexity for running Algorithm 2 on all classes is:

$$K * O(nnz(\tilde{\boldsymbol{w}})\bar{n} + |\mathcal{A}|\bar{d}) = O\left(\frac{k_p^2\bar{n}}{K\lambda^2\hat{\epsilon}^2} + nnz(X)\frac{k_p^2}{K\hat{\epsilon}}\right),$$

times the number of iterations, which is a cost proportional to the factor k_p^2/K.

2.2.3 Experiments

Table 2.4: Results and statistics for large-scale Multilabel data sets, N_{train}= number of training samples, N_{test}= number of testing samples, T_{train} = training time, T_{test} = testing time, K = number of classes, D = number of features. P@k = top-k accuracy. DiSMEC and PPDSparse are parallelized with 100 cores. We highlight the best result for each metric, except that for T_{train} we highlight best results among single-core solvers (left four) and parallel solvers. For all experiments, we set a memory limit to be 100G. Experiments that exceeded limits are marked *Memory Limit Exceeded* (MLE).

Data	Metrics	FastXML	PfastreXML	SLEEC	PDSparse	DiSMEC	PPDSparse
Amazon-670K	T_{train}	**5624s**	6559s	20904s		174135s	**921.9s**
N_{train}=490449	P@1 (%)	33.12	32.87	35.62		43.00	**43.04**
N_{test}=153025	P@3 (%)	28.98	29.52	31.65	MLE	38.23	**38.24**
D=135909	P@5 (%)	26.11	26.82	28.85		34.93	**34.94**
K=670091	model size	**4.0G**	6.3G	6.6G		8.1G	5.3G
	T_{test}/N_{test}	**1.41ms**	1.98ms	6.94ms		148ms	20ms
WikiLSHTC-325K	T_{train}	**19160s**	20070s	39000s	94343s	271407s	**353s**
N_{train}=1778351	P@1 (%)	50.01	57.17	58.34	60.70	64.00	**64.13**
N_{test}=587084	P@3 (%)	32.83	37.03	36.7	39.62	**42.31**	42.10
D=1617899	P@5 (%)	24.13	27.19	26.45	29.20	**31.40**	31.14
K=325056	model size	14G	16G	650M	**547M**	8.1G	4.9G
	T_{test}/N_{test}	**1.02ms**	1.47ms	4.85ms	3.89ms	65ms	290ms
Delicious-200K	T_{train}	8832.46s	8807.51s	**4838.7s**	5137.4s	38814s	**2869s**
N_{train}=196606	P@1 (%)	**48.85**	26.66	47.78	37.69	44.71	45.05
N_{test}=100095	P@3 (%)	**42.84**	23.56	42.05	30.16	38.08	38.34
D=782585	P@5 (%)	**39.83**	23.21	39.29	27.01	34.7	34.90
K=205443	model size	1.3G	20G	2.1G	**3.8M**	18G	9.4G
	T_{test}/N_{test}	1.28ms	7.40ms	2.685ms	**0.432ms**	311.4ms	275ms
AmazonCat-13K	T_{train}	11535s	13985s	119840s	**2789s**	11828s	**122.8s**
N_{train}=1186239	P@1 (%)	**94.02**	86.06	90.56	87.43	92.72	92.72
N_{test}=306782	P@3 (%)	**79.93**	76.24	76.96	70.48	78.11	78.14
D=203882	P@5 (%)	**64.90**	63.65	62.63	56.70	63.40	63.41
K=13330	model size	9.7G	11G	12G	**15M**	2.1G	355M
	T_{test}/N_{test}	1.21ms	1.34ms	13.36ms	0.87ms	**0.20ms**	1.82ms

Table 2.5: Results and statistics for small Multilabel data sets, N_{train}= number of training samples, N_{test}= number of testing samples, T_{train} = training time, T_{test} = testing time, K = number of classes, D = number of features. P@k = top-k accuracy. DiSMEC and PPDSparse are parallelized with 100 cores. We highlight the best result for each metric, except that for T_{train} we highlight best results among single-core solvers (left four) and parallel solvers.

Data	Metrics	FastXML	PfastreXML	SLEEC	PDSparse	DiSMEC	PPDSparse
Mediamill	T_{train}	276.4s	293.2s	9504s	**23.8s**	**12.15s**	34.1s
N_{train}=30993	P@1 (%)	84.27	84.08	**87.37**	83.64	84.83	84.42
N_{test}=12914	P@3 (%)	67.34	67.45	**72.60**	66.13	67.17	67.26
D=120	P@5 (%)	53.06	53.23	**58.39**	50.90	52.80	52.78
K=101	model size	87M	88M	104M	**20K**	412K	412K
	T_{test}/N_{test}	0.27ms	0.37ms	4.95ms	**0.004ms**	0.142ms	0.078ms
Bibtex	T_{train}	21.68s	21.47s	296.86s	**7.71s**	**0.203s**	0.232s
N_{train}=4880	P@1 (%)	63.66	63.18	**64.77**	62.36	63.69	63.69
N_{test}=2515	P@3 (%)	39.42	**39.67**	38.97	36.50	38.80	39.43
D=1836	P@5 (%)	28.60	**29.47**	28.50	26.50	28.30	28.67
K=159	model size	34M	37M	5.2M	**20K**	2.1M	2.5M
	T_{test}/N_{test}	0.64ms	0.73ms	0.70ms	**0.007ms**	0.28ms	0.094ms
RCV1-2K	T_{train}	4874.4s	4947.2s	85212s	**709.5s**	641.1s	**35.0s**
N_{train}=623847	P@1 (%)	91.14	89.79	**91.36**	90.02	90.52	91.08
N_{test}=155962	P@3 (%)	73.35	72.65	**73.38**	71.92	72.31	72.93
D=47236	P@5 (%)	**52.69**	52.23	52.50	51.23	51.25	52.10
K=2456	model size	3.9G	4.1G	1.1G	**1.6M**	209M	23M
	T_{test}/N_{test}	0.87ms	1.08ms	53.95ms	**0.066ms**	1.72ms	0.338ms
EURLex-4K	T_{train}	**315.9s**	324.4s	4543.4s	773.2s	76.07s	**9.95s**
N_{train}=15539	P@1 (%)	70.86	70.33	**79.15**	75.90	70.61	74.61
N_{test}=3809	P@3 (%)	59.06	58.61	**64.09**	61.16	57.56	59.56
D=5000	P@5 (%)	49.58	49.69	**52.09**	50.83	47.33	48.43
K=3993	model size	384M	455M	121M	25M	15M	**9.5M**
	T_{test}/N_{test}	3.65ms	5.43ms	3.67ms	**0.73ms**	2.26ms	1.5ms

In this section, we compare our proposed algorithm with state-of-the-art approaches on multiclass and multilabel problems chosen based on the experimental results shown in the Extreme Classification Repository [5] and [6, 118]. The compared methods are:

- **FastXML** [77]: An efficient and scalable tree-based algorithm. We adopted parameter setting suggested by the solver.
- **PfastreXML** [45]: An efficient and scalable tree ensemble based method improving upon FastXML by minimizing propensity-scored loss at each tree node, which leads to better performance on tail labels. Since all other methods are not adjusted based on the propensity, in our experiment we still measure performance via traditional top-k accuracy.
- **SLEEC** [8]: A non-linear solver that 1) partitions training sample into clusters and 2) compute local embeddings that preserves nearest neighbor structure within each cluster. Because of this composition of components, its performance highly relies on its parameter setting. We adopted settings suggested by the authors for each data set.
- **PDSparse** [118]: A Primal-Dual sparse method that minimizes a max-margin loss with ℓ_1-ℓ_2 regularization and enjoys sublinear complexity w.r.t. the number of classes.
- **DiSMEC** [6]: A distributed and parallelized method that learns one-versus-all classifiers with heuristic model compression (weight truncation). We run this method with 100 cores

[5] `https://manikvarma.github.io/downloads/XC/XMLRepository.html`

Table 2.6: Results and statistics for Multiclass data sets, N_{train}= number of training samples, N_{test}= number of testing samples, T_{train} = training time, T_{test} = testing time, K = number of classes, D = number of features. DiSMEC and PPDSparse are parallelized with 100 cores. We highlight the best result for each metric, except that for T_{train} we highlight best results among single-core solvers (left four) and parallel solvers.

Data	Metrics	FastXML	PfastreXML	SLEEC	PDSparse	DiSMEC	PPDSparse
aloi.bin	T_{train}	1900.9s	1901.6s	16193s	**139.8s**	92.0s	**7.05s**
N_{train}=100000	accuracy (%)	95.71	93.43	93.74	96.2	96.28	**96.38**
N_{test}=8000	model size	1.3G	1.3G	3.7G	19M	16M	**14M**
D=636911	T_{test}/N_{test}	5.05ms	5.10ms	28.00ms	0.064ms	0.02ms	**0.0178ms**
K=1000							
LSHTC1	T_{train}	1398.2s	1422.4s	5919.3s	**196.6s**	298.8s	**45.8s**
N_{train}=88806	accuracy (%)	22.04	**23.32**	12.2	22.46	22.74	22.70
N_{test}=5000	model size	937M	1.1G	631M	**88M**	142M	381M
D=347255	T_{test}/N_{test}	5.73ms	8.81ms	14.66ms	**0.40ms**	3.7ms	6.94ms
K=12294							
Dmoz	T_{train}	6475.1s	6619.7s	47490s	**2518.9s**	1972.0s	**170.60s**
N_{train}=345068	accuracy (%)	**40.76**	39.78	33.03	39.91	39.38	39.32
N_{test}=38340	model size	3.5G	3.8G	1.5G	**680M**	369M	790M
D=833484	T_{test}/N_{test}	3.29ms	3.20ms	40.43ms	**1.87ms**	4.58ms	6.58ms
K=11947							

using the same parallelization framework to our solver.

- **PPDsparse**: The proposed method with 100 cores (10 machines with 10 cores on each machine).

All compared solvers are available from Extreme Classification RepositoryXMLRepo. Other solvers not compared in this paper are i) *one-vs-all logistic regression*, *one-vs-all SVM*, *multiclass SVM*, *one-vs-all ℓ_1-regularized logistic regression*, implemented in *LibLinear* [28], ii) *Vowpal-Wabbit* [18], iii) *LEML* [125], iv) *RobustXML* [107], v) *PLT* [46], vi) *LPSR-NB* [105]. All of these have been shown less competitive in a number of previous papers [6, 118].

Experiments are conducted on four large-scale multilabel data sets, four medium-scale multilabel data sets and three large-scale multiclass data sets. We adopt data sets used by [6, 45, 77, 118] and also that from the Extreme Classification Repository XMLRepo. Large-scale multilabel data sets are *WikiLSHTC-325K*, *Delicious-200K*, *Amazon-670K* and *AmazonCat-13K*. Medium-scale Multilabel data sets are *Mediamill*, *Bibtex*, *RCV1-2K* and *EURLex-4K*. They can be found at Extreme Classification RepositoryXMLRepo. Multiclass data sets *LSHTC1*, *Dmoz* and *aloi.bin* are available [6] from authors of [118].

The data statistics and results are shown in Table 2.4, 2.5 and 2.6. For all experiments, we select our hyperparameter λ from $\{0.01, 0.1, 1\}$ and τ from $\{0.1, 1, 10\}$ to maximize the heldout performance. However, we observed that for most of data sets, $\tau = 1$, $\lambda = 0.01$ consistently gives the best performance. For large-scale multilabel datasets (Table 2.4), we use tf-idf features with sample-wise normalization as suggested by the author of [6].

Our experimental results confirmed several comments from previous work [6, 118]: 1) Among single-core solvers (PDSparse, FastXML, PfastreXML, SLEEC), PDSparse can achieve orders of magnitude speed up in terms of training time without significantly downgrading performance compared to the direct one-vs-all approach (except on Delicious-200k, a data set of missing la-

[6] `http://www.cs.utexas.edu/~xrhuang/PDSparse/`

bels) . 2) Given enough computing resources, DiSMEC is able to achieve significantly higher accuracy than other approaches on some data sets. However, their training on the largest data sets typically take few days even with 100 cores.

From Table 2.4-2.6, we illustrate how our proposed PPDSparse method combines the strength from both PDSparse and DiSMEC. By adopting one-versus-all loss, PPDSparse can achieve accuracy as good as DiSMEC on most of data sets and resolve drawbacks of the max-margin loss used by PDSparse in three ways: (i) PPDSparse reduces the memory requirement of PDSparse by orders of magnitude due to the separation of training of each class, which clears the MLE issue of PDSparse on Amazon-670K, (ii) By embarassingly parallelized to 100 cores, PPDSparse is orders of magnitude faster than both PDSparse and parallel 1-vs-all (DiSMEC), (iii) the performance of PPDSparse is less sensitive to data set of mislabeling. On Delicious-200K, a data set of missing positive labels, PPDsparse improves accuracy of PDSparse significantly.

The training of PPDSparse is consistently faster than tree-based and local embedding methods by orders of magnitude while maintaining a competitive accuracy on most of data sets. On Amazon-670K and WikiLSHTC-325K, PPDSparse (and DisMEC) enjoy a significant increase in accuracy compared to tree-based approaches. On the other hand, the prediction speed of Primal Dual sparse approaches (PPDSparse, PDSparse) are slower than tree-based methods (FastXML, PfastreXML) on problems of more than 10^5 classes, while being comparable on medium-sized data sets of $10^3 - 10^4$ classes (Table 2.6), presumably because tree-based methods enjoy logarithmic-time prediction w.r.t. the number of classes.

2.3 Compression of Deep Neural Networks

In this section, we apply the theory of *primal and dual sparsity* to the problem of compressing a Deep Neural Network (DNN). This shows for the first time that: a simple ℓ_1 regularization gives surprisingly effective trimming of DNNs both theoretically and empirically, given a ℓ_1-friendly optimizer targeting *high precision* is used for the purpose of compression.

2.3.1 Problem Setup

Let $X^{(0)} : N \times D_1^{(0)} \cdots \times D_p^{(0)} \times K^0$ be an input tensor where N is the number of samples (or batch size). We are interested in DNNs of the form

$$X^{(j)} := \sigma_{W^{(j)}}(X^{(j-1)}), \quad l = j \dots J$$

where $\sigma_{W^{(j)}}(X^{(j-1)})$ are piecewise-linear functions of both the parameter tensor $W^{(j)} : K^{(j-1)} C_0^{(j)} \cdots \times C_p^{(j)} \times K^{(j)}$ and the input tensor $X^{(j-1)} : N \times D_1^{(j-1)} \cdots \times D_p^{(j-1)}$ of (j)-th layer. Examples of such piecewise-linear function are (i) convolution layer with Relu activation

$$[\sigma_W(X)]_{i,k} := [\sum_{m=1}^{K^{(j-1)}} X_{i,:,m} \circ W_{m,:,k}]_+,$$

where $\circ$ is p-dimensional convolution operator, (ii) fully-connected layer with Relu activation

$$[\sigma_W(X)]_{i,k} := [X_{i,:} W_{:,k}]_+,$$

and also other commonly used operations such as *max-pooling*, *zero-padding* and *reshaping*. Note $X^{(J)} : N \times K$ provide K scores (i.e. logits) of each sample that relate to the labels of our target task $Y : N \times K$. Denote $L(X^{(J)}, Y)$ as the task-specific loss function. We define *Support Labels* of a DNN $X^{(J)}$ as indices (i, k) of non-zero loss subgradient w.r.t. the prediction logit:

Definition 1 (Support Labels)**.** *Let $L(X, Y)$ be a convex loss function w.r.t. the prediction logits X. The* Support Labels *regarding DNN outputs $X^{(J)}(W)$ are defined as*

$$\mathcal{S}(W) := \left\{(i,k) \in [N] \times [K] \;\middle|\; [Q]_{i,k} \neq 0, \text{ for some } Q \in \partial_X L(X^{(J)}, Y)\right\}.$$

We will denote $k_{\mathcal{S}}(W) := \frac{|\mathcal{S}(W)|}{N} \leq K$ as the average number of support labels *per sample.*

Multi-Regression In a multi-regression task, we are interested in real-valued labels, such as the location and orientation of objects in an image, which can be expressed as an $N \times K$ real-valued matrix Y. The loss function

$$L(X, Y) := \frac{1}{2}\|X - Y\|_F^2$$

are convex and differentiable, and in general we have $[\nabla_X L]_{i,k} \neq 0$, therefore all labels are support labels (i.e. $k_{\mathcal{S}} = K$).

Binary Classification In a binary classification task, the labels are binary-valued, and can be represented as a binary vector $\boldsymbol{y} : \{-1, 1\}^N$, and typical loss functions are the *logistic loss*

$$L(\boldsymbol{x}, \boldsymbol{y}) := \sum_{i=1}^{N} \log(1 + \exp(-y_i x_i)) \tag{2.47}$$

and *hinge loss*

$$L(\boldsymbol{x}, \boldsymbol{y}) := \sum_{i=1}^{N} [1 - y_i x_i]_+, \tag{2.48}$$

For (2.47) we have $k_{\mathcal{S}} = 1$ since $[\nabla L]_i \neq 0, \forall i \in [N]$. On the other hand, the hinge loss (2.48) typically has only a small portion of samples $i \in [N]$ with $[\partial L]_i \neq \{0\}$, often called *Support Vectors* in the literature of *Support Vector Machine*. In this case, our definition of *Support Labels* coincides with that of *Support Vectors*. In many applications with unbalanced positive and negative examples, such as *object detection*, we have $k_{\mathcal{S}} \ll 1$.

Multiclass/Multilabel Classification In a multiclass or multilabel classification problem, the labels of each sample can be represented as a K-dimensional binary vector $\{0, 1\}^K$ where 1/0 denotes the presence/absence of a class in the sample. Let $\mathcal{P}_i := \{k \mid y_{ik} = 1\}$ and $\mathcal{N}_i := \{k \mid y_{ik} = 0\}$ denote the positive and negative label sets. Common loss functions are the cross-entropy loss

$$L(X, Y) := \sum_{i=1}^{N} \left(\log \sum_{k=1}^{K} \exp(X_{ik}) - \frac{1}{|\mathcal{P}_i|} \sum_{k \in \mathcal{P}_i} X_{ik} \right) \tag{2.49}$$

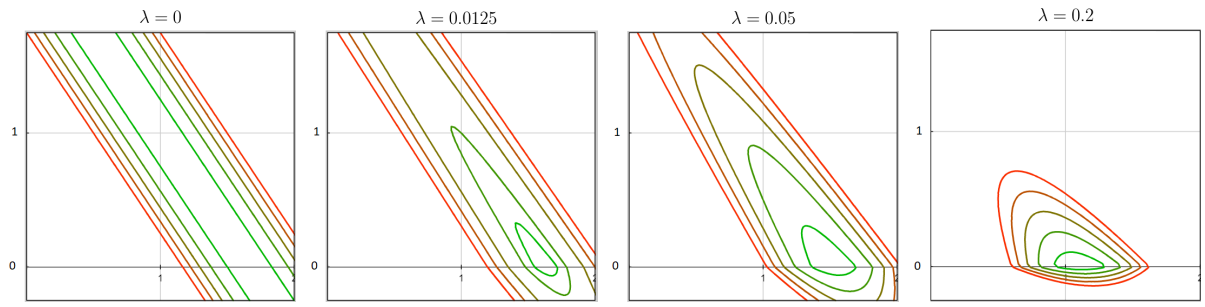

Figure 2.1: The level curves of the ℓ_1-regularized objective with 2 parameters and 1 sample: $\lambda\|\boldsymbol{w}\|_1 + \frac{1}{2}(\boldsymbol{x}^T\boldsymbol{w} - 1)^2$ where $\boldsymbol{x} = [0.6, 0.4]$. Note the stationary points have $w_2 = 0$ as long as $\lambda > 0$. However, smaller λ makes convergence to the stationary point harder for optimization algorithms.

and maximum margin loss

$$L(X, Y) := \sum_{i=1}^{N} \left(\max_{j \in \mathcal{N}_i, k \in \mathcal{P}_i} 1 + X_{ij} - X_{ik} \right). \tag{2.50}$$

Although the cross-entropy loss (3.38) has number of support labels $k_{\mathcal{S}} = K$, it has been shown that the maximum-margin loss (3.39) typically has $k_{\mathcal{S}} \ll K$ in recent studies of classification problems with extremely large number of classes [113, 118].

2.3.2 Deep-Trim: Theory and Algorithms

In this section, we aim to solve the following DNN compression problem.

Definition 2 (Deep-Trim)**.** *Given a target loss function $L(., Y)$ with training labels $Y : N \times K$, and a pre-trained DNN $X^{(J)}$ parameterized by $W := \{W^{(j)}\}_{j=1}^{J}$ of loss $L^* := L(X^{(J)}(W), Y)$, finds a compressed DNN with number of non-zero parameters $nnz(\hat{W}) \leq \tau$ and $L(X^{(J)}(\hat{W}), Y) \leq L^* + \epsilon$ for any $\epsilon > 0$.*

In the following, we show that the *Deep-Trim* problem with budget $\tau = (Nk_{\mathcal{S}}) \times J$ can be solved through the simple ℓ_1 regularization with suitable optimization algorithms, where $k_{\mathcal{S}}$ is an upper bound on the number of *support labels* for any W with $L(X^{(L)}(W), Y) \leq L^*$.

Deep-Trim Algorithms Given a loss function $L(., Y)$ and a pre-trained DNN parameterized by $W^* := \{W^{(j)}\}_{j=1}^{J}$, we initialize with W^* and apply an optimization algorithm that *guarantees descent* to the following layerwise ℓ_1-regularized objective

$$\min_{W^{(j)}} \ \lambda\|vec(W^{(j)})\|_1 + L(X^{(J)}(W), Y) \tag{2.51}$$

for all $j \in [J]$, where $vec(W^{(j)})$ denotes the vectorized version of the tensor $W^{(j)}$.

The following Theorem states that any stationary point of (2.51) has its number of non-zero parameters per layer bounded by the total number of support labels in the training set under a couple of mild conditions.

Theorem 7 (Deep-Trim with ℓ_1 penalty)**.** *Let $\hat{W}^{(j)}$ be any stationary point of objective* (2.51) *with $dim(W^{(j)}) = d$ that lies on a single piece of the piecewise-linear function $X^{(J)}(W) : N \times K$. Let $V : NK \times d$ be the Jacobian matrix of $vec(X^{(J)})$ w.r.t. $vec(W^{(j)})$. For V in* general position*, we have*

$$nnz(\hat{W}^{(j)}) \leq Nk_{\mathcal{S}}(\hat{W}).$$

for any regularization magnitude $\lambda > 0$, where $k_{\mathcal{S}}(\hat{W})$ is the average number of support labels given by the compressed DNN.

Proof. Any stationary point of (2.51) should satisfy the condition

$$V^T vec(A) + \lambda \boldsymbol{\rho} = 0 \tag{2.52}$$

where $A \in \partial L$ is an $N \times K$ subgradient matrix of the loss function w.r.t. the prediction logits, and $\boldsymbol{\rho} \in \partial \|vec(W^{(j)})\|_1$ is a d-dimensional subgradient of the ℓ_1 norm penalty. Then let $\mathcal{Q} := \{r \mid [vec(W^{(j)})]_r \neq 0\}$ be the set of indices of non-zero parameters, we have $[\boldsymbol{\rho}]_r \in \{-1, 1\}$ and thus

$$\left| \left[V^T vec(A) \right]_{\mathcal{Q}} \right| = \lambda \mathbf{1}, \tag{2.53}$$

which cannot be satisfied for V in general position (as defined in the literature of LASSO [95]) unless the number of variables is more than the number of equations in (2.53), that is, $nnz(A) = k_{\mathcal{S}} \geq nnz(W^{(j)}) = |\mathcal{Q}|$. □

Figure 2.1 illustrates an example for the regression task where, no matter how small $\lambda > 0$ is, the second coordinate is always 0 at the stationary point. Note since Theorem 7 holds for any $\lambda > 0$, by choosing $\lambda \leq \epsilon / \|vec(W^{(j)})\|_1$, any descent optimization algorithm can guarantee that

$$\lambda \|vec(\hat{W}^{(j)})\|_1 + L(X^{(J)}(\hat{W}), Y) \leq \lambda \|vec(W^{(j)})\|_1 + L(X^{(J)}(W), Y)$$

and thus

$$L(X^{(J)}(\hat{W}), Y) \leq L(X^{(J)}(W), Y) + \epsilon$$

for any $\epsilon > 0$. Then by applying the procedure for each layer $j \in [J]$, one can solve the *Deep-Trim* problem with number of non-zero parameters $\tau = (Nk_{\mathcal{S}}) \times J$.

In practice, however, the smaller λ, the harder for the optimization algorithm to get close to the stationary point, and it is crucial to choose an optimization algorithm targeting for *high precision* to find a sparse solution of (2.51). In our experiments, we found the combination of Proximal Quasi-Newton (PQN) method with SGD pre-training yields surprisingly effective trimming. We refer readers to [126] for the implementation details of the PQN algorithm.

2.3.3 Experiments

We conduct experiments on the VGG-16 network with benchmark data set CIFAR-10. Since most of the parameters of VGG-16 lie on the last 3 layers of the network, we evaluate our Deep-Trim algorithm by applying it to trim parameters of its last 3 layers. The results are as in Table 2.7.

$nnz(W^{(j)})$	FC-1	FC-2	FC-3	Total	Acc
VGG-16	102760448	16777216	40960	119578624	92.6%
VGG-16 (Trimmed)	3991	4053	373	8417	93.5%
Ratio	0.0038%	0.024%	0.91%	0.007%	

Table 2.7: The number of non-zero parameters of VGG-16, before and after applying the Deep-Trim algorithm, evaluated on the CIFAR-10 task.

We can see the Deep-Trim method based on ℓ_1-regularization gives surprisingly good result—more than 10000x reduction on the number of parameters while improving the accuracy by 1% absolutely. Note although during the experiments we employ the *cross-entropy loss* (3.38) which has a number of support labels $NK = 5 \times 10^5$ on the CIFAR-10 data set, we suspect a more careful analysis could improve our Theorem 7 to give a tighter bound for loss with entries of gradient close to 0 but not exactly 0, making the bound for *cross-entropy loss* (3.38) match that of *maximum-margin loss* (3.39).

Chapter 3

Greedy Optimization via Decomposition

In real-world problems, the output variables could be inter-dependent, which makes the learning and inference challenging when the output domain is large, as the prediction of an instance requires a joint optimization over output variables, which not only prohibits the direct use of our methodology developed in the last chapter, but also introduce computationally expensive burden for the communication between the beliefs of output variables. In this chapter, we propose a decomposed-learning framework that handles message passing not only at the inference level but also at the learning level, following our recent works [44, 119]. This allows the entropy of the predictive distribution over each output variable decreases before communication with other output variables, and making each factor's optimization amenable to an efficient sublinear-time method, such as PD-Sparse discussed in the last chapter, or other approaches via search data structures to be discussed in this chapter. We will introduce our technique under the contexts of learning and inference in section 3.1 and 3.2 respectively. Then in section 3.3, we show how our loss decomposition technique can be perfectly combined with a recent thread of works on Maximum Inner Product Search (MIPS) for learning of large output domains [72, 99].

3.1 Decomposition for Learning Structured Predictor

Structured prediction is prevalent with wide applications in Natural Language Processing (NLP), Computer Vision, and Bioinformatics to name a few, where one is interested in outputs of strong interdependence. Although many dependency structures yield intractable inference problems, approximation techniques such as convex relaxations with theoretical guarantees [56, 64, 80] have been developed. However, solving the relaxed problems (LP, QP, SDP, etc.) is computationally expensive for factor graphs of large output domain and results in prohibitive training time when embedded into a learning algorithm relying on inference oracles [48, 59]. For instance, many applications in NLP such as Machine Translation [29], Speech Recognition [106], and Semantic Parsing [21] have output domains as large as the size of vocabulary, for which the prediction of even a single sentence takes considerable time.

One approach to avoid inference during training is by introducing a loss function conditioned on the given labels of neighboring output variables [82]. However, it also introduces more variance to the estimation of model and could degrade testing performance significantly. Another

thread of research aims to formulate parameter learning and output inference as a joint optimization problem that avoids treating inference as a subroutine [62, 65]. In this appraoch, the structured hinge loss is reformulated via dual decomposition, so both messages between factors and model parameters are treated as first-class variables. The new formulation, however, does not yield computational advantage due to the constraints entangling the two types of variables. In particular, [62] employs a hybrid method (DLPW) that alternatingly optimizes model parameters and messages, but the algorithm is not significantly faster than directly performing stochastic gradient on the structured hinge loss. More recently, [65] proposes an approximate objective for structural SVMs that leads to an algorithm considerably faster than DLPW on problems requiring expensive inference. However, the approximate objective requires a trade-off between efficiency and approximation quality, yielding an $O(1/\epsilon^2)$ overall iteration complexity for achieving ϵ sub-optimality.

The contribution of this new approach is twofold. First, we propose a Greedy Direction Method of Multiplier (GDMM) algorithm that decomposes the training of a structural SVM into factorwise multiclass SVMs connected through sparse messages confined to the active labels. The algorithm guarantees an $O(\log(1/\epsilon))$ iteration complexity for achieving an ϵ sub-optimality and each iteration requires only one pass of *Factorwise Maximization Oracles (FMOs)* over every factor. Second, we show that the FMO can be realized in time sublinear to the cardinality of factor domains, hence is considerably more efficient than a structured maximization oracle when it comes to large output domain. For problems consisting of numerous binary variables, we further give realization of a joint FMO that has complexity sublinear to the number of factors. We conduct experiments on both chain-structured problems that allow exact inference and fully-connected problems that rely on Linear Program relaxations, where we show the proposed approach is orders-of-magnitude faster than current state-of-the-art training algorithms for Structured SVMs.

3.1.1 Problem Setup

Structured prediction aims to predict a set of outputs $\boldsymbol{y} \in \mathcal{Y}(\boldsymbol{x})$ from their interdependency and inputs $\boldsymbol{x} \in \mathcal{X}$. Given a feature map $\boldsymbol{\phi}(\boldsymbol{x}, \boldsymbol{y}) : \mathcal{X} \times \mathcal{Y}(\boldsymbol{x}) \to \mathbb{R}^d$ that extracts relevant information from $(\boldsymbol{x}, \boldsymbol{y})$, a linear classifier with parameters $\boldsymbol{w}$ can be defined as $h(\boldsymbol{x}; \boldsymbol{w}) = arg\max_{\boldsymbol{y} \in \mathcal{Y}(\boldsymbol{x})} \langle \boldsymbol{w}, \boldsymbol{\phi}(\boldsymbol{x}, \boldsymbol{y}) \rangle$, where we estimate the parameters $\boldsymbol{w}$ from a training set $\mathcal{D} = \{(\boldsymbol{x}_i, \bar{\boldsymbol{y}}_i)\}_{i=1}^n$ by solving a regularized *Empirical Risk Minimization (ERM)* problem

$$\min_{\boldsymbol{w}} \quad \frac{1}{2}\|\boldsymbol{w}\|^2 + C \sum_{i=1}^{n} L(\boldsymbol{w}; \boldsymbol{x}_i, \bar{\boldsymbol{y}}_i) \ . \tag{3.1}$$

In case of a Structural SVM [92, 96], we consider the structured hinge loss

$$L(\boldsymbol{w}; \boldsymbol{x}, \bar{\boldsymbol{y}}) = \max_{\boldsymbol{y} \in \mathcal{Y}(\boldsymbol{x})} \langle \boldsymbol{w}, \ \boldsymbol{\phi}(\boldsymbol{x}, \boldsymbol{y}) - \boldsymbol{\phi}(\boldsymbol{x}, \bar{\boldsymbol{y}}) \rangle + \delta(\boldsymbol{y}, \bar{\boldsymbol{y}}), \tag{3.2}$$

where $\delta(\boldsymbol{y}, \bar{\boldsymbol{y}}_i)$ is a task-dependent error function, for which the Hamming distance $\delta_H(\boldsymbol{y}, \bar{\boldsymbol{y}}_i)$ is commonly used. Since the size of domain $|\mathcal{Y}(\boldsymbol{x})|$ typically grows exponentially with the number of output variables, the tractability of problem (C.7) lies in the decomposition of the responses

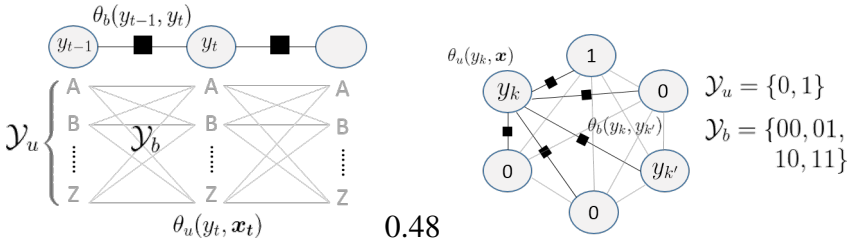

Figure 3.1: (left) Factors with large output domains in Sequence Labeling. (right) Large number of factors in a Correlated Multilabel Prediction problem. Circles denote variables and black boxes denote factors. ($\mathcal{Y}_u$: domain of unigram factor. $\mathcal{Y}_b$: domain of bigram factor.)

$\langle \boldsymbol{w}, \boldsymbol{\phi}(\boldsymbol{x}, \boldsymbol{y}) \rangle$ into several factors, each involving only a few outputs. The factor decomposition can be represented as a bipartite graph $G(\mathcal{F}, \mathcal{V}, \mathcal{E})$ between factors $\mathcal{F}$ and variables $\mathcal{V}$, where an edge $(f, j) \in \mathcal{E}$ exists if the factor f involves the variable j. Typically, a set of factor templates $\mathcal{T}$ exists so that factors of the same template $F \in \mathcal{T}$ share the same feature map $\boldsymbol{\phi}_F(.)$ and parameter vector $\boldsymbol{w}_F$. Then the response on input-output pair $(\boldsymbol{x}, \boldsymbol{y})$ is given by

$$\langle \boldsymbol{w},\ \boldsymbol{\phi}(\boldsymbol{x}, \boldsymbol{y}) \rangle = \sum_{F \in \mathcal{T}} \sum_{f \in F(\boldsymbol{x})} \langle \boldsymbol{w}_F, \boldsymbol{\phi}_F(\boldsymbol{x}_f, \boldsymbol{y}_f) \rangle, \tag{3.3}$$

where $F(\boldsymbol{x})$ denotes the set of factors on $\boldsymbol{x}$ that share a template F, and $\boldsymbol{y}_f$ denotes output variables relevant to factor f of domain $\mathcal{Y}_f = \mathcal{Y}_F$. We will use $\mathcal{F}(\boldsymbol{x})$ to denote the union of factors of different templates $\{F(\boldsymbol{x})\}_{F \in \mathcal{T}}$. Figure 3.1 shows two examples that both have two factor templates (i.e. unigram and bigram) for which the responses have decomposition $\sum_{f \in u(\boldsymbol{x})} \langle \boldsymbol{w}_u, \phi_u(\boldsymbol{x}_f, y_f) \rangle + \sum_{f \in b(\boldsymbol{x})} \langle \boldsymbol{w}_b, \phi_b(y_f) \rangle$. Unfortunately, even with such decomposition, the maximization in (3.2) is still computationally expensive. First, most of graph structures do not allow exact maximization, so in practice one would minimize an upper bound of the original loss (3.2) obtained from relaxation [64, 93]. Second, even for the relaxed loss or a tree-structured graph that allows polynomial-time maximization, its complexity is at least linear to the cardinality of factor domain $|\mathcal{Y}_f|$ times the number of factors $|\mathcal{F}|$. This results in a prohibitive computational cost for problems with large output domain. As in Figure 3.1, one example has a factor domain $|\mathcal{Y}_b|$ which grows quadratically with the size of output domain; the other has the number of factors $|\mathcal{F}|$ which grows quadratically with the number of outputs. A key observation of this paper is, in contrast to the structural maximization (3.2) that requires larger extent of exploration on locally suboptimal assignments in order to achieve global optimality, the *Factorwise Maximization Oracle (FMO)*

$$\boldsymbol{y}_f^* := \underset{\boldsymbol{y}_f}{argmax}\ \langle \boldsymbol{w}_F, \boldsymbol{\phi}(\boldsymbol{x}_f, \boldsymbol{y}_f) \rangle \tag{3.4}$$

can be realized in a more efficient way by maintaining data structures on the factor parameters $\boldsymbol{w}_F$. In the next section, we develop globally-convergent algorithms that rely only on FMO, and provide realizations of *message-augmented FMO* with cost sublinear to the size of factor domain or to the number of factors.

3.1.2 Dual-Decomposed Learning

We consider an upper bound of the loss (3.2) based on a Linear Program (LP) relaxation that is tight in case of a tree-structured graph and leads to a tractable approximation for general factor graphs [62, 93]:

$$L^{LP}(\boldsymbol{w}; \boldsymbol{x}, \bar{\boldsymbol{y}}) = \max_{(\boldsymbol{q},\boldsymbol{p}) \in \mathcal{M}_L} \sum_{f \in \mathcal{F}(\boldsymbol{x})} \left\langle \boldsymbol{\theta}_f(\boldsymbol{w}), \boldsymbol{q}_f \right\rangle \tag{3.5}$$

where $\boldsymbol{\theta}_f(\boldsymbol{w}) := \left(\left\langle \boldsymbol{w}_F, \boldsymbol{\phi}_F(\boldsymbol{x}_f, \boldsymbol{y}_f) - \boldsymbol{\phi}_F(\boldsymbol{x}_f, \bar{\boldsymbol{y}}_f) \right\rangle + \delta_f(\boldsymbol{y}_f, \bar{\boldsymbol{y}}_f) \right)_{\boldsymbol{y}_f \in \mathcal{Y}_f}$. $\mathcal{M}_L$ is a polytope that constrains $\boldsymbol{q}_f$ in a $|\mathcal{Y}_f|$-dimensional simplex $\Delta^{|\mathcal{Y}_f|}$ and also enforces local consistency:

$$\mathcal{M}_L := \left\{ \begin{array}{l} \boldsymbol{q} = (\boldsymbol{q}_f)_{f \in \mathcal{F}(\boldsymbol{x})} \\ \boldsymbol{p} = (\boldsymbol{p}_j)_{j \in \mathcal{V}(\boldsymbol{x})} \end{array} \middle| \begin{array}{ll} \boldsymbol{q}_f \in \Delta^{|\mathcal{Y}_f|}, & \forall f \in F(\boldsymbol{x}), \forall F \in \mathcal{T} \\ M_{jf} \boldsymbol{q}_f = \boldsymbol{p}_j, & \forall (j, f) \in \mathcal{E}(\boldsymbol{x}) \end{array} \right\},$$

where M_{jf} is a $|\mathcal{Y}_j|$ by $|\mathcal{Y}_f|$ matrix that has $M_{jf}(y_j, \boldsymbol{y}_f) = 1$ if y_j is consistent with $\boldsymbol{y}_f$ (i.e. $y_j = [\boldsymbol{y}_f]_j$) and $M_{jf}(y_j, \boldsymbol{y}_f) = 0$ otherwise. For a tree-structured graph $G(\mathcal{F}, \mathcal{V}, \mathcal{E})$, the LP relaxation is tight and thus loss (3.5) is equivalent to (3.2). For a general factor graph, (3.5) is an upper bound on the original loss (3.2). It is observed that parameters $\boldsymbol{w}$ learned from the upper bound (3.5) tend to tightening the LP relaxation and thus in practice lead to tight LP in the testing phase [64]. Instead of solving LP (3.5) as a subroutine, a recent attempt formulates (C.7) as a problem that optimizes $(\boldsymbol{p}, \boldsymbol{q})$ and $\boldsymbol{w}$ jointly via dual decomposition [62, 65]. We denote $\boldsymbol{\lambda}_{jf}$ as dual variables associated with constraint $M_{jf}\boldsymbol{q}_f = \boldsymbol{p}_j$, and $\boldsymbol{\lambda}_f := (\boldsymbol{\lambda}_{jf})_{j \in \mathcal{N}(f)}$ where $\mathcal{N}(f) = \{j \mid (j, f) \in \mathcal{E}\}$. We have

$$\begin{aligned}
L^{LP}(\boldsymbol{w}; \boldsymbol{x}, \bar{\boldsymbol{y}}) &= \max_{\boldsymbol{q},\boldsymbol{p}} \min_{\boldsymbol{\lambda}} \sum_{f \in \mathcal{F}(\boldsymbol{x})} \langle \boldsymbol{\theta}_f(\boldsymbol{w}), \boldsymbol{q}_f \rangle + \sum_{j \in \mathcal{N}(f)} \langle \boldsymbol{\lambda}_{jf}, M_{jf}\boldsymbol{q}_f - \boldsymbol{p}_j \rangle && (3.6) \\
&= \min_{\boldsymbol{\lambda} \in \Lambda} \sum_{f \in \mathcal{F}(\boldsymbol{x})} \max_{\boldsymbol{q}_f \in \Delta^{|\mathcal{Y}_f|}} (\boldsymbol{\theta}_f(\boldsymbol{w}) + \sum_{j \in \mathcal{N}(f)} M_{jf}^T \boldsymbol{\lambda}_{jf})^T \boldsymbol{q}_f && (3.7) \\
&= \min_{\boldsymbol{\lambda} \in \Lambda} \sum_{f \in \mathcal{F}(\boldsymbol{x})} \left(\max_{\boldsymbol{y}_f \in \mathcal{Y}_f} \theta_f(\boldsymbol{y}_f; \boldsymbol{w}) + \sum_{j \in \mathcal{N}(f)} \lambda_{jf}([\boldsymbol{y}_f]_j) \right) && (3.8) \\
&= \min_{\boldsymbol{\lambda} \in \Lambda} \sum_{f \in \mathcal{F}(\boldsymbol{x})} L_f(\boldsymbol{w}; \boldsymbol{x}_f, \bar{\boldsymbol{y}}_f, \boldsymbol{\lambda}_f) && (3.9)
\end{aligned}$$

where (C.16) follows the strong duality, and the domain $\Lambda = \left\{ \boldsymbol{\lambda} \middle| \sum_{(j,f) \in \mathcal{E}(\boldsymbol{x})} \boldsymbol{\lambda}_{jf} = \boldsymbol{0}, \forall j \in \mathcal{V}(\boldsymbol{x}) \right\}$ follows the maximization w.r.t. $\boldsymbol{p}$ in (C.15). The result (3.9) is a loss function $L_f(.)$ that penalizes the response of each factor separately given $\boldsymbol{\lambda}_f$. The ERM problem (C.7) can then be expressed as

$$\min_{\boldsymbol{w}, \boldsymbol{\lambda} \in \Lambda} \sum_{F \in \mathcal{T}} \left(\frac{1}{2} \|\boldsymbol{w}_F\|^2 + C \sum_{f \in F} L_f(\boldsymbol{w}_F; \boldsymbol{x}_f, \bar{\boldsymbol{y}}_f, \boldsymbol{\lambda}_f) \right), \tag{3.10}$$

where $F = \bigcup_{i=1}^N F(\boldsymbol{x}_i)$ and $\mathcal{F} = \bigcup_{F \in \mathcal{T}} F$. The formulation (3.10) has an insightful interpretation: each factor template F learns a multiclass SVM given by parameters $\boldsymbol{w}_F$ from factors $f \in F$, while each factor is augmented with messages $\boldsymbol{\lambda}_f$ passed from all variables related to f.

Algorithm 5 Greedy Direction Method of Multiplier

0. Initialize $t = 0$, $\boldsymbol{\alpha}^0 = \mathbf{0}$, $\boldsymbol{\lambda}^0 = \mathbf{0}$ and $\mathcal{A}^0 = \mathcal{A}^{init}$.
for $t = 0, 1, ...$ **do**
1. Compute $(\boldsymbol{\alpha}^{t+1}, \mathcal{A}^{t+1})$ via one pass of Algorithm 6, 7, or 1.
2. $\boldsymbol{\lambda}_{jf}^{t+1} = \boldsymbol{\lambda}_{jf}^{t} + \eta \left(M_{jf} \boldsymbol{\alpha}_f^{t+1} - \boldsymbol{\alpha}_j^{t+1} \right),\ j \in \mathcal{N}(f),\ \forall f \in \mathcal{F}$.
end for

Greedy Direction Method of Multiplier Let $\alpha_f(\boldsymbol{y}_f)$ be dual variables for the factor responses $z_f(\boldsymbol{y}_f) = \langle \boldsymbol{w}, \boldsymbol{\phi}(\boldsymbol{x}_f, \boldsymbol{y}_f) \rangle$ and $\{\boldsymbol{\alpha}_j\}_{j \in \mathcal{V}}$ be that for constraints in Λ. The dual problem of (3.10) can be expressed as [1]

$$\begin{aligned} \min_{\boldsymbol{\alpha}_f \in \Delta^{|\mathcal{Y}_f|}} \quad & G(\boldsymbol{\alpha}) := \frac{1}{2} \sum_{F \in \mathcal{T}} \| \boldsymbol{w}_F(\boldsymbol{\alpha}) \|^2 - \sum_{j \in \mathcal{V}} \boldsymbol{\delta}_j^T \boldsymbol{\alpha}_j \\ s.t. \quad & M_{jf} \boldsymbol{\alpha}_f = \boldsymbol{\alpha}_j,\ j \in \mathcal{N}(f), f \in \mathcal{F}. \\ & \boldsymbol{w}_F(\boldsymbol{\alpha}) = \sum_{f \in F} \Phi_f^T \boldsymbol{\alpha}_f \end{aligned} \tag{3.11}$$

where $\boldsymbol{\alpha}_f$ lie in the shifted simplex

$$\Delta^{|\mathcal{Y}_f|} := \left\{ \boldsymbol{\alpha}_f \ \middle|\ \alpha_f(\bar{\boldsymbol{y}}_f) \leq C\ ,\ \alpha_f(\boldsymbol{y}_f) \leq 0,\ \forall \boldsymbol{y}_f \neq \bar{\boldsymbol{y}}_f\ ,\ \sum_{\boldsymbol{y}_f \in \mathcal{Y}_f} \alpha_f(\boldsymbol{y}_f) = 0. \right\}. \tag{3.12}$$

Problem (3.11) can be interpreted as a summation of the dual objectives of $|\mathcal{T}|$ multiclass SVMs (each per factor template), connected with consistency constraints. To minimize (3.11) one factor at a time, we adopt a *Greedy Direction Method of Multiplier (GDMM)* algorithm that alternates between minimizing the *Augmented Lagrangian* function

$$\min_{\boldsymbol{\alpha}_f \in \Delta^{|\mathcal{Y}_f|}} \mathcal{L}(\boldsymbol{\alpha}, \boldsymbol{\lambda}^t) := G(\boldsymbol{\alpha}) + \frac{\rho}{2} \sum_{j \in \mathcal{N}(f)\ , f \in \mathcal{F}} \left\| \boldsymbol{m}_{jf}(\boldsymbol{\alpha}, \boldsymbol{\lambda}^t) \right\|^2 - \| \boldsymbol{\lambda}_{jf}^t \|^2 \tag{3.13}$$

and updating the Lagrangian Multipliers (of consistency constraints)

$$\boldsymbol{\lambda}_{jf}^{t+1} = \boldsymbol{\lambda}_{jf}^{t} + \eta \left(M_{jf} \boldsymbol{\alpha}_f - \boldsymbol{\alpha}_j \right). \forall j \in \mathcal{N}(f),\ f \in \mathcal{F}, \tag{3.14}$$

where $\boldsymbol{m}_{jf}(\boldsymbol{\alpha}, \boldsymbol{\lambda}^t) = M_{jf} \boldsymbol{\alpha}_f - \boldsymbol{\alpha}_j + \boldsymbol{\lambda}_{jf}^t$ plays the role of messages between $|\mathcal{T}|$ multiclass problems, and η is a constant step size. The procedure is outlined in Algorithm 5. The minimization (3.13) is conducted in an approximate and greedy fashion, in the aim of involving as few dual variables as possible. We discuss two greedy algorithms that suit two different cases in the following.

[1] $\boldsymbol{\alpha}_j$ is also dual variables for responses on unigram factors. We define $\mathcal{U} := \mathcal{V}$ and $\boldsymbol{\alpha}_f := \boldsymbol{\alpha}_j, \forall f \in \mathcal{U}$.

Factor of Large Domain For problems with large factor domains, we minimize (3.13) via a variant of *Frank-Wolfe* algorithm with *away steps* (AFW) [57], outlined in Algorithm 6. The AFW algorithm maintains the iterate $\boldsymbol{\alpha}^t$ as a linear combination of bases constructed during iterates

$$\boldsymbol{\alpha}^t = \sum_{\boldsymbol{v} \in \mathcal{A}^t} c_{\boldsymbol{v}}^t \boldsymbol{v}, \quad \mathcal{A}^t := \{\boldsymbol{v} \mid c_{\boldsymbol{v}}^t \neq 0\} \tag{3.15}$$

Algorithm 6 Away-step Frank-Wolfe (AFW)

repeat
1. Find $\boldsymbol{v}^+$ satisfying (3.16).
2. Find $\boldsymbol{v}^-$ satisfying (3.17).
3. Compute $\boldsymbol{\alpha}^{t+1}$ by (3.18).
4. Maintain active set $\mathcal{A}^t$ by (3.15).
5. Maintain $\boldsymbol{w}_F(\boldsymbol{\alpha})$ by (3.11).

until a non-drop step is performed.

Algorithm 7 Block-Greedy Coordinate Descent

for $i \in [n]$ **do**
1. Find f^* satisfying (3.19).
2. $\mathcal{A}_i^{s+1} = \mathcal{A}_i^s \cup \{f^*\}$.

for $f \in \mathcal{A}_i$ **do**
3.1 Update $\boldsymbol{\alpha}_f$ by (3.20).
3.2 Maintain $\boldsymbol{w}_F(\boldsymbol{\alpha})$ via (3.11).

end for

end for

where $\mathcal{A}^t$ maintains an active set of bases of non-zero coefficients. Each iteration of AFW finds a direction $\boldsymbol{v}^+ := (\boldsymbol{v}_f^+)_{f \in \mathcal{F}}$ leading to the most descent amount according to the current gradient, subject to the simplex constraints:

$$\boldsymbol{v}_f^+ := \underset{\boldsymbol{v}_f \in \Delta^{|\mathcal{Y}_f|}}{argmin} \langle \nabla_{\boldsymbol{\alpha}_f} \mathcal{L}(\boldsymbol{\alpha}^t, \boldsymbol{\lambda}^t), \boldsymbol{v}_f \rangle = C(\boldsymbol{e}_{\bar{\boldsymbol{y}}_f} - \boldsymbol{e}_{\boldsymbol{y}_f^*}), \ \forall f \in \mathcal{F} \tag{3.16}$$

where $\boldsymbol{y}_f^* := arg\max_{\boldsymbol{y}_f \in \mathcal{Y}_f \setminus \{\bar{\boldsymbol{y}}_f\}} \langle \nabla_{\boldsymbol{\alpha}_f} \mathcal{L}(\boldsymbol{\alpha}^t, \boldsymbol{\lambda}^t), \boldsymbol{e}_{\boldsymbol{y}_f} \rangle$ is the non-ground-truth labeling of factor f of highest response. In addition, AFW finds the *away direction*

$$\boldsymbol{v}^- := \underset{\boldsymbol{v} \in \mathcal{A}^t}{argmax} \langle \nabla_{\boldsymbol{\alpha}} \mathcal{L}(\boldsymbol{\alpha}^t, \boldsymbol{\lambda}^t), \boldsymbol{v} \rangle, \tag{3.17}$$

which corresponds to the basis that leads to the most descent amount when being removed. Then the update is determined by

$$\boldsymbol{\alpha}^{t+1} := \begin{cases} \boldsymbol{\alpha}^t + \gamma_F \boldsymbol{d}_F, & \langle \nabla_{\boldsymbol{\alpha}} \mathcal{L}, \boldsymbol{d}_F \rangle < \langle \nabla_{\boldsymbol{\alpha}} \mathcal{L}, \boldsymbol{d}_A \rangle \\ \boldsymbol{\alpha}^t + \gamma_A \boldsymbol{d}_A, & otherwise. \end{cases} \tag{3.18}$$

where we choose between two descent directions $\boldsymbol{d}_F := \boldsymbol{v}^+ - \boldsymbol{\alpha}^t$ and $\boldsymbol{d}_A := \boldsymbol{\alpha}^t - \boldsymbol{v}^-$. The step size of each direction $\gamma_F := arg\min_{\gamma \in [0,1]} \ \mathcal{L}(\boldsymbol{\alpha}^t + \gamma \boldsymbol{d}_F)$ and $\gamma_A := arg\min_{\gamma \in [0, c_{\boldsymbol{v}^-}]} \ \mathcal{L}(\boldsymbol{\alpha}^t + \gamma \boldsymbol{d}_A)$ can be computed exactly due to the quadratic nature of (3.13). A step is called *drop step* if a step size $\gamma^* = c_{\boldsymbol{v}^-}$ is chosen, which leads to the removal of a basis $\boldsymbol{v}^-$ from the active set, and therefore the total number of drop steps can be bounded by half of the number of iterations t. Since a drop step could lead to insufficient descent, Algorithm 6 stops only if a *non-drop step* is performed. Note Algorithm 6 requires only a factorwise greedy search (3.16) instead of a structural maximization (3.2). Later we will show how the factorwise search can be implemented much more efficiently than structural ones. All the other steps (2-5) in Algorithm 6 can be

computed in $O(|\mathcal{A}_f| nnz(\boldsymbol{\phi}_f))$, where $|\mathcal{A}_f|$ is the number of active states in factor f, which can be much smaller than $|\mathcal{Y}_f|$ when output domain is large.

In practice, a *Block-Coordinate Frank-Wolfe (BCFW)* method has much faster convergence than *Frank-Wolfe* method (Algorithm 6) [59, 75], but proving linear convergence for *BCFW* is also much more difficult [75], which prohibits its use in our analysis. In our implementation, however, we adopt the *BCFW* version since it turns out to be much more efficient. We include a detailed description on the BCFW version in Appendix-A (Algorithm 1).

Large Number of Factors Many structured prediction problems, such as alignment, segmentation, and multilabel prediction (Fig. 3.1, right), comprise binary variables and large number of factors with small domains, for which Algorithm 6 does not yield any computational advantage. For this type of problem, we minimize (3.13) via one pass of *Block-Greedy Coordinate Descent (BGCD)* (Algorithm 7) instead. Let $Q_{\max}$ be an upper bound on the eigenvalue of Hessian matrix of each block $\nabla^2_{\boldsymbol{\alpha}_f} \mathcal{L}(\boldsymbol{\alpha})$. For binary variables of pairwise factor, we have $Q_{\max}$=$4(\max_{f \in F} \|\boldsymbol{\phi}_f\|^2 + 1)$. Each iteration of BGCD finds a factor that leads to the most progress

$$f^* := \underset{f \in \mathcal{F}(\boldsymbol{x}_i)}{argmin} \left(\min_{\boldsymbol{\alpha}_f + \boldsymbol{d} \in \Delta^{|\mathcal{Y}_f|}} \langle \nabla_{\boldsymbol{\alpha}_f} \mathcal{L}(\boldsymbol{\alpha}^t, \boldsymbol{\lambda}^t), \boldsymbol{d} \rangle + \frac{Q_{\max}}{2} \|\boldsymbol{d}\|^2 \right). \tag{3.19}$$

for each instance $\boldsymbol{x}_i$, adds them into the set of active factors $\mathcal{A}_i$, and performs updates by solving block subproblems

$$\boldsymbol{d}^*_f = \underset{\boldsymbol{\alpha}_f + \boldsymbol{d} \in \Delta^{|\mathcal{Y}_f|}}{argmin} \quad \langle \nabla_{\boldsymbol{\alpha}_f} \mathcal{L}(\boldsymbol{\alpha}^t, \boldsymbol{\lambda}^t), \boldsymbol{d} \rangle + \frac{Q_{\max}}{2} \|\boldsymbol{d}\|^2 \tag{3.20}$$

for each factor $f \in \mathcal{A}_i$. Note $|\mathcal{A}_i|$ is bounded by the number of GDMM iterations and it converges to a constant much smaller than $|\mathcal{F}(\boldsymbol{x}_i)|$ in practice. We address in the next section how a joint FMO can be performed to compute (3.19) in time sublinear to $|\mathcal{F}(\boldsymbol{x}_i)|$ in the binary-variable case.

Convergence Analysis The analysis leverages recent analysis on the global linear convergence of Frank-Wolfe variants [57] for function of the form (3.13) with a polyhedral domain, and also the analysis in [42] for Augmented Lagrangian based method. This type of greedy Augmented Lagrangian Method was also analyzed previously under different context [111, 112, 117].

Let $d(\boldsymbol{\lambda}) = \min_{\boldsymbol{\alpha}} \ \mathcal{L}(\boldsymbol{\alpha}, \boldsymbol{\lambda})$ be the dual objective of (3.13), and let

$$\Delta^t_d := d^* - d(\boldsymbol{\lambda}^t), \quad \Delta^t_p := \mathcal{L}(\boldsymbol{\alpha}^t, \boldsymbol{\lambda}^t) - d(\boldsymbol{\lambda}^t) \tag{3.21}$$

be the dual and primal suboptimality of problem (3.11) respectively. We have the following theorems.

Theorem 8 (Convergence of GDMM with AFW)**.** *The iterates* $\{(\boldsymbol{\alpha}^t, \boldsymbol{\lambda}^t)\}_{t=1}^{\infty}$ *produced by Algorithm 5 with step 1 performed by Algorithm 6 has*

$$E[\Delta^t_p + \Delta^t_d] \leq \epsilon \ \textit{for} \ \ t \geq \omega \log(\frac{1}{\epsilon}) \tag{3.22}$$

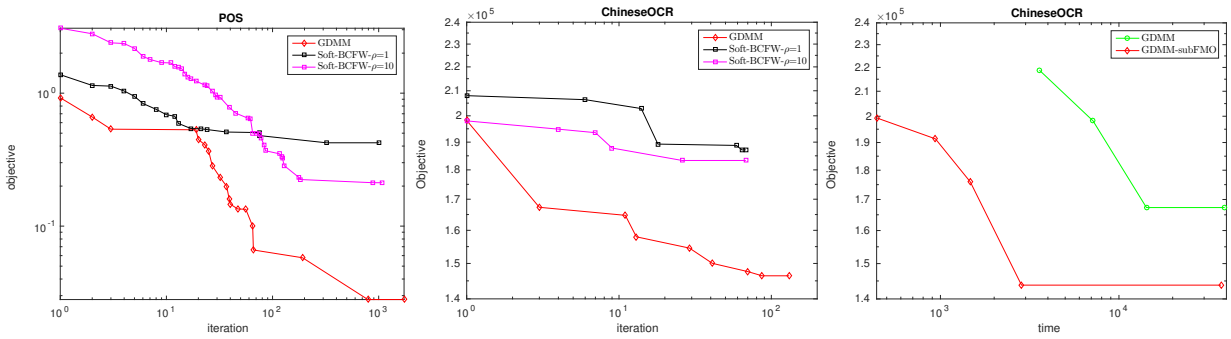

Figure 3.2: (left) Compare two FMO-based algorithms (GDMM, Soft-BCFW) in number of iterations. (right) Improvement in training time given by sublinear-time FMO.

for any $0 < \eta \leq \frac{\rho}{4+16(1+\nu)mQ/\mu_{\mathcal{M}}}$ *with* $\omega = \max\left\{2(1+4\frac{mQ(1+\nu)}{\mu_{\mathcal{M}}}), \frac{\tau}{\eta}\right\}$, *where* $\mu_{\mathcal{M}}$ *is the generalized geometric strong convexity constant of* (3.13)*,* Q *is the Lipschitz-continuous constant for the gradient of objective* (3.13)*, and* $\tau > 0$ *is a constant depending on optimal solution set.*

Theorem 9 (Convergence of GDMM with BGCD)**.** *The iterates* $\{(\boldsymbol{\alpha}^t, \boldsymbol{\lambda}^t)\}_{t=1}^{\infty}$ *produced by Algorithm 5 with step 1 performed by Algorithm 7 has*

$$E[\Delta_p^t + \Delta_d^t] \leq \epsilon \ \textit{for} \ \ t \geq \omega_1 \log(\frac{1}{\epsilon}) \tag{3.23}$$

for any $0 < \eta \leq \frac{\rho}{4(1+Q_{\max}\nu/\mu_1)}$ *with* $\omega_1 = \max\left\{2(1+\frac{Q_{\max}\nu}{\mu_1}), \frac{\tau}{\eta}\right\}$, *where* μ_1 *is the generalized strong convexity constant of objective* (3.13) *and* $Q_{\max} = \max_{f\in\mathcal{F}} Q_f$ *is the factorwise Lipschitz-continuous constant on the gradient.*

3.1.3 Experiments

In this section, we compare with existing approaches on Sequence Labeling and Multi-label prediction with pairwise interaction. The algorithms in comparison are: (i) *BCFW*: a Block-Coordinate Frank-Wolfe method based on structural oracle [59], which outperforms other competitors such as Cutting-Plane, FW, and online-EG methods in [59]. (ii) *SSG*: an implementation of the Stochastic Subgradient method [86]. (iii) *Soft-BCFW*: Algorithm proposed in ([65]), which avoids structural oracle by minimizing an approximate objective, where a parameter ρ controls the precision of the approximation. We tuned the parameter and chose two of the best on the figure. For BCFW and SSG, we adapted the MATLAB implementation provided by authors of [59] into C++, which is an order of magnitude faster. All other implementations are also in C++. The results are compared in terms of primal objective (achieved by $\boldsymbol{w}$) and test accuracy.

Our experiments are conducted on 4 public datasets: *POS*, *ChineseOCR*, *RCV1-regions*, and *EUR-Lex* (directory codes). For sequence labeling we experiment on *POS* and *ChineseOCR*. The *POS* dataset is a subset of Penn treebank[2] that contains 3,808 sentences, 196,223 words, and 45 POS labels. The HIT-MW[3] *ChineseOCR* dataset is a hand-written Chinese character

[2] `https://catalog.ldc.upenn.edu/LDC99T42`
[3] `https://sites.google.com/site/hitmwdb/`

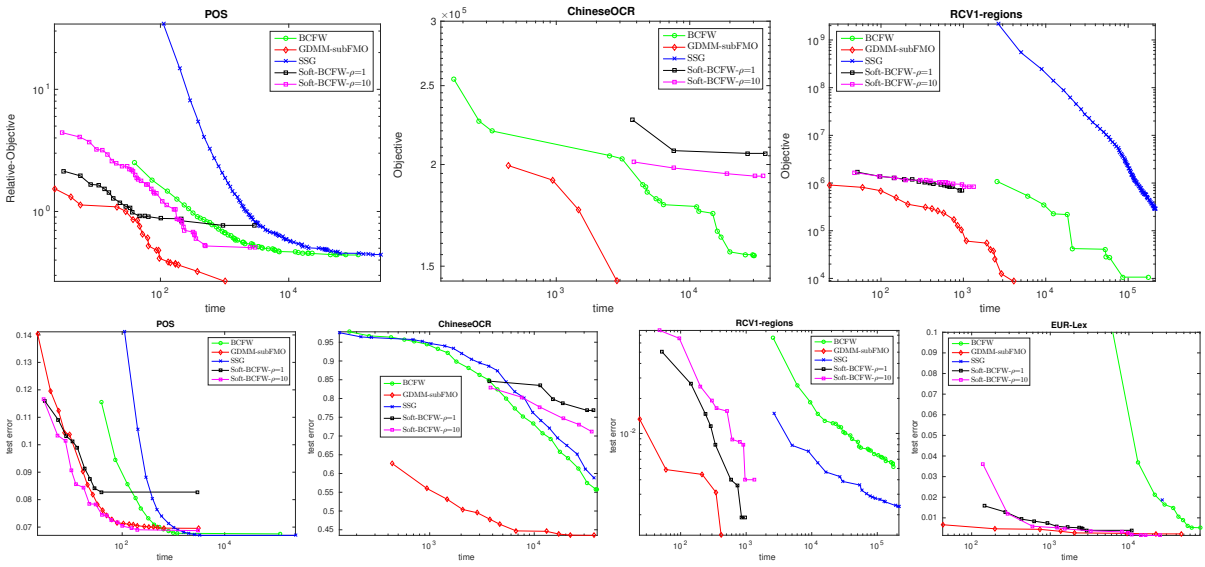

Figure 3.3: Primal Objective v.s. Time and Test error v.s. Time plots. Note that figures of objective have showed that SSG converges to a objective value much higher than all other methods, this is also observed in [59]. Note the training objective for the EUR-Lex data set is too expensive to compute and we are unable to plot the figure.

dataset from [91]. The dataset has 12,064 hand-written sentences, and a total of 174,074 characters. The vocabulary (label) size is 3,039. For the Correlated Multilabel Prediction problems, we experiment on two benchmark datasets *RCV1-regions*[4] and *EUR-Lex* (directory codes)[5]. The *RCV1-regions* dataset has 228 labels, 23,149 training instances and 47,236 features. Note that a smaller version of *RCV1* with only 30 labels and 6000 instances is used in [62, 65]. *EUR-Lex* (directory codes) has 410 directory codes as labels with a sample size of 19,348. We first compare GDMM (without subFMO) with Soft-BCFW in Figure 3.4. Due to the approximation (controlled by ρ), Soft-BCFW can converge to a suboptimal primal objective value. While the gap decreases as ρ increases, its convergence becomes also slower. GDMM, on the other hand, enjoys a faster convergence. The sublinear-time implementation of FMO also reduces the training time by an order of magnitude on the ChineseOCR data set, as showed in Figure 3.4 (right). More general experiments are showed in Figure 3.3. When the size of output domain is small (POS dataset), GDMM-subFMO is competitive to other solvers. As the size of output domain grows (ChineseOCR, RCV1, EUR-Lex), the complexity of structural maximization oracle grows linearly or even quadratically, while the complexity of GDMM-subFMO only grows sublinearly in the experiments. In particular, when running on ChineseOCR and EUR-Lex, each iteration of SSG, GDMM, BCFW and Soft-BCFW take over 10^3 seconds, while it only takes a few seconds in GDMM-subFMO.

[4] `www.csie.ntu.edu.tw/~cjlin/libsvmtools/datasets/multilabel.html`
[5] `mulan.sourceforge.net/datasets-mlc.html`

3.2 Decomposition for MAP Inference

The last section deals with the learning of model parameters of a structured predictor. In this section, we assume the model parameters have been estimated, and the problem is the inference of inter-dependent output variables of potentially huge domain, given the inputs. The problem is often called the *MAP inference*. In particular, we show how our greedy optimization method could be employed to develop an algorithm of sublinear cost w.r.t. the output domain for a MAP linear-program relaxation.

3.2.1 Problem Setup

We study the standard MAP inference problem described by a factor graph $\mathcal{G} = \{\mathcal{V}, \mathcal{F}, \mathcal{E}\}$, where $\mathcal{V}, \mathcal{F}, \mathcal{E}$ are the sets of random variables, factors, and connecting edges, respectively. We denote the neighboring variables for each factor $f \in \mathcal{F}$ as $\partial(f) = \{i|(f,i) \in \mathcal{E}\}$ and similarly the neighboring factors for each variable $i \in \mathcal{V}$ as $\partial(i) = \{f|(f,i) \in \mathcal{E}\}$. Each variable $i \in \mathcal{V}$ takes a value x_i in a finite label set, i.e. $x_i \in \mathcal{X}_i$. Variables and factors are associated with local score functions $\theta_i, \forall i \in \mathcal{V}$ and θ_f, $f \in \mathcal{F}$, respectively.

The goal of the Maximum-a-Posteriori (MAP) problem is to find an assignment $\{x_i \in \mathcal{X}_i | i \in \mathcal{V}\}$ (i.e. each variable takes one value) that maximizes the sum of local score functions:

$$\max_{x} \quad \sum_{i \in \mathcal{V}} \theta_i(x_i) + \sum_{f \in \mathcal{F}} \theta_f(y_f) \tag{3.24}$$

Here $y_f \in \mathcal{Y}_f := \prod_{i \in \partial(f)} \mathcal{X}_i$ collects assignments from variables in $\partial(f)$. Note that the factors associated with only one variable can be absorbed by its neighboring variable, so we assume all factors are of order at least two.

We consider a standard LP relaxation for (3.24). Specifically, we replace x_i and y_f with marginal vectors $\boldsymbol{x}_i \in \Delta_i = \Delta^{|\mathcal{X}_i|}$ and $\boldsymbol{y}_f \in \Delta_f = \Delta^{|\mathcal{Y}_f|}$, where $\Delta^M = \{a \in \mathbb{R}^M | a \succeq \mathbf{0}, \sum_{i=1}^M a_i = 1\}$ is a simplex of dimension M. The marginal vectors are highly constrained with each other. The LP relaxations typically relax these constraints as linear constraints. In this paper, we consider standard linear constraints given by $\forall (f,i) \in \mathcal{E}, M_{if}\boldsymbol{y}_f = \boldsymbol{x}_i$, where

$$M_{if}(x_i, y_f) = \begin{cases} 1 & \text{if } y_f \sim x_i \\ 0 & \text{otherwise} \end{cases}.$$

and $y_f \sim x_i$ means that x_i is consistent with y_f's configuration.

Let $\boldsymbol{\theta}_i$ and $\boldsymbol{\theta}_f$ be vector representations of the local score functions, we derive the LP relaxation in this paper:

$$\begin{aligned} \max_{\substack{\boldsymbol{x}_i \in \Delta_i \\ \boldsymbol{y}_f \in \Delta_f}} \quad & \sum_{i \in \mathcal{V}} \boldsymbol{\theta}_i^T \boldsymbol{x}_i + \sum_{f \in \mathcal{F}} \boldsymbol{\theta}_f^T \boldsymbol{y}_f \\ s.t. \quad & \forall (f,i) \in \mathcal{E}, M_{if}\boldsymbol{y}_f = \boldsymbol{x}_i \end{aligned} \tag{3.25}$$

3.2.2 Greedy Direction Method of Multiplier

Augmented Lagrangian Applying the Augmented Lagrangian method, the objective function becomes (constraints remain the same):

$$\begin{aligned}\mathcal{L}(\{\boldsymbol{x}_i\},\{\boldsymbol{y}_f\}) &= \sum_{i\in\mathcal{V}}(-\boldsymbol{\theta}_i)^T\boldsymbol{x}_i + \sum_{f\in\mathcal{F}}(-\boldsymbol{\theta}_f)^T\boldsymbol{y}_f \\ &+ \sum_{(f,i)\in\mathcal{E}}\Big(\frac{\rho}{2}\|M_{if}\boldsymbol{y}_f - \boldsymbol{x}_i + \frac{1}{\rho}\boldsymbol{\mu}_{if}^t\|_2^2\Big).\end{aligned} \tag{3.26}$$

For optimization, we alternate between optimizing the primal variables $\{\boldsymbol{x}_i, \boldsymbol{y}_f\}$ and the dual variables $\boldsymbol{\mu}_{if}$. Note that the primal constraints $\boldsymbol{x}_i \in \Delta_i$ and $\boldsymbol{y}_f \in \Delta_f$ are strictly enforced during optimization.

GDMM We proceed to describe the proposed greedy direction method of multiplier (or GDMM) for optimizing (3.26) (Please refer to Algorithm 8 for details). GDMM alternates between updating the dual variables and sequentially optimizing the individual primal variables $\boldsymbol{x}_i$ and $\boldsymbol{y}_f$ w.r.t. a judiciously selected active set of states within the factor. However, the cost of such optimization is still expensive when the label space is large. This leads to the key component of this paper, i.e., by maintaining active sets of coordinates for each variable, one can pass only *active messages* to the nearby factors. Once the messages become sparse, the selection of active variables can be implemented efficiently using pre-built data structures. These active sets are initialized as empty sets and are gradually augmented in an on-demand manner. Our approach is motivated by the observations that the non-zero entries in $\boldsymbol{x}_i$ and $\boldsymbol{y}_f$ are relatively sparse and the size of these active sets are small throughout the optimization. These observations could result in significant speedup of the algorithm. Next, we present each step of the GDMM method in detail: optimizing the primal variables with respect to the active sets, updating the active sets, and updating the dual variables.

Primal variable optimization. The primal variables are optimized by the Fully-corrective Frank-Wolfe (FCFW) with approximate correction [58] which alternates between optimizing $\{\boldsymbol{x}_i\}$ and $\{\boldsymbol{y}_f\}$ until a certain precision is achieved. For each $i \in V$, we use $\mathcal{A}_i$ and $\bar{\mathcal{A}}_i$ to denote its active set and the complementary set, respectively. The subproblem of (3.26) w.r.t. $\mathcal{A}_i$ is given by

$$\min_{\substack{\boldsymbol{x}_i\in\Delta_i \\ \boldsymbol{x}_i(\bar{\mathcal{A}}_i)=\boldsymbol{0}}} -\boldsymbol{\theta}_i^T\boldsymbol{x}_i + \frac{\rho}{2}\sum_{f\in\partial(i)}\|M_{if}\boldsymbol{y}_f - \boldsymbol{x}_i + \frac{1}{\rho}\boldsymbol{\mu}_{if}^t\|_2^2$$

which is equivalent to the simplex projection problem

$$\min_{\substack{\boldsymbol{x}_i\in\Delta_i \\ \boldsymbol{x}_i(\bar{\mathcal{A}}_i)=\boldsymbol{0}}} \left\|\boldsymbol{x}_i - \left(\sum_{f\in\partial(i)}\Big(\frac{M_{if}\boldsymbol{y}_f}{|\partial(i)|} + \frac{1}{\rho}\boldsymbol{\mu}_{if}^t\Big) + \frac{\boldsymbol{\theta}_i}{\rho|\partial(i)|}\right)\right\|_2^2 \tag{3.27}$$

As in [26], (3.27) can be solved via a simplex projection operation with time complexity $O(|\mathcal{A}_i|\log(|\mathcal{A}_i|))$. This also implies that, when fixing $\boldsymbol{\mu}$, $\boldsymbol{x}$ can be written as a simple function $\boldsymbol{x}(\boldsymbol{y})$.

Algorithm 8 GDMM for Large Factor Domain

Initialization: $\forall i \in \mathcal{V}, \boldsymbol{x}_i \leftarrow \mathbf{0}, \mathcal{A}_i \leftarrow \emptyset; \forall f \in \mathcal{F}, \boldsymbol{y}_f \leftarrow \mathbf{0}, \mathcal{A}_f \leftarrow \emptyset; \forall (f,i) \in \mathcal{E}, \boldsymbol{\mu}_{if}^0 \leftarrow \mathbf{0}$.
repeat
  for $s = 1, 2, \cdots, S$ **do**
    for $\gamma = 1, 2, 4, 8, \cdots, \gamma_{\max}$ **do**
      1. $\forall f \in \mathcal{F}$, update $\mathcal{A}_f$, $\boldsymbol{y}_f(\mathcal{A}_f)$ according to (3.31) and (3.29) with $Q = \gamma Q_{max}$.
      2. $\forall i \in \mathcal{V}$, update $\mathcal{A}_i$, $\boldsymbol{x}_i(\mathcal{A}_i)$ according to (3.30) and (3.27)
      3. Break if $\mathcal{L}(\boldsymbol{x}', \boldsymbol{y}') - \mathcal{L}(\boldsymbol{x}, \boldsymbol{y}) \leq \sigma \langle \nabla_{\boldsymbol{y}} \mathcal{L}, \Delta \boldsymbol{y} \rangle$
    end for
  end for
  for $\forall (f,i) \in \mathcal{E}$ **do**
    3. Compute $\boldsymbol{\mu}_{if}^{t+1}$ based on (3.32)
  end for
  4. $t \leftarrow t + 1$
until primal/dual infeasibility $< \epsilon$

Now we consider optimizing the variables associated with each factor $f \in \mathcal{F}$. Let $\mathcal{A}_f$ and $\bar{\mathcal{A}}_f$ be its active set and the complementary set. The subproblem w.r.t. $\mathcal{A}_f$ is

$$\begin{aligned} &\min_{\substack{\boldsymbol{y}_f \in \Delta_f \\ \boldsymbol{y}_f(\bar{\mathcal{A}}_f) = \mathbf{0}}} \mathcal{L}_f(\boldsymbol{x}, \boldsymbol{y}; \boldsymbol{\mu}) \\ &:= -\boldsymbol{\theta}_f^T \boldsymbol{y}_f + \frac{\rho}{2} \sum_{i \in \partial(f)} \| M_{if} \boldsymbol{y}_f - \boldsymbol{x}_i + \frac{1}{\rho} \boldsymbol{\mu}_{if}^t \|_2^2 \end{aligned} \tag{3.28}$$

Instead of solving (3.28) exactly, we propose to minimize a quadratic upper bound of the function (3.28). Let $Q = \rho \|M\|^2$ where $\|.\|$ is the spectral norm. (3.28) becomes:

$$\min_{\substack{\boldsymbol{y}_f^+ \in \Delta_f \\ \boldsymbol{y}_f^+(\bar{\mathcal{A}}_f) = \mathbf{0}}} \frac{Q}{2} \| \boldsymbol{y}_f^+ - \boldsymbol{y}_f \|_2^2 + \nabla_{\boldsymbol{y}_f} \mathcal{L}^T (\boldsymbol{y}_f^+ - \boldsymbol{y}_f) \tag{3.29}$$

where $\nabla_{\boldsymbol{y}_f} \mathcal{L} = \sum_{i \in \partial(f)} M_{if}^T (\boldsymbol{\mu}_{if}^t - \rho \boldsymbol{x}_i) - \boldsymbol{\theta}_f$. Note the quadratic upper bound (3.29) is tighter than that used in the Proximal Gradient Descent scheme described in [7] by a factor of $\frac{|\mathcal{Y}_f|}{|\mathcal{A}_f|}$. It can be solved by a simplex projection

$$\min_{\substack{\boldsymbol{y}_f^+ \in \Delta_f \\ \boldsymbol{y}_f^+(\bar{\mathcal{A}}_f) = \mathbf{0}}} \| \boldsymbol{y}_f^+ - (\boldsymbol{y}_f - \frac{1}{Q} (\sum_{i \in \partial(f)} M_{if}^T (\boldsymbol{\mu}_{if}^t - \rho \boldsymbol{x}_i) - \boldsymbol{\theta}_f)) \|_2^2$$

in time O($|\mathcal{A}_f| \log(|\mathcal{A}_f|)$). In practice, computing Q could be expensive, so we introduce a line-search procedure over Q, starting from a lower bound Q_{max} where $Q_{max} := \max_{f \in \mathcal{F}} Q_{\mathcal{A}_f}$ and $Q_{\mathcal{A}_f} = \rho \| [M_{if}]_{\mathcal{A}_f} \|^2$. In practice, we found the descent amount often passes the line-search condition without backtracking. Although we introduce a fully-corrective loop ($s = 1 ... S$) in

Algorithm 8 for ease of our analysis, we found setting $S = 1$ suffices for fast convergence in our experiments. Note by maintaining a small set of non-zero variables, we limit the size of messages (non-zero dual variables), yielding an efficient scheme for finding new active variables as introduced next.

Updating the active sets. The active sets are updated by finding a currently non-active coordinate with largest gradient magnitude. For each $i \in \mathcal{V}$, we update the corresponding active set as

$$\mathcal{A}_i \leftarrow \mathcal{A}_i \cup \arg\max_{x_i \notin \mathcal{A}_i} \nabla_{x_i} \mathcal{L}_i(\boldsymbol{x}, \boldsymbol{y}; \boldsymbol{\mu}) \tag{3.30}$$

where $\nabla \mathcal{L}_{x_i}(\boldsymbol{x}, \boldsymbol{y}; \boldsymbol{\mu}) = -\sum_{f \in \partial(i)} [\rho(M_{if}\boldsymbol{y}_f - \boldsymbol{x}_i) + \boldsymbol{\mu}_{if}^t]_{x_i} - \boldsymbol{\theta}_i(x_i)$. Similarly, for each $f \in \mathcal{F}$, we update the corresponding active set as

$$\mathcal{A}_f \leftarrow \mathcal{A}_f \cup \arg\max_{y_f \notin \mathcal{A}_f} \nabla_{y_f} \mathcal{L}_f(\boldsymbol{x}, \boldsymbol{y}; \boldsymbol{\mu}) \tag{3.31}$$

where $\nabla_{y_f} \mathcal{L}_f(\boldsymbol{x}, \boldsymbol{y}; \boldsymbol{\mu}) = \sum_{i \in \partial(f)} [M_{if}^T(\rho(M_{if}\boldsymbol{y}_f - \boldsymbol{x}_i) + \boldsymbol{\mu}_{if}^t)]_{y_f} - \boldsymbol{\theta}_f(y_f)$. Note that the new active coordinates can be searched efficiently as long as the messages passed from nearby factors $(M_{if}\boldsymbol{y}_f - \boldsymbol{x}_i)$ are sparse. Specifically, as $\boldsymbol{\theta}_i$ and $\boldsymbol{\theta}_f$ are constant, we can build *sorted lists* on $\boldsymbol{\theta}_i$, $\boldsymbol{\theta}_f$ in the preprocessing phase. When executing our algorithm, the selection (3.30) can be done in time $O(|\mathcal{A}_i| + \sum_{f \in \partial(i)} |\mathcal{A}_f|)$ since the size of non-zero messages is bounded by $\sum_{f \in \partial(i)} |\mathcal{A}_f|$.

The time complexity of the coordinate selection of factor based on (3.31) is O($|\mathcal{A}_i||\mathcal{A}_{i'}| + |\mathcal{A}_f|$), where the messages are denoted by $\boldsymbol{\delta}_{if} = \rho(M_{if}\boldsymbol{y}_f - \boldsymbol{x}_i) + \boldsymbol{\mu}_{if}^t$. For a pairwise-interaction factor $f = (i, i') \in \mathcal{F}$ with $y_f = (x_i, x_{i'})$ and $i, i' \in \mathcal{V}$, we can find a coordinate of largest gradient magnitude (3.31) by searching in four divisions of the coordinates: (i) $\{(x_i, x_{i'})|, \boldsymbol{\delta}_{if}(x_i) = \boldsymbol{\delta}_{i'f}(x_{i'}) = 0\}$, (ii) $\{(x_i, x_{i'})|\boldsymbol{\delta}_{if}(x_i) = 0\}$, (iii) $\{(x_i, x_{i'})|\boldsymbol{\delta}_{i'f}(x_{i'}) = 0\}$ and (iv) $\{(x_i, x_{i'})|\boldsymbol{\delta}_{if}(x_i) \neq 0, \boldsymbol{\delta}_{i'f}(x_{i'}) \neq 0\}$. In particular, (i) can be done in O(1) via a sorted list on $\boldsymbol{\theta}_f$. (ii), (iii) can be done in time O($|\mathcal{A}_i||\mathcal{A}_{i'}| + |\mathcal{A}_f|$) via sorted lists built on the sets $\{\theta_f(x_i, x_{i'})|x_i = k\}$, $\{\theta_f(x_i, x_{i'})|x_{i'} = k\}$ respectively. The complexity of (iv) is $O(|\mathcal{A}_i||\mathcal{A}_{i'}|)$.

To maintain a compact active set, after solving (3.27) and (3.29), we remove all coordinates x_i, y_f with $\boldsymbol{x}_i(x_i) = 0$, $\boldsymbol{y}_f(y_f) = 0$ from $\mathcal{A}_i$, $\mathcal{A}_f$, respectively.

Updating the dual After optimizing the primal variables, the dual variables are updated as

$$\boldsymbol{\mu}_{if}^{t+1} = \boldsymbol{\mu}_{if}^t + \eta(M_{if}\boldsymbol{y}_f - \boldsymbol{x}_i), \qquad \forall i \in \mathcal{V},\ f \in \mathcal{F}, \tag{3.32}$$

where η is the dual step size. In the analysis section we show for a sufficiently small η, Algorithm 8 globally converges to the optimum.

GDMM for a Large Number of Factors For applications such as alignment, segmentation and multilabel prediction, the factor graphs comprise a large number of factors with binary random variables. In this section, we introduce a variant of the GDMM specifically for such problems. The procedure is sketched in Algorithm 9.

The idea behind Algorithm 9 is to maintain two active sets for variables and factors, denoted as $\mathcal{A}_\mathcal{V} \subseteq \mathcal{V}$ and $\mathcal{A}_\mathcal{F} \subseteq \mathcal{F}$, respectively. Then we ensure that at each iteration a variable i^*

Algorithm 9 GDMM for Large Number of Factors

Initialization: $\forall i \in \mathcal{V}, \boldsymbol{x}_i \leftarrow \mathbf{0}$; $\forall f \in \mathcal{F}, \boldsymbol{y}_f \leftarrow \mathbf{0}$; $\forall (f,i) \in \mathcal{E}, \boldsymbol{\mu}_{if}^0 \leftarrow \mathbf{0}$; $t \leftarrow 0$. $\mathcal{A}_\mathcal{V} \leftarrow \emptyset$; $\mathcal{A}_\mathcal{F} \leftarrow \emptyset$.

repeat

1. Find i^* satisfying (3.33), $\mathcal{A}_\mathcal{V} \leftarrow \mathcal{A}_\mathcal{V} \cup \{i^*\}$.
2. $\mathcal{A}_\mathcal{F} \leftarrow \mathcal{A}_\mathcal{F} \cup \{f \in \mathcal{F} | \partial(f) \subseteq \mathcal{A}_\mathcal{V}\}$
3. $\forall i \in \mathcal{A}_\mathcal{V}$, update $\boldsymbol{x}_i(\mathcal{X}_i)$ according to (3.27).
4. Find f^* satisfying (3.34), $\mathcal{A}_\mathcal{F} \leftarrow \mathcal{A}_\mathcal{F} \cup \{f^*\}$.
5. $\forall f \in \mathcal{A}_\mathcal{F}$, update $\boldsymbol{y}_f(\mathcal{Y}_f)$ according to (3.29).

for $\forall f \in \mathcal{A}_\mathcal{F}, i \in \partial(f)$ **do**

6. Update $\boldsymbol{\mu}_{if}^{t+1}$ base on (3.32)

end for

7. $t \leftarrow t+1$

until primal/dual infeasibility $< \epsilon$

satisfying

$$i^* = \underset{i}{argmin} \min_{\boldsymbol{x}_i + \boldsymbol{d} \in \Delta_i} \langle \nabla_{\boldsymbol{x}_i} \mathcal{L}_i(\boldsymbol{x}, \boldsymbol{y}; \boldsymbol{\mu}), \boldsymbol{d} \rangle + \frac{1}{2}\|\boldsymbol{d}\|^2 \tag{3.33}$$

and a factor f^* satisfying

$$f^* = \underset{f}{argmin} \min_{\boldsymbol{y}_f + \boldsymbol{d} \in \Delta_f} \langle \nabla_{\boldsymbol{y}_f} \mathcal{L}_f(\boldsymbol{x}, \boldsymbol{y}; \boldsymbol{\mu}), \boldsymbol{d} \rangle + \frac{Q_f}{2}\|\boldsymbol{d}\|^2. \tag{3.34}$$

are in the active sets (i.e. $i^* \in \mathcal{A}_\mathcal{V}$ and $f^* \in \mathcal{A}_\mathcal{F}$).

Note that all variables are binary and all factors have only 4 states, i.e. $\mathcal{Y}_f = \{(0,0),(0,1),(1,0),(1,1)\}$. Therefore one can always write $x_i(0) = 1 - x_i(1)$ and write $y_f(0,0), y_f(0,1), y_f(1,0)$ as a function of $y_f(1,1), x_j(1)$ and $x_i(1)$ by enforcing consistency constraints $y_f(1,0) + y_f(1,1) = x_i(1)$ and $y_f(0,1) + y_f(1,1) = x_j(1)$ for a factor connecting to two variables i and j. Then one can verify the remaining constraints between factors and variables are:

$$\begin{aligned} x_i(1) - y_f(1,1) = y_f(1,0) \geq 0 \\ x_j(1) - y_f(1,1) = y_f(0,1) \geq 0 \\ 1 - x_j(1) - x_i(1) + y_f(1,1) = y_f(0,0) \geq 0 \end{aligned} \tag{3.35}$$

One can encode (3.35) as the new set of constraints $M_{if}\boldsymbol{y}_f - \boldsymbol{x}_i = 0$ by introducing slack variables for the inequalities. Under this scheme, we show an efficient method to include all variables and factors that satisfy (3.33) and (3.34) in the active sets.

We achieve this by first pushing any inactive f into $\mathcal{A}_\mathcal{F}$ if its neighboring variables i and j are both active. Note this accounts for only a small fraction of the factors if $|\mathcal{A}_\mathcal{V}| \ll |\mathcal{V}|$, and thus can be computed efficiently. Then we consider any inactive factor f with either i or j inactive. For this type of factor, we have (3.35) satisfied and $\boldsymbol{\mu}_{if} = \boldsymbol{\mu}_{jf} = \mathbf{0}$, and the gradient of each factor $y_f := y_f(1,1)$ is simply

$$\nabla_{y_f} \mathcal{L}_f(\boldsymbol{x}, \boldsymbol{y}; \boldsymbol{\mu}) = -\theta_f(1,1).$$

Therefore, the search of active factor (3.34) reduces to finding the minimal $\theta_f(1,1)$ among $f \notin \mathcal{A}_\mathcal{F}$, which can be easily done via a sorted list of $-\boldsymbol{\theta}_f(1,1)$. For each iteration, we add the first inactive f from the head until we have $-\boldsymbol{\theta}_f(1,1) > 0$ which can be done with $O(1)$ amortized cost. In our experiments, only a small fraction of $\mathcal{F}$ will be added to $\mathcal{A}_\mathcal{F}$. One can find in the appendix the statistics of active size $|\mathcal{A}_\mathcal{F}|$ in our experiments.

Convergence Analysis In this section, we show the GDMM algorithms for the large factor domains (Algorithm 8) and a large number of factors (Algorithm 9) admit linear convergence. To state the main result formally, we introduce the following notations. We use $\boldsymbol{z}^t = (\boldsymbol{x}^t, \boldsymbol{y}^t)$ to encapsulate the primal variables, and denote the feasible space as $\mathcal{M} := \{(\boldsymbol{x}, \boldsymbol{y}) \in \ | \ \boldsymbol{x}_i \in \Delta_i, \forall i \in \mathcal{V}; \boldsymbol{y}_f \in \Delta_f, \forall f \in \mathcal{F}\}$. Then the Augmented Lagrangian is expressed as

$$\mathcal{L}(\boldsymbol{z}, \boldsymbol{\mu}) = \langle -\boldsymbol{\theta} + M^T \boldsymbol{\mu}, \boldsymbol{z} \rangle + \frac{\rho}{2} \|M\boldsymbol{z}\|^2,$$

where $M\boldsymbol{z} = \mathbf{0}$ collects all the constraints of the form

$$M_{if}\boldsymbol{y}_f - \boldsymbol{x}_i = \mathbf{0}, \quad \forall (i,f) \in \mathcal{E}.$$

To assess the convergence rate, denote a primal minimizer respect to a dual iterate $\boldsymbol{\mu}^t$ at iteration t as

$$\bar{\boldsymbol{z}}^t := arg\min_{\boldsymbol{z} \in \mathcal{M}} \ \mathcal{L}(\boldsymbol{z}, \boldsymbol{\mu}^t). \tag{3.36}$$

The convergence rate is measured by the primal gap Δ_p^t and the dual gap Δ_d^t:

$$\Delta_p^t := \mathcal{L}(\boldsymbol{z}^{t+1}, \boldsymbol{\mu}^t) - \mathcal{L}(\bar{\boldsymbol{z}}^t, \boldsymbol{\mu}^t)$$
$$\Delta_d^t := d^* - \mathcal{L}(\bar{\boldsymbol{z}}^t, \boldsymbol{\mu}^t)$$

where $d^* := \max_{\boldsymbol{\mu}} \min_{\boldsymbol{z} \in \mathcal{M}}; \mathcal{L}(\boldsymbol{z}, \boldsymbol{\mu})$ is the optimal dual objective value. The following part states two major results for the convergence of the GDMM in the form of both Algorithm 8 and Algorithm 9.

Theorem 10 (Convergence of Algorithm 8)**.** *Let $Q = \rho\|M\|^2$. For any constant dual step size η satisfying*

$$0 < \eta \leq \frac{\rho}{4(1 + |\mathcal{F}|Q/m_\mathcal{M})},$$

the iterates given by Algorithm 8 obey

$$\Delta_p^t + \Delta_d^t \leq \epsilon, \ \forall t \geq \max\left\{2(1 + \frac{|\mathcal{F}|Q}{m_\mathcal{M}}), \frac{\tau}{\eta}\right\} \log(\frac{1}{\epsilon}).$$

Here $m_\mathcal{M}$ is the generalized geometric strong convexity constant of function $\mathcal{L}(., \boldsymbol{\mu})$ on the domain $\mathcal{M}$ (defined in Lemma 12), and τ is a constant characterized in Lemma 11.

Theorem 11 (Convergence of Algorithm 9)**.** *Let $Q_{\max} := \max_{f \in \mathcal{F}} Q_f$. For any constant dual step size η satisfying*

$$0 < \eta \leq \frac{\rho}{4(1 + Q_{max}/m_1)},$$

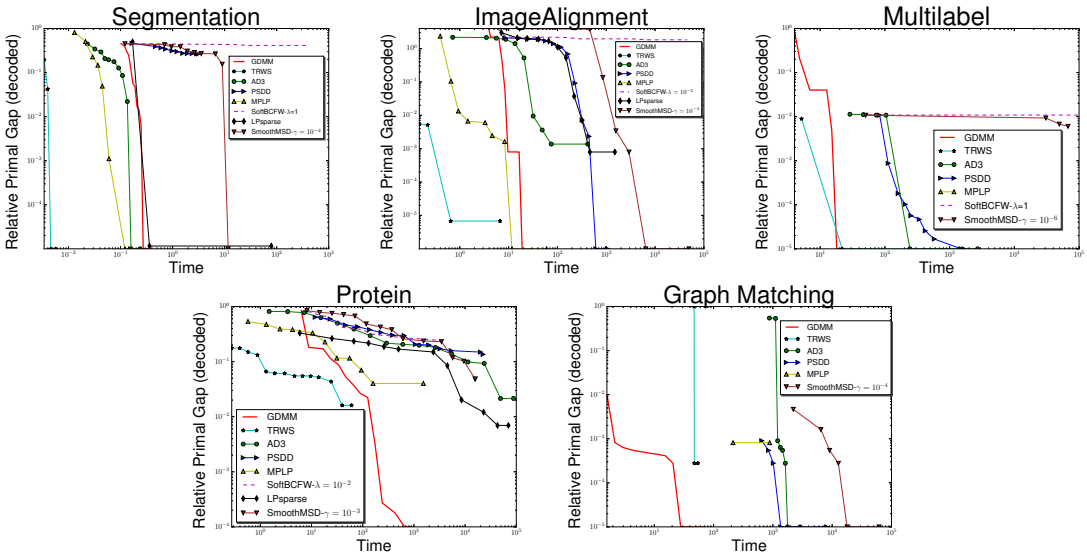

Figure 3.4: Convergence results for data sets with (a)(b): small #state and small #factor; (c): large #factor; (d),(e): large #states. In this paper we only consider decoded (integer solution) primal objective. Relative Primal Gap (decoded) is defined as $\frac{P^*-P}{|P^*|}$ where P is the current primal objective and P^* is the best primal objective found among all algorithms.

the iterates given by Algorithm 9 have

$$\Delta_p^t + \Delta_d^t \leq \epsilon, \ \forall t \geq \max\left\{2(1+\frac{Q_{max}}{m_1}), \frac{\tau}{\eta}\right\}\log(\frac{1}{\epsilon})$$

where m_1 is a generalized strong convexity constant of function $\mathcal{L}(., \boldsymbol{\mu})$ measured in ℓ_1-norm (defined in Lemma B.36).

Theorem 10 and 11 are based on the analysis framework of [42] for the Augmented Lagrangian Method, as well as the recent results on the linear convergence of Frank-Wolfe variants [57] and Greedy Coordinate Descent for non-strongly convex functions of the form (B.2) [74, 114].

3.2.3 Experiments

Here we evaluate the GDMM algorithm by analyzing the performance of GDMM and compare it with state-of-the-art MAP inference techniques.

Experimental Setup **Benchmark datasets.** We select five datasets that involve large output domains. These datasets are selected from two benchmarks: OPENGM[7] and *The Probabilistic*

[6]The specific instances we used for each dataset are: 16_16_s.21.uai (Segmentation), fileforGal_400markers.uai (ImageAlignment), EurLex (Multilabel), and 2BBN.uai (Protein)

[7]`http://hciweb2.iwr.uni-heidelberg.de/opengm/`

Dataset	$\lvert\mathcal{X}_i\rvert$	$\lvert\mathcal{Y}_f\rvert$	$\lvert\mathcal{V}\rvert$	$\lvert\mathcal{F}\rvert$
Segmentation	21	441	226	842
ImageAlignment	83	6889	400	3334
MultiLabel	2	4	3884	7544670
Protein	404	163216	37	703
GraphMatching	1034	1069156	188	1864

Table 3.1: Data Statistics [6]. $|\mathcal{X}_i|$ denotes the maximum domain size of an output variable, $|\mathcal{Y}_f|$ is the maximum domain size of a factor, and $|\mathcal{V}|$, $|\mathcal{F}|$ denote the number of nodes, factors respectively.

Dataset	Segmentation		ImageAlignment		Multilabel		Protein		Graph Matching	
Algorithm	Time	Primal	Time	Primal	Time	Primal	Time	Primal	Time	Primal
LBP	0.406s	**-252.39**	4.136s	-6907	5012.6s	**899.06**	86407s	11904	17696s	**3733**
TRWBP	1.6336s	**-252.39**	17.2s	-6907	25474s	**899.06**	75205s	11888	1598s	**3733**
TreeEP	11.514s	**-252.39**	2666s	-6916.95	2863.2s	**899.06**	2010.4s	11974	TLE	TLE
HAK	6.75s	**-252.39**	274.5s	-6908	TLE	TLE	10013s	10740	87886s	3717
GBP	0.22s	**-252.39**	6.07s	-6908	TLE	TLE	NE	NE	3432s	3721
SoftBCFW	20.89s	-356.57	5977s	-19572	54.22s	889.32	1838s	9250	MLE	MLE
LPsparse	0.57s	**-252.39**	370s	-6913.1	N/A	N/A	32000s	12179	MLE	MLE
SmoothMSD	11.19s	**-252.39**	6462s	**-6907.6**	53462s	893.66	13648s	11670	15312s	**3733**
MPLP	0.12s	**-252.39**	**9.85s**	**-6907.6**	45514s	846.64	124.3s	11771	212.5s	3730
PSDD	2.596s	-319.38	531.4s	**-6907.6**	1503s	**899.06**	20865s	10601.6	1192.9s	**3733**
AD^3	0.15s	**-252.39**	60.2s	-6917.09	243.03s	**899.06**	48137s	12000.8	1659s	**3733**
TRW-S	**0.0046s**	**-252.39**	0.65s	-6907	21.89s	**899.06**	39.67s	12067	47.3s	3732
GDMM	0.26s	**-252.39**	18.09s	**-6907.6**	**16.02s**	**899.06**	**661s**	**12263**	**25.9s**	**3733**

Table 3.2: Primal : best decoded (integer solution) primal objective; Time: time taken to reach the decoded primal objective. The shortest time taken to reach the best MAP objective among all methods are marked with boldface. For all experiments we set memory limit = 100G and time limit = 2 day. Experiments violating these limits are marked MLE: Memory Limit Exceeded, or TLE: Time Limit Exceeded; NE means the solver throws error message "quantity not normalizable".

Inference Challenge 2011(PASCAL)[8]. As shown in Table 3.1, they cover a diverse set of examples with varying scales, network connectivities, and output domain sizes. To help understand the performance of the algorithms, we briefly describe each dataset:

- **Protein.** [27, 109] A Protein Folding dataset that is included in both PASCAL and OpenGM. Its factor graph has bigram factors between every pair of variables.
- **Graph Matching.** A real world social network matching dataset generated using Facebook network data from SNAP[9]. We attempt to match subgraphs to the original graph. In particular, we construct the factor graph as follows: 1. one variable for each node in a subgraph, with vertex set of the original graph as its domain. We use inner product of the features provided from SNAP to generate the unigram factors. 2. For each induced edge on the subgraph, we introduce a bigram factor for the two variables and use inverse of the shortest distance on the graph to generate the bigram factor.

[8] `http://www.cs.huji.ac.il/project/PASCAL/showNet.php`
[9] `http://snap.stanford.edu`

- **Segmentation.** A dataset from PASCAL which has a grid-4 neighborhood structure.
- **ImageAlignment.** A dataset from PASCAL.
- **Multilabel.** A multilabel data set from Mulan [10]. The MAP inference problem is generated from a trained model and an instance from the testset.

Compared Algorithms. The compared algorithms include several top performing algorithms as well as widely used MAP inference algorithms.

- TRWS [53]: a variant of the *Tree-Reweighted max-product message passing* (TRW) [101] algorithm that a) decomposes the graph into *monotonic chains* instead of trees, and b) updates messages according to a global order on each chain. We used implementation from the OPENGM Library [3, 4], which wraps the original TRWS code.
- AD^3: The *Alternating Directions Dual Decomposition* algorithm with public implementation [11] provided by the authors of [61].
- PSDD: The *Projected Subgradient Dual Decomposition* algorithm by [54]. Its implementation is contained in the AD^3 code.
- MPLP: The *Max-Product Linear Programming* algorithm [30, 89, 90] with public implementation [12]. For a fair comparison, we disabled its tightening process to make sure that GDMM and MPLP optimize the same objective function.
- According to the comparison figures in [63], we add two leading methods in terms of performance.
  - SmoothMSD: as a variant of the coordinate minimization method on smoothed dual (the *CD soft* in [63]), we implemented the smooth Max-Sum Diffusion (MSD) algorithm in [104].
  - SoftBCFW: the *Block-Coordinate Frank-Wolfe* for soft-constrained primal, described as Algorithm 1 by [63].

  For each experiment, we choose $\gamma \in \{10^{-2}, 10^{-3}, 10^{-4}, 10^{-5}, 10^{-6}\}$ for SmoothMSD and $\lambda \in \{1, 10^{-1}, 10^{-2}, 10^{-3}, 10^{-4}\}$ for SoftBCFW. Among parameters that give the best decoded primal objective, we select the fastest.
- LPsparse: A recently proposed general-purpose LP solver for problems with sparse structures [117].
- Algorithm implemented in libDAI [70], including the *Loopy Belief Propagation* (LBP) [55], the *Tree-Reweighted Belief Propagation* (TRWBP) [100], the *Tree Expectation Propagation* (TreeEP)[78], the *Double-loop GBP* (HAK) [40], and *Generalized Belief Propagation* (GBP) [110].

In all experiments, we pick $\rho \in \{1, 0.1, 0.01, 0.001\}$ and $\eta \in \{0.1, 1\}$ for GDMM. Generally speaking, the performance of MAP inference algorithms is largely determined by the structure and domain size of the factor graph. Among these five datasets, Segmentation and Image Alignment represent factor graphs with small domain sizes and a small number of factors; Multilabel

[10] `http://mulan.sourceforge.net/datasets-mlc.html`
[11] `http://www.cs.cmu.edu/~ark/AD3/`
[12] `http://cs.nyu.edu/~dsontag/code/README_v2.html`

represents graphs with a large number of factors; Protein and Graph Matching represent graphs with a large number of factors and large domain sizes.

Benchmark Evaluation In this section, we present our benchmark evaluation results. All participating methods are compared in terms of running time and decoded (integer solution) primal objective. Table 3.2 presents the statistics, and Figure 3.4 illustrates the convergence behavior.

Overall, GDMM is the top-performing algorithm. It returns the best solutions among all five datasets. In terms of running time, it is also competent with the top algorithms on each dataset.

On the grid-4 structured factor graphs such as Segmentation, TRWS and MPLP are generally fast and accurate. But they are also less accurate on several examples. AD^3 could provide more accurate solutions, but its active-set method introduces a variable selection procedure that has cost linear to the factor domain size, which results in the slow convergence on Protein and Graph Matching. In contrast, GDMM takes advantage of the active sets of size sublinear to the factor domains (see Appendix C for details), which leads to orders of magnitude speedup. Algorithms such as SoftBCFW and SmoothMSD minimize over a smoothed objective function. However SoftBCFW can not explicityly enforce consistency constraints and hence often stuck at solutions with poor quality. Meanwhile SmoothMSD can provide solutions with good quality at the cost of slow convergence.

Another key feature of the GDMM is its small memory overhead. Most algorithms require memory that is linear in the sum of factor domain sizes, due to the dense messages and primal variables. For example, for the Graph Matching dataset, the memory consumptions for the compared algorithms are AD^3 / PSDD: 69G, MPLP: 45G, TRWS:13G, LPsparse / SoftBCFW:$>$100G. GDMM addresses this issue by maintaining an active set and sparse messages, results in a memory footprint of only 260M.

3.3 Decomposition with Maximum Inner-Product Search (MIPS)

For problems with large output spaces, evaluation of the loss function and its gradient are expensive, typically taking linear time in the size of the output space. In chapter 2, we have discussed one thread of our approach to deal with large output domain through a variety types of sparse structures. However, many models of interest in practice could produce *dense* embedding of dimension ranging from hundreds to thousands, on which the efficiency gained by a primal-sparse algorithm could be less than the case of *sparse* feature maps of dimension up to millions.

In such case of *dense medium-dimensional embedding*, another thread of methods have been developed to speed up learning via efficient data structures for *Nearest-Neighbor Search (NNS)* or *Maximum Inner-Product Search (MIPS)*. However, the performance of such data structures typically degrades when dimension grows. In this section, we propose a novel technique to reduce the intractable high dimensional NNS or MIPS search problem to several much more tractable lower dimensional ones via *dual decomposition* of the loss function. At the same time, we demonstrate guaranteed convergence to the original loss via a greedy message passing procedure. In our experiments on multiclass and multilabel classification with hundreds of thousands

of classes, as well as training skip-gram word embeddings with a vocabulary size of half a million, our technique consistently improves the accuracy of search-based gradient approximation methods and outperforms sampling-based gradient approximation methods by a large margin.

3.3.1 Problem Setup

Let $\mathcal{X}$ denote the input space and $\mathcal{Y}$ the output space, and let $K := |\mathcal{Y}|$. In this paper we focus on the situation where K is extremely large, on the order of hundreds of thousands or larger. We are interested in learning a scoring function $f : \mathcal{X} \to \mathbb{R}^K$ for a large output space $\mathcal{Y}$ from a given class of such functions, $\mathcal{F}$. Labeled samples are pairs $(\boldsymbol{x}, \mathcal{P})$ with $\boldsymbol{x} \in \mathcal{X}$ and $\mathcal{P} \subseteq \mathcal{Y}$ which denotes the set of correct labels for the input point $\boldsymbol{x}$. We use the notation $\mathcal{N} := \mathcal{Y} \setminus \mathcal{P}$ to denote the set of negative labels for the example. Given a collection of training samples $\{(\boldsymbol{x}_i, \mathcal{P}_i)\}_{i=1}^N$, the learning objective takes the following form:

$$\min_{f \in \mathcal{F}} \frac{1}{N} \sum_{i=1}^{N} L(f(\boldsymbol{x}_i), \mathcal{P}_i).$$

where $L : \mathbb{R}^K \times 2^{\mathcal{Y}} \to \mathbb{R}$ is a loss function such that $L(\boldsymbol{z}, \mathcal{P})$ penalizes the discrepancy between the score vector $\boldsymbol{z} \in \mathbb{R}^K$ and a set of positive labels $\mathcal{P} \subseteq \mathcal{Y}$. The evaluation of the loss function and its gradient with respect to the score vector, $\nabla_{\boldsymbol{z}} L(\boldsymbol{z}, \mathcal{P})$, typically has cost growing linearly with the size of the output space K, and thus is expensive for problems with huge output spaces.

The key to our method for reducing the complexity of loss and gradient evaluation will be the following linear structural assumption on the class of scoring functions $\mathcal{F}$: there is an *embedding dimension* parameter $D \in \mathbb{N}$ such that for every $f \in \mathcal{F}$, we can associate a weight matrix $\boldsymbol{W} \in \mathbb{R}^{K \times D}$ and feature map $\boldsymbol{\phi} : \mathcal{X} \to \mathbb{R}^D$ so that for all $\boldsymbol{x} \in \mathcal{X}$,

$$f(\boldsymbol{x}) = \boldsymbol{W}\boldsymbol{\phi}(\boldsymbol{x}). \tag{3.37}$$

We will assume that $D \ll K$, say on the order of a few hundreds or thousands, so that we can explicitly evaluate $\boldsymbol{\phi}(\boldsymbol{x})$.

The problem we consider is the following: given f and a batch of samples $\{\boldsymbol{x}_i, \mathcal{P}_i\}_{i=1}^N$, compute an approximation to the empirical loss $\frac{1}{N}\sum_{i=1}^N L(f(\boldsymbol{x}_i), \mathcal{P}_i)$ and its gradient. This is an important subroutine that naturally arises in either full batch gradient descent or minibatch stochastic gradient descent.

The main challenge here is to construct data structures that preprocess the matrix $\boldsymbol{W}$ so that good approximations to the loss $f(\boldsymbol{x}_i, \mathcal{P}_i)$ and its gradient can be computed without computing the vector $f(\boldsymbol{x})$ entirely: i.e. we desire sublinear (in K) time computation of such approximations given access to an appropriate data structure.

Before proceeding to our dual decomposition based search technique, we give a few examples of problems with large output space that fit in our framework:

1. **Extreme Classification.** In extreme classification problems, popular classification loss functions include *Cross-Entropy Loss*

$$L(\boldsymbol{z}, \mathcal{P}) := \sum_{k \in \mathcal{P}} \log\left(\sum_{j=1}^K \exp(z_j)\right) - z_k \tag{3.38}$$

and *Max-Margin Loss*

$$L(\boldsymbol{z}, \mathcal{P}) := \left[\max_{k \in \mathcal{P}, j \in \mathcal{N}} \; z_k - \; z_j + 1 \right]_+ . \tag{3.39}$$

For multiclass problems, $|\mathcal{P}| = 1$, while for multilabel problems we usually have $|\mathcal{P}| \ll K$. A typical scoring function takes the form

$$f(\boldsymbol{x}) := \boldsymbol{W}\boldsymbol{\phi}(\boldsymbol{x}). \tag{3.40}$$

Here, $\boldsymbol{\phi}(\boldsymbol{x})$ is a feature map constructed either from the domain knowledge or via learning (e.g., a neural network). Both of them fit the structural assumption (3.37).

2. **Metric Learning.** In Metric Learning problems, during training we learn a function

$$f(\boldsymbol{x}) = [-d(\boldsymbol{x}, \boldsymbol{y})]_{\boldsymbol{y} \in \mathcal{Y}}, \tag{3.41}$$

that denotes the dissimilarities of the point $\boldsymbol{x}$ to a collection of points $\boldsymbol{y} \in \mathcal{Y}$. Common choices of the dissimilarity function include the squared Euclidean distance $d(\boldsymbol{x}, \boldsymbol{y}) = \|\boldsymbol{\psi}(\boldsymbol{x}) - \boldsymbol{\psi}(\boldsymbol{y})\|_2^2$ parameterized by a nonlinear transformation $\boldsymbol{\psi} : \mathcal{X} \to \mathbb{R}^d$ for some $d \in \mathbb{N}$, and, more generally, the squared Mahalanobis distance $d(\boldsymbol{x}, \boldsymbol{y}) = (\boldsymbol{\psi}(\boldsymbol{x}) - \boldsymbol{\psi}(\boldsymbol{y}))^\top \boldsymbol{M}(\boldsymbol{\psi}(\boldsymbol{x}) - \boldsymbol{\psi}(\boldsymbol{y}))$ parameterized by $\boldsymbol{\psi}$ and a positive definite matrix $\boldsymbol{M}$. The candidate set $\mathcal{Y}$ could be the whole set of training samples $\{\boldsymbol{x}_i\}_{i=1}^N$, or a collection of latent proxies $\{\boldsymbol{y}_k\}_{k=1}^K$ as suggested by a recent state-of-the-art method [71]. For each sample $(\boldsymbol{x}, \mathcal{P})$, the goal is to learn a distance function s.t. the positive candidates $\mathcal{P}$ are closer to $\boldsymbol{x}$ than the negative ones. Common loss functions for the task are *Neighborhood Component Analysis (NCA) loss* [31]

$$L(\boldsymbol{z}, \mathcal{P}) := \sum_{k \in \mathcal{P}} \log \left(\textstyle\sum_{j=1}^K \exp(z_j) \right) - z_k \tag{3.42}$$

and the *Triplet loss* [103]

$$L(\boldsymbol{z}, \mathcal{P}) = \sum_{k \in \mathcal{P}} \sum_{j \in \mathcal{N}} [z_k - z_j + 1]_+. \tag{3.43}$$

It is easy to see that such scoring functions satisfy the structural assumption (3.37): for the scoring function f given by the squared Mahalanobis distance parameterized by $\boldsymbol{\psi}$ and $\boldsymbol{M}$, the matrix $\boldsymbol{W}$ consists of the rows $\langle -\boldsymbol{\psi}(\boldsymbol{y})^\top \boldsymbol{M}\boldsymbol{\psi}(\boldsymbol{y}), 2\boldsymbol{\psi}(\boldsymbol{y})^\top \boldsymbol{M}, -1 \rangle$ for each $\boldsymbol{y} \in \mathcal{Y}$, and $\boldsymbol{\phi}(\boldsymbol{x}) = \langle 1, \boldsymbol{\psi}(\boldsymbol{x})^\top, \boldsymbol{\psi}(\boldsymbol{x})^\top \boldsymbol{M}\boldsymbol{\psi}(\boldsymbol{x}) \rangle^\top$. Thus the embedding dimension $D = d + 2$.

3. **Word Embeddings.** In the standard *word2vec* training [66], the input space $\mathcal{X}$ is the vocabulary set, and the output space $\mathcal{Y} = \mathcal{X}$; thus K is the vocabulary size. The *Skip-gram* objective learns a scoring function f of the following form:

$$f(\boldsymbol{x}) = \langle \boldsymbol{\phi}(\boldsymbol{y})^\top \boldsymbol{\phi}(\boldsymbol{x}) \rangle_{\boldsymbol{y} \in \mathcal{X}}, \tag{3.44}$$

where $\boldsymbol{\phi}(\cdot)$ is a latent word embedding. This clearly fits the structural assumption (3.37): the rows of the matrix $\boldsymbol{W}$ are the embeddings $\boldsymbol{\phi}(\boldsymbol{y})$ for all $\boldsymbol{y} \in \mathcal{X}$.

Algorithm 10 Loss and Gradient Approximation via Search

A sample $(\boldsymbol{x}, \mathcal{P})$, accuracy parameter $\tau > 0$, and access to a MIPS data structure $\mathcal{T}$ for the rows of $\boldsymbol{W}$. Approximations to $L(f(\boldsymbol{x}), \mathcal{P})$, $\nabla L(f(\boldsymbol{x}), \mathcal{P})$. Query $\mathcal{T}$ with $\phi(\boldsymbol{x})$ and threshold τ to find $\mathcal{S} := \{k \mid |[f(\boldsymbol{x})]_k| > \tau\}$. Construct a sparse approximation $\tilde{\boldsymbol{z}}$ for $f(\boldsymbol{x})$ by setting $\tilde{z}_k = f(\boldsymbol{x})_k$ for $k \in \mathcal{S} \cup \mathcal{P}$, and $\tilde{z}_k = 0$ for $k \notin \mathcal{S} \cup \mathcal{P}$. Return $L(\tilde{\boldsymbol{z}}, \mathcal{P})$ and $\nabla L(\tilde{\boldsymbol{z}}, \mathcal{P})$.

Then given a text corpus $\mathcal{D}$, the loss function[13] for a sample $(\boldsymbol{x}, \mathcal{P})$ where $\mathcal{P}$ is the set of words in the corpus appearing within a certain size window around the input word $\boldsymbol{x}$, is given by

$$L(\boldsymbol{z}, \mathcal{P}) = q_{\boldsymbol{x}} \sum_{\boldsymbol{y} \in \mathcal{P}} q_{\boldsymbol{y}|\boldsymbol{x}} \cdot [\log \left(\textstyle\sum_{\boldsymbol{y}' \in \mathcal{X}} \exp(z_{\boldsymbol{y}'}) \right) - z_{\boldsymbol{y}}] \tag{3.45}$$

where $q_{\boldsymbol{x}}$ is the empirical unigram frequency of $\boldsymbol{x}$ and $q_{\boldsymbol{y}|\boldsymbol{x}}$ is the empirical frequency of observing $\boldsymbol{y}$ within a window of $\boldsymbol{x}$ in the corpus $\mathcal{D}$.

Loss and Gradient Approximation via Search All the loss functions we considered in the applications mentioned share a key feature: their value can be well approximated by the scores of the positive labels and the *largest* scores of the negative labels. Similarly, their gradients are dominated by the coordinates corresponding to the positive labels and the negative labels with the largest scores. For example, the *Max-Margin loss* (3.39) is completely determined by the largest score of the negative labels and the lowest scores of the positive labels, and its gradient is non-zero only on the negative label with largest score and the positive label with lowest score. Similarly, for the *Cross-Entropy loss* (3.38), the coordinates of the gradient corresponding to the negative classes are dominated by the ones with the highest score; the gradient coordinates decrease exponentially as the scores decrease.

This key property suggests the following natural idea for approximating these losses and their gradients: since the the score function f satisfies the linear structural property (3.37), we can compute the largest scores efficiently via a *Maximum Inner Product Search* (MIPS) data structure [87]. This data structure stores a large data set of vectors $\boldsymbol{v}_1, \boldsymbol{v}_2, \ldots, \boldsymbol{v}_K \in \mathbb{R}^D$ and supports queries of the following form: given a target vector $\boldsymbol{u} \in \mathbb{R}^D$ and a threshold τ, it returns the vectors $\boldsymbol{v}_i$ stored in it that satisfy $|\boldsymbol{v}_i^\top \boldsymbol{u}| \geq \tau$ in time that is typically sublinear in K. Thus, we can preprocess $\boldsymbol{W}$ by storing the rows of $\boldsymbol{W}$ in an efficient MIPS data structure. Then for each sample $\boldsymbol{x}$, we can compute the highest scores by querying this data structure with the target vector $\boldsymbol{\phi}(\boldsymbol{x})$ and some reasonable threshold τ, computing approximations to the loss and gradient from the returned vectors (and treating all other scores as 0). This method is depicted in Algorithm 10.

The error in this approximation is naturally bounded by τ times the ℓ_∞ Lipschitz constant of $L(\cdot, \mathcal{P})$. For most loss functions considered in this paper, the ℓ_∞ Lipschitz constant is reasonably small: 2 for *Max-Margin loss*, $O(P_{\max} \log(K))$ for *Cross-Entropy loss* (here, $P_{\max}$ is the maximum number of positive labels for any example), etc.

[13] This is a more compact reformulation of the loss function in [66].

The main difficulty in applying this approach in practice is the curse of dimensionality: the dependence on D is *exponential* for exact methods, and even for approximate methods, such as Locality-Sensitive Hashing, the cost still implicitly depends on the dimension as points become far apart when the intrinsic dimensionality is high [60].

To deal with the curse of dimensionality, we introduce a novel search technique based on dual decomposition. This method, and its analysis, are given in the following section.

In order to apply and analyze the technique, we need the loss functions to be smooth (i.e. have Lipschitz continuous gradients). For non-smooth losses like *Max-Margin loss* (3.39), we apply Nesterov's smoothing technique, which constructs a surrogate loss function with guaranteed approximation quality by adding a strongly convex term to the Fenchel conjugate of the loss:

$$L_\mu(\boldsymbol{z}) := \max_{\boldsymbol{\alpha}} \langle \boldsymbol{z}, \boldsymbol{\alpha} \rangle - \left(L^*(\boldsymbol{\alpha}) + \frac{\mu}{2} \|\boldsymbol{\alpha}\|^2 \right). \tag{3.46}$$

Here, μ is a smoothing parameter that ensures that the surrogate loss has $\frac{1}{\mu}$ Lipschitz continuous gradients while approximating the original loss function to within $O(\mu)$. This *Smoothed Max-Margin loss* has gradient

$$\nabla L(\boldsymbol{z}) := \mathbf{proj}_{\mathcal{C}}(\tfrac{\boldsymbol{z}+\mathbf{1}_{\mathcal{N}}}{\mu}) \tag{3.47}$$

where $\mathbf{1}_{\mathcal{N}}$ denotes a vector containing 0 for indices $k \in \mathcal{P}$ and 1 for $k \in \mathcal{N}$, and $\mathbf{proj}_{\mathcal{C}}(.)$ denotes the projection onto the bi-simplex $\mathcal{C} = \{\boldsymbol{\alpha} \mid \sum_{k\in\mathcal{N}} \alpha_k = \sum_{k\in\mathcal{P}} -\alpha_k \leq 1,\ \alpha_{\mathcal{N}} \geq 0, \alpha_{\mathcal{P}} \leq 0\}$. The Smoothed Max-Margin loss and its gradient can again be computed using the largest few scores.

3.3.2 Loss-Decomposition-Guided Search

We now describe our loss decomposition method. Recall the linear structural assumption (3.37): $f(\boldsymbol{x}) = \boldsymbol{W}\boldsymbol{\phi}(\boldsymbol{x})$ for all $\boldsymbol{x} \in \mathcal{X}$. In this section, we will keep $(\boldsymbol{x}, \mathcal{P})$ fixed, and we will drop the dependence on $\mathcal{P}$ in L for convenience and simply use the notation $L(f(\boldsymbol{x}))$ and $\nabla L(f(\boldsymbol{x}))$.

While MIPS over the D-dimensional rows of $\boldsymbol{W}$ can be computationally expensive, we can exploit the linear structure of f by decomposing it: chunking the D coordinates of the vectors in $\mathbb{R}^D$ into B blocks, each of size D/B. Here $B \in \mathbb{N}$ is an integer; larger B leads to easier MIPS problems but reduces accuracy of approximations produced. Let $\boldsymbol{W}^{(1)}, \boldsymbol{W}^{(2)}, \ldots, \boldsymbol{W}^{(B)}$ be the corresponding block partitioning of $\boldsymbol{W}$ obtained by grouping together the columns corresponding to the coordinates in each block. Similarly, let $\boldsymbol{\phi}^{(1)}(\boldsymbol{x}), \boldsymbol{\phi}^{(2)}(\boldsymbol{x}), \ldots, \boldsymbol{\phi}^{(B)}(\boldsymbol{x})$ be the conformal partitioning of the coordinates of $\boldsymbol{\phi}(\boldsymbol{x})$.

Now define the overall score vector $\boldsymbol{z} := f(\boldsymbol{x}) = \boldsymbol{W}\boldsymbol{\phi}(\boldsymbol{x})$, and per-chunk score vectors $\boldsymbol{z}_j = \boldsymbol{W}^{(j)}\boldsymbol{\phi}^{(i)}(\boldsymbol{x})$, for $j \in [B]$. Then we have $\boldsymbol{z} = \sum_{j=1}^{B} \boldsymbol{z}_j$, in other words, we have a decomposition of the score vector. The following theorem states that the loss of a decomposable score vector can itself be decomposed into several parts connected through a set of message variables. This theorem is key to decoupling the variables into lower dimensional chunks that can be optimized separately via an efficient MIPS data structure. While this theorem can be derived by applying dual decomposition to the convex conjugate of the loss function, here we provide a simpler direct proof by construction.

Algorithm 11 Greedy Message Passing

a sample $\boldsymbol{x}$, threshold parameters $\tau_1, \tau_2 > 0$, and access to B MIPS data structures $\mathcal{T}_j$ storing the rows of $\boldsymbol{W}^{(j)}$, for $j \in [B]$ Approximation to $L(f(\boldsymbol{x}))$ and $\nabla L(f(\boldsymbol{x}))$. Query $\mathcal{T}_j$ with $\boldsymbol{\phi}^{(j)}(\boldsymbol{x})$ and threshold τ to find $\mathcal{S}_j := \{k \mid |[\boldsymbol{z}_j]_k| > \tau_1\}$. Construct a sparse approximation $\tilde{\boldsymbol{z}}_j$ for $\boldsymbol{z}_j$ by setting $[\tilde{\boldsymbol{z}}_j]_k = [\boldsymbol{z}_j]_k$ for $k \in \mathcal{S}_j \cup \mathcal{P}$, and $[\tilde{\boldsymbol{z}}_j]_k = 0$ for $k \notin \mathcal{S} \cup \mathcal{P}$. **for** $t = 1, 2, \ldots$ (until converged) **do**

4: Compute the set

$$\mathcal{A} := \bigcup_{j \in [B]} \{k \mid |[\nabla L(B(\tilde{\boldsymbol{z}}_j + \boldsymbol{\lambda}_j))]_k| > \tau_2\}.$$

5: Compute $[\boldsymbol{\lambda}_j^*]_k = \frac{1}{B}[\tilde{\boldsymbol{z}}]_k - [\tilde{\boldsymbol{z}}_j]_k$ for all $k \in \mathcal{A}$ and all $j \in [B]$.

6: Compute the step size $\eta = \frac{2}{t+2}$.

7: For all $k \in \mathcal{A}$ and all $j \in [B]$, update

$$[\boldsymbol{\lambda}_j]_k \leftarrow \eta[\boldsymbol{\lambda}_j^*]_k + (1-\eta)[\boldsymbol{\lambda}_j]_k.$$

8: **end for**

9: Return $\frac{1}{B}\sum_{j=1}^{B} L(B(\tilde{\boldsymbol{z}}_j + \boldsymbol{\lambda}_j))$ and $\frac{1}{B}\sum_{j=1}^{B} \nabla L(B(\tilde{\boldsymbol{z}}_j + \boldsymbol{\lambda}_j))$.

Theorem 12. *Let $L : \mathbb{R}^D \to \mathbb{R}$ be a convex function, and let $\boldsymbol{z} \in \mathbb{R}^D$ be decomposed as a sum of B vectors as follows: $\boldsymbol{z} = \sum_{j=1}^{B} \boldsymbol{z}_j$. Then $L(\boldsymbol{z})$ is equal to the optimum value of the following convex minimization problem:*

$$\min_{\boldsymbol{\lambda}_j \in \mathbb{R}^D,\, j \in [B]} \frac{1}{B}\sum_{j=1}^{B} L(B(\boldsymbol{z}_j + \boldsymbol{\lambda}_j)) \; s.t. \; \sum_{j=1}^{B} \boldsymbol{\lambda}_j = 0. \tag{3.48}$$

Proof. First, for any $\boldsymbol{\lambda}_1, \boldsymbol{\lambda}_2, \ldots, \boldsymbol{\lambda}_B \in \mathbb{R}^D$ such that $\sum_{j=1}^{B} \boldsymbol{\lambda}_j = 0$, by Jensen's inequality applied to the convex function L, we have $L(\boldsymbol{z}) \leq \frac{1}{B}\sum_{j=1}^{B} L(B(\boldsymbol{z}_j + \boldsymbol{\lambda}_j))$. On the other hand, if we set $\boldsymbol{\lambda}_j = \frac{1}{B}\boldsymbol{z} - \boldsymbol{z}_j$ for all $j \in [B]$, we have $L(\boldsymbol{z}) = \frac{1}{B}\sum_{j=1}^{B} L(B(\boldsymbol{z}_j + \boldsymbol{\lambda}_j))$. □

Loss Decomposition Guided Search (LDGS) Theorem (12) is the basis for our algorithm for computing approximations to the loss and its gradient. This approximation is computed by approximately solving the convex minimization problem (3.48) *without computing the whole score vector* $\boldsymbol{z}$, using a form of descent method on the $\boldsymbol{\lambda}_j$ variables (which we refer to as "message passing"). The gradient computations required for each step can be (approximately) done using an efficient MIPS data structure storing the D/B dimensional rows of $\boldsymbol{W}^{(j)}$. The details of the algorithm are given in Algorithm 13. It can be viewed as running a version of the Frank-Wolfe algorithm on an appropriate convex function.

A sublinear in K time implementation of step 5 in the algorithm relies on the fact that both $\tilde{\boldsymbol{z}}_j$ and $\boldsymbol{\lambda}_j$ are sparse vectors, which in turn relies on the fact that gradients of the loss functions of interest are either sparse or concentrated on a few coordinates. Step 9 in the algorithm moves the current solution towards the optimal solution $\boldsymbol{\lambda}_j^*$ that we have a closed form formula for, thanks

to the constructive proof of Theorem (12). This movement is only done for the set of coordinates of the gradients of high magnitude identified in step 5 of the algorithm, thus ensuring that only a few coordinates are updated. Thus essentially the algorithm is performing a greedy descent towards the optimal solution.

Error Analysis Define $\tilde{\boldsymbol{z}} = \sum_{j=1}^{B} \tilde{\boldsymbol{z}}_j$. Note that $\|\boldsymbol{z} - \tilde{\boldsymbol{z}}\|_\infty \leq B\tau_1$, so the error in approximating $L(\boldsymbol{z})$ by $L(\tilde{\boldsymbol{z}})$ is at most $B\tau_1$ times the ℓ_∞ Lipschitz constant of L, which is typically small as explained earlier. The algorithm essentially a runs a Frank-Wolfe type method to converge to $L(\tilde{\boldsymbol{z}})$. In the following, we analyze the convergence rate of the greedy message passing algorithm (Algorithm 13) to $L(\tilde{\boldsymbol{z}})$. The analysis relies on smoothness of the loss function. A function is said to be $1/\mu$-smooth if its gradients are Lipschitz continuous with constant $1/\mu$. For the *Cross-Entropy loss* (3.38) we have $\mu = 1$, and for the *smoothed max-margin loss* (3.46), μ is a tunable parameter, and we found setting $\mu \in [1, 5]$ works well in our experiments.

To analyze the algorithm, denote by $\boldsymbol{\Lambda}$ the BK dimensional vector $\langle \boldsymbol{\lambda}_1, \boldsymbol{\lambda}_2, \ldots, \boldsymbol{\lambda}_B \rangle$ in any given step in the loop of the algorithm. Similarly let $\boldsymbol{\Lambda}^*$ denote the BK dimensional vector composed of $\boldsymbol{\lambda}_j^*$. Define $G(\boldsymbol{\Lambda}) = \frac{1}{B}\sum_{j=1}^{B} L(B(\tilde{\boldsymbol{z}}_j + \boldsymbol{\lambda}_j))$, i.e. the objective function in (3.48).

Theorem 13 (Greedy Message Passing)**.** *Suppose the loss function L is $1/\mu$-smooth. Then the suboptimality gap of $\boldsymbol{\Lambda}$ in the t-th step of the loop can be bounded as follows:*

$$G(\boldsymbol{\Lambda}) - G(\boldsymbol{\Lambda}^*) \leq \frac{2B\|\boldsymbol{\Lambda}^*\|^2}{\mu(t+2)} + 2\tau_2 \ln(t)\|\boldsymbol{\Lambda}^*\|_1$$

Proof. Since the loss function L is $1/\mu$-smooth, it is easy to check that G is B/μ-smooth. Thus, if $\Delta\boldsymbol{\Lambda}$ is the change in $\boldsymbol{\Lambda}$ in a given step of the loop in the algorithm, then

$$G(\boldsymbol{\Lambda} + \Delta\boldsymbol{\Lambda}) - G(\boldsymbol{\Lambda}) \leq \eta\langle \nabla G(\boldsymbol{\Lambda}), \Delta\boldsymbol{\Lambda}\rangle + \frac{\eta^2 B}{2\mu}\|\Delta\boldsymbol{\Lambda}\|^2.$$

Note that $\Delta\boldsymbol{\Lambda}$ equals $\boldsymbol{\Lambda}^* - \boldsymbol{\Lambda}$ in all coordinates except those corresponding to $k \notin \mathcal{A}$ for all $j \in [B]$, and the magnitude of the gradient in those coordinates is at most τ_2. Thus we have $\langle \nabla G(\boldsymbol{\Lambda}), \Delta\boldsymbol{\Lambda}\rangle \leq \langle \nabla G(\boldsymbol{\Lambda}), \boldsymbol{\Lambda}^* - \boldsymbol{\Lambda}\rangle + \tau_2\|\boldsymbol{\Lambda}^*\|_1$. Here, we used the fact that each coordinate of $\boldsymbol{\Lambda}$ lies between 0 and the corresponding coordinate of $\boldsymbol{\Lambda}^*$. Next, by the convexity of G, we have $\langle \nabla G(\boldsymbol{\Lambda}), \boldsymbol{\Lambda}^* - \boldsymbol{\Lambda}\rangle \leq G(\boldsymbol{\Lambda}^*) - G(\boldsymbol{\Lambda})$. Putting all the bounds together and following some algebraic manipulations, we have

$$\begin{aligned} &G(\boldsymbol{\Lambda} + \Delta\boldsymbol{\Lambda}) - G(\boldsymbol{\Lambda}^*) \\ &\leq (1-\eta)(G(\boldsymbol{\Lambda}) - G(\boldsymbol{\Lambda}^*)) + \eta\tau_2\|\boldsymbol{\Lambda}^*\|_1 + \frac{\eta^2 B}{2\mu}\|\boldsymbol{\Lambda}^*\|^2. \end{aligned} \tag{3.49}$$

Here, we used the fact that each coordinate of $\boldsymbol{\Lambda}$ lies between 0 and the corresponding coordinate of $\boldsymbol{\Lambda}^*$ to get the bound $\|\Delta\boldsymbol{\Lambda}\|^2 \leq \|\boldsymbol{\Lambda}^*\|^2$.

Now, using the fact that $\eta = \frac{2}{t+2}$ in iteration t, a simple induction on t implies the claimed bound on $G(\boldsymbol{\Lambda}) - G(\boldsymbol{\Lambda}^*)$.

□

Thus, to ensure that the suboptimality gap is at most ϵ, it suffices to run the greedy procedure for $T = \frac{B\|\mathbf{\Lambda}^*\|^2}{4\mu\epsilon}$ steps with $\tau_2 = \frac{\epsilon}{4\ln(T)\|\mathbf{\Lambda}^*\|_1}$. While this theorem provides a proof of convergence for the algorithm to any desired error level, the bound it provides is quite weak. In practice, we found that running just *one step* of the loop suffices to improve performance over direct search-based methods.

If, in addition to being smooth, the loss function is also strongly convex (which can be achieved by adding some ℓ_2^2 regularization, for instance) then we can also show convergence of the gradients. This is because for strongly convex functions the convergence of gradients can be bounded by in terms of the convergence of the loss value. This is a very standard analysis and we omit it for the sake of clarity.

Cost Analysis. Exact gradient evaluation for a single sample can be computed in $O(DK)$ time. Directly applying a search-based gradient approximation (Algorithm 10) has a cost of $O(DQ_D(K))$, where $Q_D(K)$ is the number of classes retrieved in the MIPS data structure in order to find all classes of significant gradients. The query cost $Q_D(K)$ has a strong dependency on the dimension D.

Exact MIPS has a cost $Q_D(K)$ exponential in D [60, 87]. For approximate search methods, such as Locality Sensitive Hashing (LSH), the cost $Q_D(K)$ typically only implicitly depends on the dimension. Our method (Algorithm 13) divides D into B subproblems of dimension D/B with a cost per message passing iteration of $O(DQ_{D/B}(K) + DB|\mathcal{A}|)$, where $\mathcal{A}$ is the set computed in step 4 of Algorithm 13. Note $Q_{D/B}(K)$ decreases with B rapidly (exponentially in the exact case) and therefore one can select B such that $Q_{D/B}(K) \ll Q_D(K)$ and balance two terms s.t. $(DQ_{D/B}(K) + DB|\mathcal{A}|) \ll DK$.

3.3.3 Practical Consideration

MIPS queries. In practice when using the MIPS data strcuctures, instead of retrieving all classes with scores more than the threshold τ_1, it is more efficient to to retrieve the top Q classes with the highest scores. In our implementation, we use *Spherical Clustering* [5] as the MIPS data structure, where the number of clusters C is selected such that $K/C \leq Q$ and $C \leq Q$. Note this requires $Q \geq \sqrt{K}$, leading to a speedup bounded by $\sqrt{K}$. Similarly, for computing the active set $\mathcal{A}$ in step 4 of Algorithm 13, we can compute an appropriate threshold τ_2 using the properties of the loss function. In the case of margin-based losses, (3.39) and (3.43), and their smoothed versions (3.46), the gradient is sparse so τ_2 can be set to 0 or some very small value. Loss functions like (3.38), (3.42) typically have exponentially decayed gradient magnitudes over the non-confusing negative classes. For these losses, classes can be retrieved in decreasing order of gradient magnitude, using a lower bound on the partition function $Z = \sum_k \exp z_k$ summing over only the subset of retrieved classes in order to decide whether more classes need to be retrieved or not.

Updates of data structures. During training the model parameters determining f will change, and the data structures $\mathcal{T}_j$ need to be updated. These data structures stores rows of $\boldsymbol{W}$ and treats $\boldsymbol{\phi}(\boldsymbol{x})$ as query. For loss functions with a sparse gradient, such as (3.39), (3.43), and their

smoothed versions (3.46), the number of updated rows of $\boldsymbol{W}$, k_r, is much smaller than K and Q (the number of classes retrieved for a query). Thus the cost for re-indexing rows of $\boldsymbol{W}$ is $k_r C(D/B)B = k_r CD$, where C is the number of inner products required to index each row, which is much smaller than the costs of query and updates. For tasks with large number of updated rows ($k_r \approx Q$), the method is still effective with a larger mini-batch size N_b. As the costs of query and updates grow with N_b while the number of rows to re-index is bounded by K, the cost of maintaining data structure becomes insignificant.

Sampling for initialization. For a randomly initialized model, the early iterates of learning have gradients evenly distributed over the classes, as the scores of all classes are close to each other. Therefore, it is unnecessary to search candidates of significant gradient magnitude in the early stage. In practice, one can switch from a sampling-based gradient approximation to a search-based gradient approximation after a number of mini-batch updates.

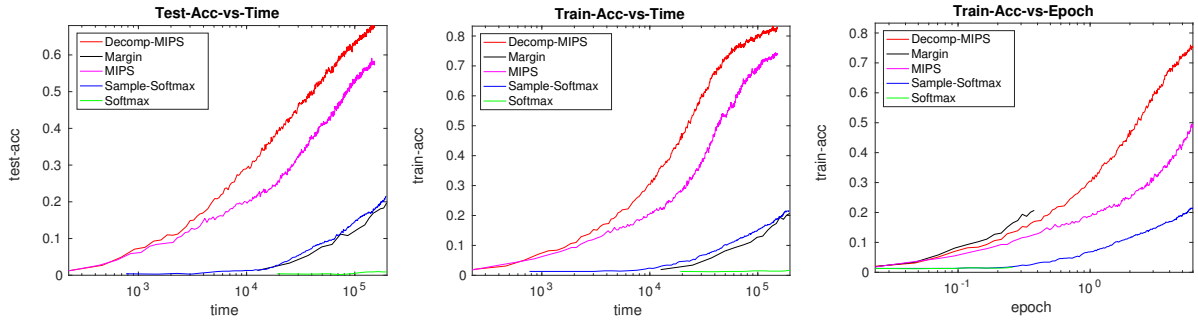

Figure 3.5: Results of multiclass classification on the *Megaface* data set: Test Accuracy vs. Training time (left), Test Accuracy vs. Training Time (middle), and Training Accuracy vs. number of epochs (right). Note the x-axis is in log-scale.

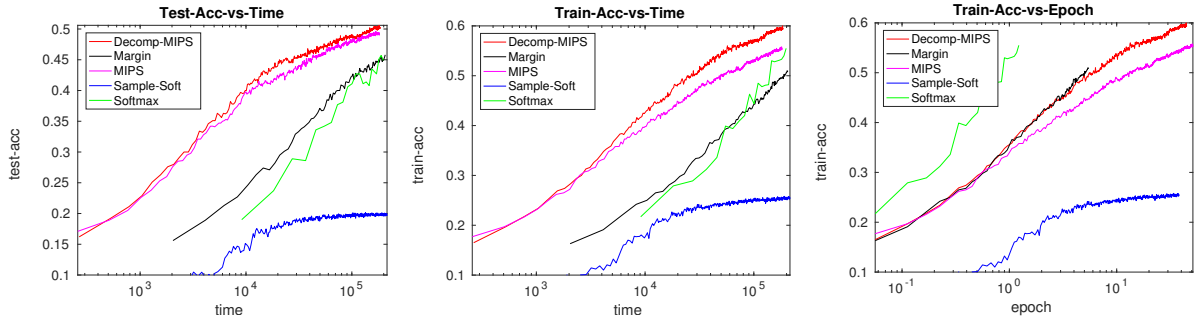

Figure 3.6: Results of multilabel classification on the *WikiLSHTC* data set: Test Accuracy vs. Training time (left), Test Accuracy vs. Training Time (middle), and Training Accuracy vs. number of epochs (right). Note the x-axis is in log-scale.

3.3.4 Experiments

In this section, we conduct experiments on three types of problems: (i) multiclass classification (face recognition), (ii) multilabel classification (document tagging), and (iii) Unsupervised Word Embedding (Skip-gram objective (3.45)). For multiclass and multilabel classification, we employ a Stochastic Gradient Descent (SGD) optimization algorithm, with an initial step size chosen from $\{1, 0.1, 0.01\}$ for the best performance of each method, with a $1/(1+t)$ cooling scheme where t is the iteration counter. The mini-batch size is 10 and all methods are parallelized with 10 CPU cores in a shared-memory architecture, running on a dedicated machine. All the implementation are in C++. The following loss functions and gradient evaluation methods are compared for the experiments on multiclass and multilabel classification:

- **Softmax**: exact gradient evaluation of the cross-entropy loss (3.38). For multiclass, we have $|\mathcal{P}| = 1$ and for multilabel, $|\mathcal{P}| \ll K$.
- **Sampled-Softmax**: the sampling strategy in [14, 47], which includes all positive classes of the instances and uniformly subsamples from the remaining negative classes. Here we choose sample size as $K/100$.
- **Margin**: exact gradient evaluation of the *smoothed max-margin loss* (3.46), where we choose $\mu = 1$ for the case of multiclass, and $\mu = 5$ for the case of multilabel. The bi-simplex projection (3.47) is computed in $O(K \log K)$ using the procedure described in [118]. Note the gradient update for this loss is faster than that for *cross-entropy*, as the loss gradient is very sparse, making the backward pass much faster.
- **MIPS**: search-based gradient evaluation (Algorithm 10) with *smoothed max-margin loss* (same setting to **Margin**). We use *Spherical Clustering* [5] with 100 centroids as the MIPS data structure, and a batch query of size $K/100$.
- **Decomp-MIPS**: gradient evaluation via decomposed search (Algorithm 13, $T = 1$ iteration). We divide the inner product into $B = 8$ factors in the multiclass experiment and $B = 4$ in the multilabel case. The settings for MIPS data structure are the same as above.

#Identities	#Images	Embed. Dim.
672K	4.7M	128

Table 3.3: Statistics of the MegaFace dataset.

Multiclass Classificatoin For multiclass classification we conduct experiments on the largest publicly available facial recognition dataset *MegaFace* (Challenge 2)[14], where each identity is considered a class, and each sample is an image cropped by a face detector. The data set statistics are shown in Table 3.3.

We employ the *FaceNet* architecture [83][15] pre-trained on the *MS-Celeb-1M* dataset, and fine-tune its last layer on the *MegaFace* dataset. The input of the last layer is an embedding

[14] `http://megaface.cs.washington.edu/`.
[15] `github.com/davidsandberg/facenet`

of size 128, which is divided into $B = 8$ factors, each of dimension 16, in the *Decomp-MIPS* method.

The result is shown in Figure 3.5, where all methods are run for more than one day. Firstly, comparing methods that optimize the (smoothed) max-margin loss (*Decomp-MIPS*, *MIPS* and *Margin*) shows that both *Decomp-MIPS*, *MIPS* speed up the iterates by $1 \sim 2$ orders of magnitude. However, *MIPS* converges at an accuracy much lower than *Decomp-MIPS* and the gap gets bigger when running for more iterations. Note the time and epochs are in log scale. Secondly, *Softmax* has a much slower progress compared to *Margin*. Note both of them do not even finish one epoch (4.7M samples) after one day, while the progress of *Margin* is much better, presumably because its focus on the *confusing identities*. *Sampled-Softmax* has much faster iterates, but the progress per iterate is small, leading to slower overall progress compared to the MIPS-based approaches.

Multilabel Classification For multilabel classification, we conduct experiments on *WikiLSHTC* [76], a benchmark data set in the Extreme Classification Repository[16], where each class is a catalog tag in the Wikipedia, and each sample is a document with bag of words representation. The data statistics are shown in Table 3.4.

We train a one-hidden-layer fully-connected feedforward network for the multilabel classification task. The first layer has input dimension equal to the vocabulary size (1.6M) and an output of dimension 100. The second layer has output size equal to the number of classes (325K), with different loss functions and approximations for different methods in comparison. The training result also produces document and work embedding as by-products. For *Decomp-MIPS*, the input of the last layer is divided into $B = 4$ factors, each of dimension 25.

We run all the compared methods for more than one day and the result is shown in Figure 3.6. First, for this multilabel task, *Softmax* has very good per-iteration progress, significantly more than that from the other three approaches based on the smoothed max-margin loss (*Margin*, *MIPS*, *Decomp-MIPS*). However, the iterates of *Softmax* are much slower than the others as it has a dense loss gradient and thus a slower backpropagation, so that when comparing training time, *Softmax* performs similarly to *Margin*. On the other hand, when comparing *Margin Decomp-MIPS*, and *MIPS* in progress per epoch, the updates of *Decomp-MIPS* achieve almost the same progress as the exact gradient calculation of *Margin*, while *MIPS* has a significant drop in its training accuracy compared with *Margin* and *Decomp-MIPS*, since it runs for more iterations. Overall, the *MIPS*-based methods lead to an order of magnitude speedup, while *Decomp-MIPS* retains the accuracy of the exact method. On the other hand, *Sampled-Softmax* has an extremely slow per-iteration progress despite its fast iterates, and could not reach a comparable accuracy to other methods even after one day.

#Label	#Sample	Embed. Dim.	Vocab. Size
325K	1.8M	100	1.6M

Table 3.4: Statistics of the *WikiLSHTC* data set. On average, each sample has 3.19 positive labels, and each class appears in 17.46 samples as a positive class.

[16] `manikvarma.org/downloads/XC/XMLRepository.html`

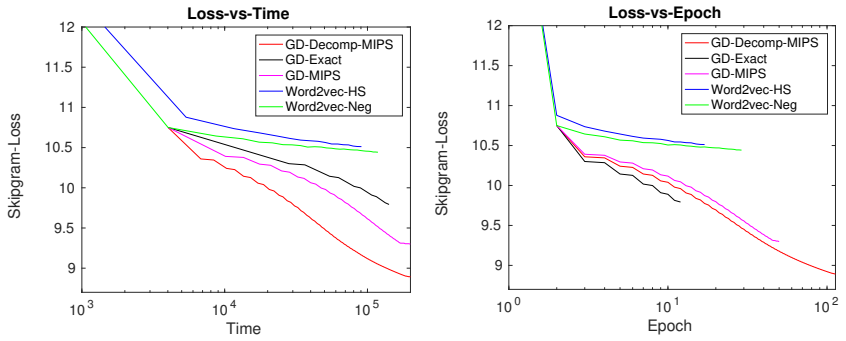

Figure 3.7: Results on word embedding with *Skip-gram* objective, where *GD-Exact*, *GD-MIPS*, and *GD-Decomp-MIPS* are initialized with a model trained by one epoch of *Word2vec-Neg*.

Vocab. Size	#Words	Embed. Dim.	Window Size
$\approx$ 451K	$\approx$ 680M	100	8

Table 3.5: Statistics of the BillionW dataset.

Unsupervised Word Embedding In this section, we evaluate the proposed gradient approximation technique on the word embedding task with the *Skip-gram* learning objective (3.45) and compare it with two widely-used gradient approximation methods — *Hierarchical Softmax (Word2vec-HS)* and *Negative Sampling (Word2vec-Neg)* [66] implemented in the *word2vec*[17] package released by the authors. The sample size for *Word2vec-Neg* is selected from $\{5, 10, 15, 20, 25\}$.

We use the benchmark data set *BillonW*[18] of almost a half million vocabulary size. The data statistics are provided in Table 3.5. Following [66], we use a window of size 8 and subsample *frequent words* in the corpus. Each word w is dropped with probability $\max\{1 - \sqrt{\frac{t}{f_w}}, 0\}$ where f_w is the relative frequency of the word in the corpus, and $t = 10^{-4}$ is a threshold parameter.

Note that the *Skip-gram* objective (3.45) is presented in a collapsed form equivalent to the one in [66]. Here, all terms of the same input-output pairs are grouped together and weighted by the frequency.

We compute gradients from the positive outputs by summing over the empirical input-output distribution $q_{\boldsymbol{x}}$, $q_{\boldsymbol{y}|\boldsymbol{x}}$ in (3.45). Then we perform gradient descent (GD) updates on the parameters of input words $\{\boldsymbol{\phi}(\boldsymbol{x})\}_{\boldsymbol{x}\in\mathcal{X}}$ and output words $\{\boldsymbol{\phi}(\boldsymbol{y})\}_{\boldsymbol{y}\in\mathcal{X}}$ alternately. We use *GD*, *GD-MIPS* and *GD-Decomp-MIPS* to denote the algorithm with different strategies of loss approximations. As mentioned in Section 3.3.3, since in the early iterates the model has quite evenly distributed gradient over candidates, we use 1 epoch of *Word2vec-Neg* to initialize *GD*, *GD-MIPS* and *GD-Decomp-MIPS*. For this task, we have many more negative classes of significant gradient magnitude than in the multilabel and multiclass experiments. So we use a batch query of size $K/20$ instead of $K/100$ to the MIPS structure. All the compared methods are parallelized with 24 CPU cores.

The results are shown in Figure 3.7. After the first epoch, methods based on alternating gradient descent (GD) (with the collapsed objective (3.45)) have faster convergence per epoch, and the

[17] `code.google.com/archive/p/word2vec/`
[18] `www.statmt.org/lm-benchmark/`

iterations of *GD-Deomp-MIPS* are 5 times faster than those of *GD* while having a significantly better objective value than *GD-MIPS* for the same training time.

Chapter 4

Convex Estimator for Latent-Variable Models

The principle of high-dimensional sparse estimation not only helps us develop new algorithms for existing learning objective, but also equipps us with new tools when it comes to the design of new estimators and learning objectives. In this chapter, we show how the ability to handle problems of huge number of variables allows us to develop convex estimators for Latent-Variable Models, such as *Latent-Feature Model* and *Generalized Mixed Regression* in section 4.1 and 4.2 respectively, that enjoy strong approximation guarantees without any restrictive assumption on the data [122].

4.1 Latent-Feature Models

Latent variable models are widely used in unsupervised learning, in part because they provide compact and interpretable representations of the distribution over the observed data. The most common and simplest such latent variable model is a mixture model, which associates each observed object with a *latent class*. However, in many real-world applications, observations are better described by a combination of *latent features* than a single latent class. Accordingly, admixture or mixed membership models have been proposed [2], that in the simplest settings, assign each object to a convex combination of latent classes. For instance, a document object could be modeled as a convex combination of topic objects. There are many settings however where a convex combination might be too restrictive, and the objects are better modeled as simply a collection of latent classes. An example is web image, which can often described by multiple tags rather than a single class, or even by a convex combination of tag objects. Another example is the model of user, who might have multiple interests in the context of a recommendation system, or be involved in multiple communities in a social network. With such settings in mind, [32] proposed a latent feature model (LFM), where each observed object can be represented by a binary vector that indicates the presence or absence of each of a collection of *latent features*. Their proposed model extended earlier models with a similar flavor for specialized settings, such as [98] for bag of words models for text. The latent feature model can also be connected to sparse PCA models [22, 49] by considering a pointwise product of the binary feature incidence

vector with another real-valued vector. As [32] showed, LFM handily outperforms clustering as an efficient and interpretable data representation, particularly in settings where the object can be naturally represented as a collection of latent features or parts.

However, the estimation (inference) of an LFM from data is difficult, due to the combinatorial nature of the binary feature incidence vectors. Indeed, with N samples, and K latent features, the number of possible binary matrices consisting of the N binary feature incidence vectors is 2^{NK}. And not in the least, the log-likelihood of LFM is not a concave function of its parameters.

Given that the finite feature case seems intractable, right from the outset, attention has focused on the nonparametric infinite feature case, where a prior known as the *Indian Buffet Process (IBP)* has been proposed for the infinite binary matrices consisting of the feature incidence vectors given infinite set of latent features [33]. While the IBP prior provides useful structure, inference remains a difficult problem, and in practice, one often relies on local search methods [11] to find an estimate of parameters, or employ Markov Chain Monte Carlo (MCMC) [24] or variational methods [25] to obtain an approximate posterior distribution. However, none of these approaches can provide guarantees on the quality of solution in polynomial time.

Note that both in the mixture model, as well as the admixture model cases, the parametric variants have been hugely popular alongside or perhaps even more so than the nonparametric variants e.g. clustering procedures based on finite number of clusters, or topic models with a finite number of topics. This is in part because the parametric variants have a lower model complexity, which might be desired under certain settings, and also have simpler inference procedures. However, in the LFM case, the parametric variant has received very little attention, which might suggest the relatively lesser popularity for LFMs when compared to mixture or admixture/topic models.

Accordingly, we consider the question of computationally tractable estimation of parametric LFMs. In the nonparametric setting with an IBP prior, [97] have proposed the use of spectral methods, which bypasses the problem of non-concave log-likelihood by estimating the *moments* derived from the model, and then recovers parameters by solving a system of equations. Their spectral methods based procedure produces consistent estimates of LFMs in polynomial time, however with a sample complexity that has a high-order (more than six-order) polynomial dependency on the number of latent features and the occurrence probability of each feature. Moreover, the application of spectral methods requires knowledge of the distribution, which results in non-robustness to model mis-specification in practice. Under a noiseless setting, [88] leveraged identifiability conditions under which the solution is unique, to propose an algorithm for a parametric LFM. Their algorithm is guaranteed to recover the parameters in the noiseless setting, but with the caveat that it has a computational complexity that is *exponential* in the number of latent features.

We note that even under the assumption of a nonparametric LFM, specifically an *Indian Buffet Process with Linear Gaussian Observations*, deriving its MAP point estimate under low-variance asymptotics following the approach of *MAD-Bayes Asymptotics* [11] yields an objective similar to that of a parametric LFM with an additional term that is linear in the number of latent features. Thus, developing computationally tractable approaches for parametric LFMs would be broadly useful. In the following, we propose the *Latent Feature Lasso*, a novel convex estimation procedure for the estimation of a Latent Feature Model using atomic-norm regularization. We construct a greedy algorithm with strong optimization guarantees for the estimator by relating

each greedy step to a MAX-CUT like problem. We also provide a risk bound for the estimator under general data distribution settings, which trades off between risk and sparsity, and has a sample complexity linear in the number of components and dimension. Under the noiseless setting, we also show that Latent Feature Lasso estimator recovers the parameters of LFM under an identifiability condition similar to that proposed in [88].

4.1.1 Problem Setup

A Latent Feature Model represents data as a combination of *latent features*. Let $x \in \mathbb{R}^D$ be an observed random vector that is generated as:

$$x = W^T z + e,$$

where $z \in \{0,1\}^K$ is a latent binary feature incidence vector that denotes the presence or absence of K features, $W \in \mathbb{R}^{K \times D}$ is an unknown matrix of K latent features of dimension D, and $e \in \mathbb{R}^D$ is an unknown noise vector. We say that the model is biased when $E[e|z] = E[x|z] - W^T z \neq 0$, and which we allow in our analysis. Suppose we observe N samples of the random vector x. It will be useful in the sequel to collate the various vectors corresponding to the N samples into matrices. We collate the observations into a matrix $X \in \mathbb{R}^{N \times D}$, the N latent incidence vectors into a matrix $Z \in \{0,1\}^{N \times K}$, and the noise vectors into an $N \times D$ matrix ϵ. We thus obtained the vectorized form of the model as $X = ZW + \epsilon$.

Most existing works on LFM make two strong assumptions. The first is that the model has zero bias $\mathcal{E}[e|z] = 0$ [11, 24, 25, 33, 39, 88, 97, 129]. The second common but strong class of assumptions is distributional [39, 97]:

$$p(x|z) = N(W^T z, \sigma^2 I) \ , \ p(z) = Bern(\pi),$$

where $Bern(\pi)$ denotes the distribution of K independent Bernoulli with $z_k \sim Bern(\pi_k)$. In the Nonparametric Bayesian setting [11, 24, 25, 33, 129], one replaces $Bern(\pi)$ with an *Indian Buffet Process* IBP(α) over the $N \times K^+$ binary incidence matrix $Z \in \{0,1\}^{N \times K^+}$ where K^+ can be inferred from data instead of being specified a-priori. We note that both classes of assumptions need not hold in practice: the zero bias assumption $E[x|z] = W^T z$ is stringent given the linearity of the model, while the Bernoulli and IBP distributional assumptions are also restrictive, in part since they assume independence between the presence of two features z_{ik} and $z_{ik'}$. Our method and analyses do not impose either of these assumptions.

It is useful to contrast the different estimation goals ranging over the LFM literature. In the Bayesian approach line of work [11, 24, 33, 39, 129], the goal is to infer the posterior distribution $P(Z, W|X)$ given X. The line of work using Spectral Methods [97] on the other hand aim to estimate $p(z)$, $p(x|z)$ in turn by estimating parameters (π, W). In some other work [88], they aim to estimate W, leaving the distribution of z unmodeled. In this paper, we focus on the more realistic setting where we make *no assumption* on $p(x)$ except that of boundedness, and aim to find an LFM W^* that minimizes the risk

$$r(W) := E[\min_{z \in \{0,1\}^K} \frac{1}{2}\|x - W^T z\|^2]. \tag{4.1}$$

where the expectation is over the random observation x.

4.1.2 Latent Feature Lasso

We first consider the non-convex formulation that was also previously studied in [11] as asymptotics of the MAP estimator of IBP Linear-Gaussian model:

$$\min_{K\in\mathbb{N},Z\in\{0,1\}^{N\times K},W\in\mathbb{R}^{K\times D}} \frac{1}{2N}\|X-ZW\|_F^2+\lambda K. \tag{4.2}$$

The estimation problem in [88] could also be cast in the above form with $\lambda = 0$ and K treated as a fixed hyper-parameter, while [11] treats K as a variable and controls it through λ. (4.2) is a combinatorial optimization of $N \times K + 1$ integer variables. In the following we develop a tight convex approximation to (4.2) with ℓ_2 regularization on W, by introducing a type of atomic norm [13].

For a fixed K, Z, consider the minimization over W of the ℓ_2 regularized version of (4.2)

$$\min_{W\in\mathbb{R}^{K\times D}} \frac{1}{2N}\|X-ZW\|_F^2+\frac{\tau}{2}\|W\|_F^2, \tag{4.3}$$

which is a convex minimization problem. Applying Lagrangian duality to (4.3) results in the following dual form

$$\max_{A\in\mathbb{R}^{N\times D}} \left\{\frac{-1}{2N^2\tau}tr(AA^TM)-\frac{1}{N}\sum_{i=1}^{N}L^*(x_i,-A_{i,:})\right\}. \tag{4.4}$$

where $M := ZZ^T$, $A \in \mathbb{R}^{N\times D}$ are dual variables that satisfy $W^* = \frac{1}{N}Z^*A^*$ at the optimum of (4.3) and (4.4), and $L^*(x,\alpha) = \langle x,\alpha\rangle + \frac{1}{2}\|\alpha\|^2$ is the convex conjugate of square loss $L(x,\xi) = \frac{1}{2}\|x-\xi\|^2$ w.r.t. its second argument.

Let $G(M, A)$ denote the objective in (4.4) for any fixed M, and let $g(M) = \max_A G(M, A)$ denote the optimal value of the objective when optimized over A. The objective in (4.2) for a fixed K could thus be simply reformulated as a minimization of this dual-derived objective $g(M)$. It can be seen that $g(M)$ is a convex function w.r.t. M since it is the maximum of linear functions of M. The key caveat however is the combinatorial structure on M since it has the form $M = ZZ^T$, $Z \in \{0,1\}^{N\times K}$. We address this caveat by introducing the following atomic norm

$$\|M\|_{\mathcal{S}} := \min_{c\geq 0} \sum_{a\in\mathcal{S}}^{K} c_a \ s.t.\ M=\sum_{a\in\mathcal{S}} c_a a. \tag{4.5}$$

with $\mathcal{S} := \{zz^T | z \in \{0,1\}^N\}$. Note $\|M\|_{\mathcal{S}} = \sum_{a\in\mathcal{S}} c_a = K$ when c_a in (4.24) are constrained at integer value $\{0,1\}$, and it serves a convex approximation to K similar to the ℓ_1-norm used in *Lasso* for the approximation of cardinality. This results in the following *Latent Feature Lasso* estimator

$$\min_M \ \{g(M)+\lambda\|M\|_S\}. \tag{4.6}$$

4.1.3 Algorithm

The estimator (4.6) seems intractable at first sight in part since the atomic norm involves a set $\mathcal{S}$ of 2^N atoms. In this section, we study a variant of approximate greedy coordinate descent

Algorithm 12 A Greedy Algorithm for Latent Feature Lasso

[0:] $\mathcal{A} = \emptyset$, $c = 0$.
for $t = 1...T$ **do**
[1:] Find a greedy atom zz^T by solving (4.28).
[2:] Add zz^T to an active set $\mathcal{A}$.
[3:] Minimize (4.7) w.r.t. coordinates in $\mathcal{A}$ via updates (4.29).
[4:] Eliminate $\{z_j z_j^T | c_j = 0\}$ from $\mathcal{A}$.
end for.

method for tractably solving problem (4.6). We begin by rewriting the optimization problem (4.6) as an ℓ_1-regularized problem with $\bar{K} = 2^N - 1$ coordinates, by expanding the matrix M in terms of the $\bar{K}$ atoms underlying the atomic norm $\|.\|_{\mathcal{S}}$:

$$\min_{c \in \mathbb{R}_+^{\bar{K}}} \underbrace{\left\{ \underbrace{g\left(\sum_{j=1}^{\bar{K}} c_j z_j z_j^T\right)}_{f(c)} + \lambda \|c\|_1 \right\}}_{F(c)} \tag{4.7}$$

where $\{z_j\}_{j=1}^{\bar{K}}$ enumerates all possible $\{0,1\}^N$ patterns except the 0 vector. Our overall algorithm is depicted in Algorithm 13. In each iteration, it finds

$$\begin{aligned} j^* &:= arg\max_j \ -\nabla_j f(c) \\ &= arg\max_j \ \langle -\nabla g(M), z_j z_j^T \rangle \end{aligned} \tag{4.8}$$

approximately with a constant approximation ratio via a reduction to a MAX-CUT-like problem, which we will discuss later. An active set $\mathcal{A}$ is maintained to contain all atoms $z_j z_j^T$ with non-zero coefficients c_j and the atom returned by the greedy search (4.28). Then we minimize (4.7) over coordinates in $\mathcal{A}$ by a sequence of proximal updates:

$$c^{r+1} \leftarrow \left[c^r - \frac{\nabla f(c^r) + \lambda}{\gamma |\mathcal{A}|} \right]_+, \ r = 1...T_2 \tag{4.9}$$

where γ is the Lipschitz-continuous constant of the coordinate-wise gradient $\nabla_{c_j} f(c)$.

Computing cooordinate-wise gradients. By Danskin's Theorem, the gradient of function $f(c)$ takes the form

$$\nabla_{c_j} f(c) = z_j A^* A^{*T} z_j / (2N^2 \tau), \tag{4.10}$$

which in turn requires finding the maximizer A^* of (4.4).

Computing A^*. By taking advantage of the strong duality between (4.4) and (4.3), the maximizer A^* can be found by finding the minimizer W^* of

$$\min_W \frac{1}{2N}\|X - Z_{\mathcal{A}}W\|_F^2 + \sum_{k\in\mathcal{A}} \frac{\tau}{2c_k}\|W_{k,:}\|^2 \tag{4.11}$$

and computing $A^* = (X - Z_{\mathcal{A}}W^*)$, where $Z_{\mathcal{A}}$ denotes $N \times |\mathcal{A}|$ matrix of columns taking from the active atom basis $\{z_k\}_{k\in\mathcal{A}}$.

Computing W^*. There is a closed-form solution W^* to (4.11) of the form

$$W^* = (Z_{\mathcal{A}}^T Z_{\mathcal{A}} + N\tau^{-1}(c_{\mathcal{A}}))^{-1} Z_{\mathcal{A}}^T X. \tag{4.12}$$

An efficient way of computing (4.12) is to maintain $Z_{\mathcal{A}}^T Z_{\mathcal{A}}$ and $Z_{\mathcal{A}}^T X$ whenever the active set of atoms $\mathcal{A}$ changes. This has a cost of $O(NDK_{\mathcal{A}})$ for a bound $K_{\mathcal{A}}$ on the active size, which however is almost neglectable compared to the other costs when amortized over iterations. Then the evaluation of (4.12) would cost only $O(K_{\mathcal{A}}^3 + K_{\mathcal{A}}^2 D)$ for each evaluation of different c. Similarly the matrix computation of (4.10) can be made more efficient as $\nabla_c f(c) \propto$

$$((Z_{\mathcal{A}}^T X - Z_{\mathcal{A}}^T Z_{\mathcal{A}} W^*)(Z_{\mathcal{A}}^T X - Z_{\mathcal{A}}^T Z_{\mathcal{A}} W^*)^T)$$

can be computed in $O(K^2 D + K^3)$ via the maintenance of $Z_{\mathcal{A}}^T Z_{\mathcal{A}}$, $Z_{\mathcal{A}}^T X$.

The output of Algorithm 13 is the coefficient vector c, and with the resulting latent feature matrix $W(c)$ given by (4.12). Since the solution could contain many atoms of small weight c_k. In practice, we perform a rounding procedure that ranks atoms according to the score $\{c_k\|W_{k,:}\|^2\}_{k\in\mathcal{A}}$ and then pick top K atoms as the output Z^*, and solve a simple least-squares problem to obtain the corresponding W^*.

Greedy Atom Generation A key step to the greedy algorithm (Algorithm 13) is to find the direction (4.28) of steepest descent, which however is a *convex maximization* problem with *binary constraints* that in general cannot be exactly solved in polynomial time. Fortunately in this section, we show that (4.28) is equivalent to a MAX-CUT-like *Boolean Quadratic Maximization* problem that has efficient Semidefinite relaxation with constant approximation guarantee. Furthermore, the resulting Semidefinite Programming (SDP) problem is of special structure that allows iterative method of complexity linear to the matrix size [10, 102].

In particular, let C=$\nabla g(M)$=$A^* A^{*T}/(2\tau N)$ the maximization problem

$$\max_{z\in\{0,1\}^N} \langle C, zz^T\rangle \tag{4.13}$$

can be reduced to an optimization problem over variables taking values in $\{-1, 1\}$ via the transformation $y = 2z - 1$, which results in the problem

$$\max_{y\in\{-1,1\}^N} \frac{1}{4}\left(\langle C, yy^T\rangle + 2\langle C, \mathbf{1}y^T\rangle + \langle C, \mathbf{1}\mathbf{1}^T\rangle\right). \tag{4.14}$$

where $\mathbf{1}$ denotes N-dimensional vector of all 1s. By introducing a dummy variable y_0, (4.32) can be expressed as

$$\max_{(y_0;y)\in\{-1,1\}^{N+1}} \frac{1}{4} \begin{bmatrix} y_0 \\ y \end{bmatrix}^T \begin{bmatrix} \mathbf{1}^T C \mathbf{1} & \mathbf{1}^T C \\ C\mathbf{1} & C \end{bmatrix} \begin{bmatrix} y_0 \\ y \end{bmatrix}. \tag{4.15}$$

Note that one can ensure finding a solution with $y_0 = 1$ by flipping signs of the solution vector to (4.33), since this does not change the quadratic form objective value. Denote the quadratic form matrix in (4.33) be $\hat{C}$. Problem of form (4.33) is a MAXCUT-like Boolean Quadratic problem, for which there is SDP relaxation of the form

$$\begin{aligned} \max_{Y\in\mathbb{S}^N} \quad & \langle \hat{C}, Y \rangle \\ s.t. \quad & Y \succeq 0,\ diag(Y) = \mathbf{1} \end{aligned} \tag{4.16}$$

rounding from which guarantees a solution $\hat{y}$ to (4.33) satisfying

$$\overline{h} - h(\hat{y}) \leq \rho(\overline{h} - \underline{h}) \tag{4.17}$$

for $\rho = 2/5$ [73], where $h(y)$ denotes the objective function of (4.33) and $\overline{h}$, $\underline{h}$ denote the maximum, minimum value achievable by some $y \in \{-1,1\}^{N+1}$ respectively. Note this result holds for any symmetric matrix $\hat{C}$. Since our problem has a positive-semidefinite matrix $\hat{C}$, $\underline{h} = 0$ and thus

$$-\nabla_{\hat{j}} f(c) = h(\hat{y}) \geq \mu \overline{h} = \mu(-\nabla_{j^*} f(c)) \tag{4.18}$$

for $\mu = 1 - \rho = 3/5$, where $\hat{j}$ is coordinate selected by rounding from a solution of (4.34) and j^* is the exact maximizer of (4.28).

Finally, it is noteworthy that, although solving a general SDP is computationally expensive, SDP of the form (4.34) has been shown to allow much faster solver that has linear cost w.r.t. the matrix size $nnz(\hat{C})$ [10, 102]. In our implementation we adopt the method of [102] due to its strong empirical performance.

Convergence Analysis The aim of this section is to show the convergence of Algorithm 13 under the approximation of greedy atom generation. In particular, we show the multiplicative approximation error incurred in the step (4.28) only contributes an additive approximation error proportional to λ, as stated in the following theorem.

Theorem 14. *The greedy algorithm proposed (Algorithm 13) satisfies*

$$F(c^t) - F(c^*) \leq \frac{2\gamma \|c^*\|_1^2}{\mu^2} \frac{1}{t} + \underbrace{\frac{2(1-\mu)}{\mu} \lambda \|c^*\|_1}_{\Delta(\lambda)},$$

where c^* is any reference solution, $\mu = 3/5$ is the approximation ratio given by (4.18) *and γ is the Lipschitz-continuous constant of coordinate-wise gradient $\nabla_j f(c),\ \forall j \in [K]$.*

The theorem thus shows that the iterates converge sub-linearly to within statistical precision λ of any reference solution c^* scaled in main by its ℓ_1 norm $\|c^*\|_1$. In the following theorem, we show that, with the additional assumption that $F(c)$ is strongly convex over a restricted support set $\mathcal{A}^*$, one can get a bound in terms of the ℓ_0**-norm** of a reference solution c^* with support A^*.

Theorem 15. *Let $\mathcal{A}^* \in [\bar{K}]$ be a support set and $c^* := arg\min_{c:supp(c)=\mathcal{A}^*} F(c^*)$. Suppose $F(c)$ is strongly convex on $\mathcal{A}^*$ with parameter β. The solution given by Algorithm 13 satisfies*

$$F(c^T) - F(c^*) \leq \frac{4\gamma\|c^*\|_0}{\beta\mu^2}\left(\frac{1}{T}\right) + \frac{2(1-\mu)\lambda}{\mu}\sqrt{\frac{2\|c^*\|_0}{\beta}}.$$

Let $\bar{K} = 2^N$ be the size of the atomic set. Any target latent structure Z^*W^* can be expressed as $\mathbf{Z}D(c^*)\tilde{W}^*$ where $\mathbf{Z}$ is an $N \times \bar{K}$ dictionary matrix, $D(c^*)$ is a $\bar{K} \times \bar{K}$ diagonal matrix of diagonal elements $D_{kk} = \sqrt{c_k^*}$ with $c_k^* = 1$ for columns corresponding to Z^* and $c_k^* = 0$ for the others, and $\tilde{W}^*$ is W^* padded with 0 on rows in $\{k \mid c_k = 0\}$. Then since $\|c^*\|_1 = \|c^*\|_0 = K^*$, Theorem 15 shows that our algorithm has an iteration complexity of $O(K/\epsilon)$ to achieve ϵ error, with an additional error term proportional to $\lambda\sqrt{K}$ due to the approximation made in (4.18).

Risk Analysis In this section, we investigate the performance of the output from Algorithm 13 in terms of the population risk $r(\cdot)$ defined in (4.1). Given coefficients c with support $\mathcal{A}$ obtained from algorithm (13) for T iterations, we construct the weight matrix by $\hat{W} = (\sqrt{c_\mathcal{A}})W^*$ with $W^*(c_\mathcal{A}) = \frac{1}{N}Z_\mathcal{A}^T A^*$, where A^* is the maximizer of (4.4) as a function of c. It can be seen that $\hat{W}$ satisfies

$$F(c) = \frac{1}{2N}\|X - Z_\mathcal{A}\hat{W}\|_F^2 + \frac{\tau}{2}\|\hat{W}\|_F^2 + \lambda\|c_\mathcal{A}\|_1. \tag{4.19}$$

The following theorem gives a risk bound for $\hat{W}$. Without loss of generality, we assume x is bounded and scaled such that $\|x\| \leq 1$.

Theorem 16. *Let $\hat{W} = (\sqrt{c_\mathcal{A}})W^*(c_\mathcal{A})$ be the weight matrix obtained from T iterations of Algorithm 13, and $\bar{W}$ be the minimizer of the population risk* (4.1) *with K components and $\|\bar{W}\|_F \leq R$. We then have the following bound on population risk: $r(\hat{W}) \leq r(\bar{W}) + \epsilon$ with probability $1 - \rho$ for*

$$T \geq \frac{4\gamma}{\mu^2\beta}(\frac{K}{\epsilon}) \;\; \textit{and} \;\; N = \Omega(\frac{DK}{\epsilon^3}\log(\frac{RK}{\epsilon\rho})),$$

with λ, τ chosen appropriately as functions of N.

Note the output of Algorithm 13 has number of components $\hat{K}$ bounded by number of iterations T. Therefore, Theorem (16) gives us a trade-off between risk and sparsity—one can guarantee to achieve ϵ-suboptimal risk compared to the optimal solution of size K, via $O(K/\epsilon)$ components and $\tilde{O}(DK/\epsilon^3)$ samples. Notice the result (16) is obtained without any distributional assumption on $p(x)$ and $p(z)$ except that of boundedness. Comparatively, the theoretical result obtained from Spectral Method [97] requires the knowledge/assumption of the distribution $p(x|z)$, $p(z)$, which is sensitive to model mis-specification in practice.

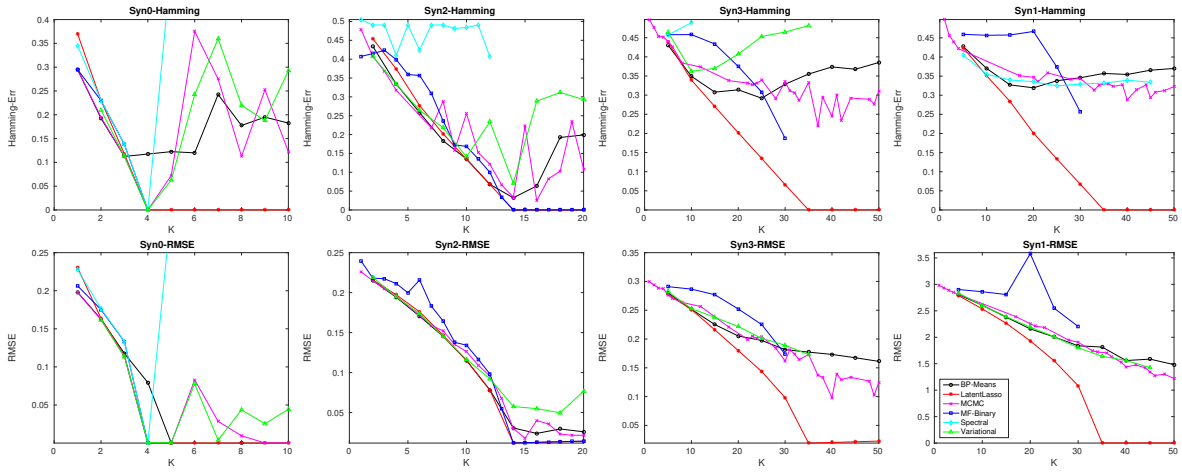

Figure 4.1: From left to right, each column are results for Syn0 (K=4), Syn2 (K=14), Syn3 (K=35) and Syn1 (K=35) respectively. The first row shows the Hamming loss between the ground-truth binary assignment matrix Z^* and the recovered ones $\hat{Z}$. The second row shows RMSE between $\Theta^* = Z^*W^*$ and the estimated $\hat{\Theta} = \hat{Z}\hat{W}$.

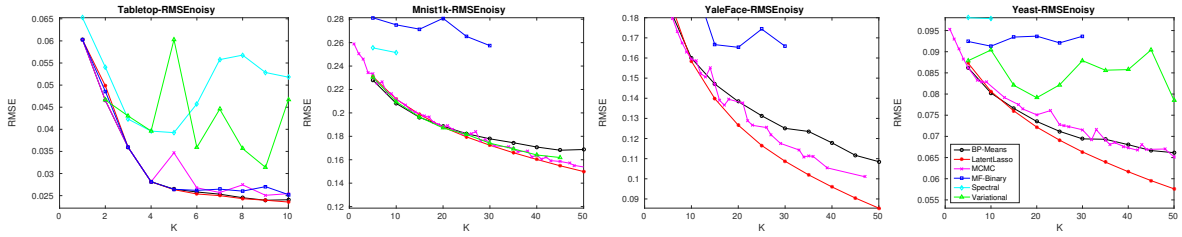

Figure 4.2: From left to right are results for Tabletop, Mnist1k, YaleFace and Yeast, where Spectral Method does not appear in the plots for YaleFace and Yeast due to a much higher RMSE, and Variational method reports a runtime error when running on the YaleFace data set.

4.1.4 Experiments

Table 4.1: Data statistics.

Dataset	N	D	K	σ	$nnz(W^*_{k,:})$
Syn0	100	196	4	0	≤ 8
Syn1	1000	1000	35	0.01	1000
Syn2	1000	900	14	0.1	49
Syn3	1000	900	35	0.1	36
Tabletop	100	8560	4	n/a	n/a
Mnist1k	1000	777	n/a	n/a	n/a
YaleFace	165	2842	n/a	n/a	n/a
Yeast	1500	104	n/a	n/a	n/a

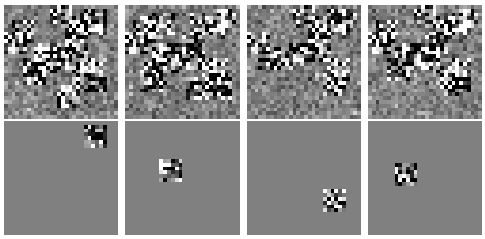

Figure 4.3: Synthetic data (i.e. Syn1, Syn2, Syn3). The first row shows observations $X_{i,:}$, and the second row shows latent features $W_{k,:}$.

In this section, we compare our proposed method with other state-of-the-art approaches on

both synthetic and real data sets. The dataset statistics are listed in Table 4.1. For the synthetic data experiments, we used a benchmark simulated dataset *Syn0* that was also used in [11, 97]. But since this has only a small number of latent features ($K = 4$), to make the task more challenging, we created additional synthetic datasets (which we denote Syn1, Syn2, Syn3) with more latent features. Figure 4.3 shows example of our synthetic data, where we reshape dimension D into an image and pick a contiguous region. Each pixel $W(k, j)$ in the region is set as $N(0, \sigma^2)$, while pixels not in the region are set to 0. In the examples of Figure 4.3, the region has size $nnz(W(k, :))$=36. Note the problem becomes harder when the region size $nnz(W(k, :))$, number of features K, or noise level σ becomes larger. For real data, we use a benchmark *Tabletop* data set constructed by [32], where there is a ground-truth number of features $K = 4$ for the 4 objects on the table. We also take two standard multilabel (multiclass) classification data sets *Yeast* and *Mnist1k* from the LIBSVM repository [1], and one Face data set *YaleFace* from the Yale Face database [2].

Given the estimated factorization (Z, W), we use the following 3 evaluation metrics to compare different algorithms:

- Hamming-Error: $\min_{\mathcal{S}:|\mathcal{S}|=K} \frac{\|Z_{:,\mathcal{S}} - Z^*\|_F^2}{NK}$.
- RMSE: $\frac{\|Z^*W^* - ZW\|_F}{\sqrt{ND}}$.
- RMSEnoisy: $\frac{\|X - ZW\|_F}{\sqrt{ND}}$.

where the first two can only be applied when the ground truth Z^* are W^* are given. For real data, we can only evaluate the noisy version of RMSE, which can be interpreted as trying to find a best approximation to the observation X via a factorization with binary components.

The methods in comparison are listed as follows: **(a) MCMC:** An accelerated version of the Collapsed Gibbs sampler for the Indian Buffet Process (IBP) model [24]. We adopted the implementation published by [3]. We ran it with 25 random restarts and recorded the best results for each K. **(b) Variational:** A Variational approximate inference method for IBP proposed in [25]. We used implementation published by the author [4]. **(c) MF-Binary:** A Matrix Factorization with the Binary Components model [88], which has recovery guarantees in the noiseless case but has a $O(K2^K)$ complexity and thus cannot scale to $K > 30$ on our machine. We use the implementation published by the author [5]. **(d) BP-Means:** A local search method that optimizes a MAD-Bayes Latent Feature objective function [11]. We used code provided by the author [6]. We ran it with 100 random restarts and recorded the best result. **(e) Spectral:** Spectral Method for IBP Linear Gaussian model proposed in [97]. We used code from the author. The implementation has a memory requirement that restricts its use to $K < 14$. **(f) LatentLasso:** The proposed Latent Feature Lasso method (Algorithm 13).

The results are shown in Figure 4.1 and 4.2. On synthetic data, we observe that, when the number of features K is small (e.g. Syn0), most of methods perform reasonably well. However, when the number of features becomes slightly larger (i.e. $K = 35$ in Syn1, Syn3), most of

[1] https://www.csie.ntu.edu.tw/ cjlin/libsvmtools/datasets/

[2] http://vision.ucsd.edu/content/yale-face-database

[3] `https://github.com/davidandrzej/PyIBP`

[4] `http://mloss.org/software/view/185/`

[5] `https://sites.google.com/site/slawskimartin/code`

[6] `https://github.com/tbroderick/bp-means`

algorithms lose their ability of recovering the hidden structure, and when they fail to do so, they can hardly find a good approximation to $\Theta^* = Z^*W^*$ even using a much larger number of components up to 50. We found the proposed *LatentLasso* method turns out to be the only method that can still recover the desired hidden structure on the Syn1 and Syn3 data sets, which gives 0 RMSE and Hamming Error. On Syn2 ($K = 14$) data set, *MF-Binary* and *LatentLasso* are the only two methods that achieve 0 RMSE and Hamming-Error. However, MF-Binary has a complexity growing exponential with K, which results in its failure on Syn1 and Syn3 due to a running time more than one day when $K > 30$. The proposed LatentLasso algorithm actually runs significantly faster than other methods in our experiments. For example, on the Syn1 dataset (N=1000, D=1000, K=35), the runtime of LatentLasso is 398s, while MCMC, Variational, MF-Binary and BP-Means all take more than 10000s to obtain their best results reported in the Figures.

4.2 Mixed Regression

In this section, we extend the convex atomic-norm-regularized estimator to the problem of *Mixed Regression (MR)*. Mixed Regression considers the estimation of K functions from a collection of input-output samples, where for each sample, the output is generated by one of the K regression functions. When fitting linear functions in a noiseless setting, this is equivalent to solving K linear systems, while at the same time, identifying which system each equation belongs to. The MR formulation can be employed as an approach to decompose a complicated function into K simpler ones, by splitting the observations into K classes. Variants of regression families such as *piecewise-linear regression* can be viewed as special cases of MR.

However, the MR problem is NP-hard in general [123] due to the simultaneous fitting of the discrete class labels as well as the regression functions. Standard approaches to the mixture problem employ local search methods such as Expectation Maximization (EM) [108] and Variational Bayes [9] that are prone to spurious local optima. There have thus been several lines of recent work studying estimation of mixed regression models with strong statistical guarantees under additional statistical assumptions. For the special case of linear function with K=2 components, [16] propose a convex nuclear norm minimization formulation that is guaranteed to estimate the two functions with minimax-optimal rates when given a sub-Gaussian design matrix. With the additional conditions of zero noise and isotropic Gaussian inputs, [123] propose an initialization for the EM algorithm to guarantee exact recovery of the true parameters. However, in addition to the stringent statistical assumptions, these methods and results are specialized to the case of two components, and seem non-trivial to generalize.

For problems with more than two components, most of the existing approaches [12, 84, 124, 127] rely on the *Tensor Methods*. In particular, for a D-dimensional linear MR problem, [12] propose a convex optimization formulation using a third-order tensor, which results in a computational cost of $O(ND^{12})$ and a sample complexity of $O(D^6/\epsilon^2)$, limiting its application to problems of small dimension. The *Tensor Decomposition* approach proposed in [84] has a sample complexity of only $O(D^3K^4/\epsilon^2)$ and is computationally efficient. However, it requires the knowledge of the input probability distribution in order to derive the *score function* used in their algorithm, which might not be available, and estimating the density over the D-dimensional in-

put variables could be an even harder problem than MR itself. Other recent work [124, 127] show that in the noiseless setting with isotropic Gaussian inputs, an Alternating Minimization algorithm initialized with the Tensor Method leads to exact recovery of the true parameters. These latter methods have sample complexities linear in D, but with $O(K^K)$, $O(K^{10})$ dependencies in K respectively. Finally, [38] observed that, under the assumption of well-separated data, one can use a guaranteed clustering algorithm to find the mixture assignment of each observation, and thus solves the MR problem as a by-product. However, the data distribution considered in MR, such as those assumed in [12, 84, 124, 127], are usually not well-separated (see our Figure 4.6 as an example).

In this work, we address a generalized version of Mixed Regression where the output can be an additive combination of several mixture components. Our approach follows the general meta-approach emerging in the recent years of addressing latent-variable model estimation from the perspective of high-dimensional sparse estimation [116, 120, 121]. We propose a novel convex estimator *MixLasso* for the mixed regression problem, which enforces the mixture structure through minimizing a carefully constructed atomic norm that acts as a surrogate function for the number of mixture components. We then propose a greedy algorithm that generates a steepest-descent component at each iteration through solving a sub-problem similar to MAX-CUT. Our analysis of the algorithm gives a risk bound that trades off prediction accuracy and model sparsity, with a sample complexity that is linear in both D and K, and without imposing any stringent assumptions on, or assuming knowledge of, the input/output distribution beyond that of boundedness, and even allowing for model mis-specification. This makes our *MixLasso* algorithm a theoretically sound method for a wide range of practical settings. Moreover, we also show how our proposed method can be easily extended to the nonlinear regression setting, to regression functions lying in a Reproducing Kernel Hilbert Space (RKHS). Our experiments with both generalized MR and standard MR show that the proposed method finds high-quality solutions in a wider range of settings when compared to existing approaches.

4.2.1 Problem Setup

In Generalized Mixed Regression, the response $y \in \mathbb{R}$, given covariates $\boldsymbol{x} \in \mathcal{X}$, is specified as:

$$y = \sum_{k=1}^{K} z_k f_k(\boldsymbol{x}) + \omega \tag{4.20}$$

where $z_k \in \{0, 1\}$, $k = 1, \ldots, K$ is a latent binary vector indicating the presence or absence of each component, and $f_k(\boldsymbol{x}_i) : \mathbb{R}^D \to \mathbb{R}$ is the regression function of k-th component. The standard mixed regression is a special case of (4.20) with additional constraint $\|\boldsymbol{z}\|_0 = 1$. Here $\omega \in \mathbb{R}$ is a noise term with both bias and variance. In other words, we consider the very general setting where we allow for model mis-specification, and in general $\mathcal{E}[\omega|\boldsymbol{x}, \boldsymbol{z}] \neq 0$. This makes our problem setting in (4.20) very practically plausible, especially when the regression functions $\{f_k(\boldsymbol{x})\}_{k=1}^K$ lie in some restricted family such as linear functions.

Our goal is to find $\mathcal{F} := \{f_k(\boldsymbol{x})\}_{k=1}^K$ minimizing the risk

$$r(\mathcal{F}) := \mathcal{E}\left[\min_{\boldsymbol{z} \in \{0,1\}^K} \frac{1}{2}(y - \sum_{k=1}^{K} z_k f_k(\boldsymbol{x}))^2\right], \tag{4.21}$$

while keeping the number of components K as small as possible. This yields a trade-off between $r(\mathcal{F})$ and K. While one can always have a small risk with $K \to \infty$, we would like to find the smallest K that achieves such risk.

4.2.2 MixLasso: A Convex Estimator

In the following, we will first focus on the linear case $f_k(\boldsymbol{x}) := \langle \boldsymbol{w}_k, \boldsymbol{x} \rangle$ and consider extension to nonlinear functions. Given a collection of i.i.d. samples $\{(\boldsymbol{x}_i, y_i)\}_{i=1}^N$, the ℓ_2-regularized Empirical Risk Minimization (ERM) problem for our task (4.21) is

$$\min_{W \in \mathbb{R}^{K \times D}, \boldsymbol{z}_i \in \{0,1\}^K} \frac{1}{2N} \sum_{i=1}^{N} (y_i - \boldsymbol{z}_i^{\mathcal{T}} W \boldsymbol{x}_i)^2 + \frac{\tau}{2} \|W\|_F^2. \tag{4.22}$$

(C.7) is a hard optimization problem in general due to the simultaneous minimization w.r.t. parameters W and binary hidden variables $\{\boldsymbol{z}_i\}_{i=1}^N$ [123]. However, given hidden variables, the problem is convex w.r.t. W, and thus, from the duality theory (C.7) is equivalent to

$$\min_{Z \in \{0,1\}^{N \times K}} \max_{\boldsymbol{\alpha} \in \mathbb{R}^N} \frac{-1}{N} \sum_{i=1}^{N} L^*(y_i, -\alpha_i) - \frac{1}{2N^2 \tau} tr(\mathcal{D}(\boldsymbol{\alpha}) X X^{\mathcal{T}} \mathcal{D}(\boldsymbol{\alpha}) Z Z^{\mathcal{T}}) \tag{4.23}$$

where $Z := (\boldsymbol{z}_i)_{i=1}^N$, $\mathcal{D}(\boldsymbol{\alpha})$ is a diagonal matrix formed by vector $\boldsymbol{\alpha}$, and $L^*(y, \alpha) = y^{\mathcal{T}} \alpha + \frac{1}{2} \|\alpha\|^2$ is the convex conjugate of square loss $L(y, \xi) = \frac{1}{2}(y - \xi)^2$. The maximizer $\boldsymbol{\alpha}^*$ of (4.23) and minimizer W^* of (C.7) are related by $W^* = \frac{1}{N\tau} \sum_{i=1}^N \alpha_i^* (\boldsymbol{z}_i \boldsymbol{x}_i^{\mathcal{T}}) = \frac{1}{N\tau} Z^{\mathcal{T}} D(\boldsymbol{\alpha}^*) X$.

A key observation for our formulation is that, although (4.23) is non-convex w.r.t. Z, it is a convex function of $M := ZZ^{\mathcal{T}}$ (since it is a maximum over linear functions of M). Therefore, the intractability of (4.23) only lies in the combinatorial constraint $M = ZZ^{\mathcal{T}}$ for some $Z \in \{0,1\}^{N \times K}$. To relax such constraint, we introduce an *atomic norm* [13] of the form

$$\|M\|_{\mathcal{S}} := \min_{c \geq 0} \sum_{a \in \mathcal{S}} c_a \;\; s.t. \;\; M = \sum_{a \in \mathcal{S}} c_a a. \tag{4.24}$$

where $\mathcal{S} := \{\boldsymbol{z}\boldsymbol{z}^{\mathcal{T}} | \boldsymbol{z} \in \{0,1\}^N\}$. Note if c_a takes integer values $\{0,1\}$, $M = \sum_{a \in \mathcal{S}} c_a a = ZZ^{\mathcal{T}}$ for some $Z \in \{0,1\}^{N \times K}$ and $\|M\|_{\mathcal{S}} = K$. When c_a is allowed to be any nonnegative number, (4.24) serves as a convex approximation to the number of components K in a sense similar to ℓ_1-norm as a convex approximation for the number of non-zero elements in *Lasso* [94]. Then the ***MixLasso*** estimator solves

$$\min_{M \in \mathbb{R}_+^{N \times N}} \; g(M) + \lambda \|M\|_S \tag{4.25}$$

where $g(M)$ is defined as

$$g(M) = \max_{\boldsymbol{\alpha} \in \mathbb{R}^N} \; -\frac{1}{2N^2 \tau} tr(\mathcal{D}(\boldsymbol{\alpha}) X X^{\mathcal{T}} \mathcal{D}(\boldsymbol{\alpha}) M) - \frac{1}{N} \sum_{i=1}^{N} L^*(y_i, -\alpha_i) \tag{4.26}$$

Algorithm 13 A Greedy Algorithm for MixLasso (4.25)

Initialize $\mathcal{A} = \emptyset$, $\boldsymbol{c} = 0$.
for $t = 1...T$ **do**
1. Find a greedy component $\boldsymbol{z}\boldsymbol{z}^{\mathcal{T}}$ by solving (4.28).
2. Add $\boldsymbol{z}\boldsymbol{z}^{\mathcal{T}}$ to the active set $\mathcal{A}$.
3. Minimize (4.27) w.r.t. coordinates $\boldsymbol{c}_{\mathcal{A}}$ in the active set $\mathcal{A}$ through updates (4.29).
4. Eliminate $\{\boldsymbol{z}^k\boldsymbol{z}^{k\mathcal{T}} | c_k = 0\}$ from $\mathcal{A}$.

end for.

4.2.3 Algorithm

The convex formulation (4.25) is still a challenging optimization problem since it involves an atomic norm defined over $\bar{K} := 2^N$ atoms. An equivalent formulation expresses (4.25) as the minimizatioin of

$$F(\boldsymbol{c}) := g\left(\sum_{k=1}^{\bar{K}} c_k \boldsymbol{z}^k \boldsymbol{z}^{k\mathcal{T}}\right) + \lambda \|\boldsymbol{c}\|_1 \tag{4.27}$$

w.r.t. $\boldsymbol{c} \in \mathbb{R}_+^{\bar{K}}$, where $\{\boldsymbol{z}^k\}_{k=1}^{\bar{K}}$ enumerates $\forall \boldsymbol{z} \in \{0,1\}^N$. We introduce a greedy algorithm (Algorithm 13) for MixLasso, which maintains a sparse set of active components and adds one more active component $\boldsymbol{z}^k\boldsymbol{z}^{kT}$ at each iteration corresponding to the steepest descent direction

$$\min_{\boldsymbol{z}\in\{0,1\}^N} \langle \nabla g(M), \boldsymbol{z}\boldsymbol{z}^{\mathcal{T}} \rangle = -\frac{1}{2N^2\tau} \max_{z\in\{0,1\}^N} \langle \mathcal{D}(\boldsymbol{\alpha}^*) X X^{\mathcal{T}} \mathcal{D}(\boldsymbol{\alpha}^*), \boldsymbol{z}\boldsymbol{z}^{\mathcal{T}} \rangle, \tag{4.28}$$

where $\boldsymbol{\alpha}^*$ is the maximizer in (4.26). (4.28) is equivalent to a *MAX-CUT* like problem that can be solved efficiently with a constant-ratio approximation guarantee. Then we minimize (4.27) w.r.t. coefficients corresponding to the active components through a sequence of proximal gradient updates:

$$c_k^{s+1} \leftarrow \left[c_k^s - \frac{1}{\gamma|\mathcal{A}|}(\boldsymbol{z}^{k\mathcal{T}} \nabla g(M^s) \boldsymbol{z}^k + \lambda)\right]_+ \tag{4.29}$$

for $k \in \mathcal{A}$, and $s = 1 \dots S$, where γ is the Lipschitz-continuous parameter of the coordinate-wise gradient $\boldsymbol{z}^{k\mathcal{T}} \nabla g(M) \boldsymbol{z}^k$. The evaluation of $\nabla g(M^s)$ involves finding the maximizer $\boldsymbol{\alpha}^*$, which can be obtained by solving the least-square problem:

$$W^* := \underset{W\in\mathbb{R}^{|\mathcal{A}|\times D}}{argmin} \frac{1}{2N} \sum_{i=1}^{N} (y_i - \boldsymbol{z}_i^{\mathcal{T}} W \boldsymbol{x}_i)^2 + \frac{\tau}{2} tr(W^{\mathcal{T}} \mathcal{D}^{-1}(\boldsymbol{c}_{\mathcal{A}}) W) \tag{4.30}$$

and compute $\alpha_i^* = (y_i - \boldsymbol{z}_i^{\mathcal{T}} W^* \boldsymbol{x}_i)$. Let E be the $N \times (|\mathcal{A}|D)$ design matrix of the least-square problem (4.30). By maintaining E, $E^{\mathcal{T}}E$ whenever the active set $\mathcal{A}$ changes, solving the least-square problem (4.30) costs $O(D^3|\mathcal{A}|^3)$ amortizedly.

Greedy Generation of Components Problem (4.28) for finding the steepest descent direction is a convex maximization problem with binary-valued variables and is hard in general. However,

we show that it is equivalent to a *Boolean Quadratic Maximization* problem similar to *MAX-CUT*, where constant-ratio approximate algorithm exists through a Semidefinite Relaxation [73]. Furthermore, the Semidefinie Relaxation of this type has scalable solver that requires only complexity linear to the coefficient matrix [10, 102].

Let $C = \mathcal{D}(\boldsymbol{\alpha}^*)XX^{\mathcal{T}}\mathcal{D}(\boldsymbol{\alpha}^*)$. The greedy step (4.28) solves a problem of the form

$$\max_{\boldsymbol{z}\in\{0,1\}^N} \langle C, \boldsymbol{z}\boldsymbol{z}^{\mathcal{T}}\rangle, \tag{4.31}$$

which can be reduced to a problem of binary variables $\boldsymbol{v} \in \{-1,1\}^N$ via a transformation $\boldsymbol{v} = 2\boldsymbol{z} - 1$:

$$\max_{\boldsymbol{v}\in\{-1,1\}^N} \frac{1}{4}\left(\langle C, \boldsymbol{v}\boldsymbol{v}^{\mathcal{T}}\rangle + 2\langle C, \mathbf{1}\boldsymbol{v}^{\mathcal{T}}\rangle + \langle C, \mathbf{1}\mathbf{1}^{\mathcal{T}}\rangle\right). \tag{4.32}$$

where $\mathbf{1}$ denotes N-dimensional vector of all 1s. By introducing a dummy variable v_0, (4.32) is equivalent to

$$\max_{(v_0;\boldsymbol{v})\in\{-1,1\}^{N+1}} \frac{1}{4}\begin{bmatrix} v_0 \\ \boldsymbol{v} \end{bmatrix}^T \begin{bmatrix} \mathbf{1}^T C\mathbf{1} & \mathbf{1}^T C \\ C\mathbf{1} & C \end{bmatrix} \begin{bmatrix} v_0 \\ \boldsymbol{v} \end{bmatrix}. \tag{4.33}$$

Note one can always find a solution of $v_0 = 1$ by flipping signs of the solution since this does not change the objective value. Let the matrix in (4.33) be $\hat{C}$. Problem of form (4.33) is a Boolean Quadratic problem similar to MAX-CUT, for which there is Semidefinite relaxation of the form

$$\begin{aligned} \max_{V\in\mathbb{S}^N} \quad & \langle \hat{C}, V\rangle \\ s.t. \quad & V \succeq 0,\ diag(V) = \mathbf{1} \end{aligned} \tag{4.34}$$

and rounding from which guarantees a solution $\hat{\boldsymbol{v}}$ to (4.33) satisfying $\overline{h} - h(\hat{\boldsymbol{v}}) \leq \rho(\overline{h} - \underline{h})$ with $\rho = 2/5$ [73], where $h(\boldsymbol{v})$ denotes the objective function of (4.33) and $\overline{h}$, $\underline{h}$ denote the maximum and minimum of the objective in (4.33) respectively. Note this result holds for any symmetric matrix $\hat{C}$. Since our problem has a positive-semidefinite matrix $\hat{C}$, we have $\underline{h} = 0$ and therefore the component $\boldsymbol{z}^k$ found this way satisfies

$$\begin{aligned} & -\boldsymbol{z}^{k\mathcal{T}}\nabla g(M)\boldsymbol{z}^k = h(\hat{\boldsymbol{v}}) \\ & \geq \mu\overline{h} = \mu \max_{\boldsymbol{z}\in\{0,1\}^N} -\boldsymbol{z}^{\mathcal{T}}\nabla g(M)\boldsymbol{z} \end{aligned} \tag{4.35}$$

with $\mu = 1 - \rho = 3/5$. Semidefinite Programming of the form (4.34) allows specialized solver with iteration cost linear to the matrix size $nnz(\hat{C})$ [10, 102]. And it is worth mentioning that, since our matrix $\hat{C}$ has low-rank structure (4.28), our implementation of the SDP solver [102] can further reduce the complexity per iteration from $nnz(\hat{C})$ to $nnz(X)$.

Nonlinear Extension A simple way to consider a nonlinear version of the MixLasso estimator is to consider each component $f_k(\boldsymbol{x})$ lying in a Reproducing Kernel Hilbert Space (RKHS) $\mathcal{H}$ with respect to some Mercer kernel $\mathcal{K}(\cdot,\cdot)$. In this setting, given $\{\boldsymbol{z}_i\}_{i=1}^N$, the minimizer $\{f_k^*\}_{k=1}^K$ of

$$\min_{f_k\in\mathcal{H}} \frac{1}{2N}\sum_{i=1}^N \left(y_i - \sum_{k=1}^K z_{ik} f_k(\boldsymbol{x}_i)\right)^2 + \frac{\tau}{2}\sum_{k=1}^K \|f_k\|_{\mathcal{H}}^2 \tag{4.36}$$

satisfies the condition of the *Representer Theorem* that ensures an expression of the form

$$f_k^*(\boldsymbol{x}) = \sum_{i=1}^{N} \alpha_i z_{ik} \mathcal{K}(\boldsymbol{x}_i, \boldsymbol{x}),\ k \in [K], \tag{4.37}$$

for the minimizer, and results in a MixLasso estimator (4.25) with

$$g(M) := \max_{\boldsymbol{\alpha} \in \mathbb{R}^N} -\frac{1}{2N^2\tau} tr(\mathcal{D}(\boldsymbol{\alpha}) Q \mathcal{D}(\boldsymbol{\alpha}) M) - \frac{1}{N} \sum_{i=1}^{N} L^*(y_i, -\alpha_i) \tag{4.38}$$

where $Q : N \times N$ is the kernel matrix with $Q_{ij} = \mathcal{K}(\boldsymbol{x}_i, \boldsymbol{x}_j)$. Then Algorithm 13 can be applied with the only difference on the evaluation of gradient $\nabla g(M)$, which requires finding the maximizer $\boldsymbol{\alpha}^*$ of (4.38) by solving the following linear system:

$$(\frac{1}{N\tau} Q \circ M + I)\boldsymbol{\alpha} = \boldsymbol{y}. \tag{4.39}$$

where $\circ$ denotes the elementwise product.

Rounding Procedure for Generalized & Standard Mixed Rregression While the atomic-norm regularization $\lambda \|M\|_{\mathcal{S}}$ is a good convex relaxation of the number of components, the number of non-zero components getting from estimator (4.25) cannot be precisely specified apriori by the hyper-parameter λ directly. In practice, it is often useful to obtain a solution $\boldsymbol{c}$ with exactly $\|\boldsymbol{c}\|_0 = K$ non-zeros. This can be achieved by setting the K coefficients of largest magnitude to 1 and all the other coefficients to 0. This results in a $N \times K$ matrix of hidden assignments $\hat{Z}$ as the output of Algorithm 13. Then, starting from $\hat{Z}$, we can perform a number of alternating minimization steps between model parameters W (or $\{f_k\}_{k=1}^K$ in general) and hidden assignments $\{\boldsymbol{z}_i\}_{i=1}^N$ until convergence, as in a standard EM algorithm (with MAP hard assignment on $\boldsymbol{z}_i$).

While we have proposed a solution of the generalized version (4.20), in some applications, it might be of interest to solve the special case of standard mixed regression, where each observation belongs to exactly one mixture component. One approach to convert a generalized mixture solution with K components to a standard mixture of J components is to find the most frequent J patterns $\boldsymbol{\mathfrak{z}}_1, \boldsymbol{\mathfrak{z}}_2, ..., \boldsymbol{\mathfrak{z}}_J$ from the estimated hidden assignments $\{\hat{\boldsymbol{z}}_i\}_{i=1}^N$, and then force each observation to choose their hidden assignments $\{\boldsymbol{z}_i\}_{i=1}^N$ from the set $\{\boldsymbol{\mathfrak{z}}_j\}_{j=1}^J$ instead of arbitrary 0-1 patterns $\{0,1\}^K$. This results in J functions $\{\mathfrak{f}_j\}_{j=1}^J$ of the form

$$\mathfrak{f}_j(\boldsymbol{x}) = \sum_{k=1}^{K} \boldsymbol{\mathfrak{z}}_{jk} f_k(\boldsymbol{x}),\ j \in [J],$$

being actually used in the training observation, and thus gives a valid model $\{\mathfrak{f}_j\}_{j=1}^J$ of standard mixed regression with J components. Then as noted previously, one can further refine this rounded solution through EM iterates of standard mixed regression, initialized with component functions $\{\mathfrak{f}_j\}_{j=1}^J$.

Analysis We assume y and $\boldsymbol{x}$ are bounded such that $|y| \leq R_y$, $\|\boldsymbol{x}\|_2 \leq R_x$. And without loss of generality, we assume the data are scaled such that $R_y = R_x = 1$. Then the following theorem guarantees the rate of convergence for Algorithm 13 up to a certain precision determined by the approximation ratio given in (4.35).

Theorem 17. *Let $F(\boldsymbol{c})$ be the objective* (4.27)*. The greedy algorithm (Algorithm 13) satisfies*

$$F(\boldsymbol{c}^T) - F(\boldsymbol{c}^*) \leq \frac{2\gamma\|\boldsymbol{c}^*\|_1^2}{\mu^2}\left(\frac{1}{T}\right). \tag{4.40}$$

for any iterate T satisfying $F(\boldsymbol{c}^T) - F(\boldsymbol{c}^*) \geq \frac{2(1-\mu)}{\mu}\lambda\|\boldsymbol{c}^*\|_1$, where $\boldsymbol{c}^*$ is any reference solution, $\mu = 3/5$ is the approximation ratio given by (4.35) *and γ is the Lipschitz-continuous constant of the coordinate-wise gradient $\boldsymbol{z}^{k\mathcal{T}}\nabla g(M)\boldsymbol{z}^k,\ \forall k \in [\bar{K}]$.*

Then the following lemma shows that, with the additional assumption that $F(\boldsymbol{c})$ is strongly convex over a restricted support set $\mathcal{A}^*$, one can get a bound in terms of the ℓ_0-norm of the reference solution.

Lemma 1. *Let $\mathcal{A}^* \in [\bar{K}]$ be a support set and $\boldsymbol{c}^* := arg\min_{\boldsymbol{c}:supp(\boldsymbol{c})=\mathcal{A}^*} F(\boldsymbol{c}^*)$. Suppose $F(\boldsymbol{c})$ is strongly convex on $\mathcal{A}^*$ with parameter β. We have*

$$\|\boldsymbol{c}^*\|_1 \leq \sqrt{\frac{2\|\boldsymbol{c}^*\|_0(F(0) - F(\boldsymbol{c}^*))}{\beta}}. \tag{4.41}$$

Since $F(0) - F(\boldsymbol{c}^*) \leq \frac{1}{2N}\sum_{i=1}^N y_i^2 \leq 1$, from (4.40) and (C.5), we have

$$F(\boldsymbol{c}^T) - F(\boldsymbol{c}^*) \leq \frac{4\gamma\|\boldsymbol{c}^*\|_0}{\beta\mu^2}\left(\frac{1}{T}\right) + \frac{2(1-\mu)\lambda}{\mu}\sqrt{\frac{2\|\boldsymbol{c}^*\|_0}{\beta}}. \tag{4.42}$$

for any $\boldsymbol{c}^* := arg\min_{\boldsymbol{c}:supp(\boldsymbol{c})=\mathcal{A}^*} F(\boldsymbol{c})$.

Then we investigate the performance of output from Algorithm 13 in terms of the risk (4.21). Given a coefficients $\boldsymbol{c}$ with support $\mathcal{A}$, we can construct the weight matrix by $\hat{W}(\boldsymbol{c}) = \mathcal{D}(\sqrt{\boldsymbol{c}_{\mathcal{A}}})W$ with $W = Z_{\mathcal{A}}^{\mathcal{T}}\mathcal{D}(\boldsymbol{\alpha}^*)X$, where $Z_{\mathcal{A}} = (\boldsymbol{z}^k)_{k\in\mathcal{A}}$ and $\boldsymbol{\alpha}^*$ is the maximizer in (4.26) as a function of $\boldsymbol{c}$. From the duality between (C.7) and (4.23), $\hat{W}$ satisfies

$$F(\boldsymbol{c}_{\mathcal{A}}) = \frac{1}{2N}\sum_{i=1}^N (y_i - \boldsymbol{z}_i^{\mathcal{T}}\hat{W}\boldsymbol{x}_i)^2 + \frac{\tau}{2}\|W\|_F^2 + \lambda\|\boldsymbol{c}_{\mathcal{A}}\|_1. \tag{4.43}$$

The following theorem gives a risk bound for the output weight matrix $\hat{W}(\boldsymbol{c})$ obtained from Algorithm 13.

Theorem 18. *Let $\mathcal{A}$, $\hat{\boldsymbol{c}}$, $\hat{W}$ be the set of active components, coefficients and corresponding weight matrix obtained from T iterations of Algorithm 13, and $\bar{W}$ be the minimizer of the population risk* (4.21) *with K components and $\|\bar{W}\|_F \leq R$. We have $r(\hat{W}) \leq r(\bar{W}) + \epsilon$ with probability $1 - \rho$ for*

$$T \geq \frac{4\gamma}{\mu^2\beta}(\frac{K}{\epsilon}) \ \text{ and } \ N = \Omega(\frac{DK}{\epsilon^3}\log(\frac{RK}{\epsilon\rho}))$$

with λ, τ chosen appropriately as functions of N.

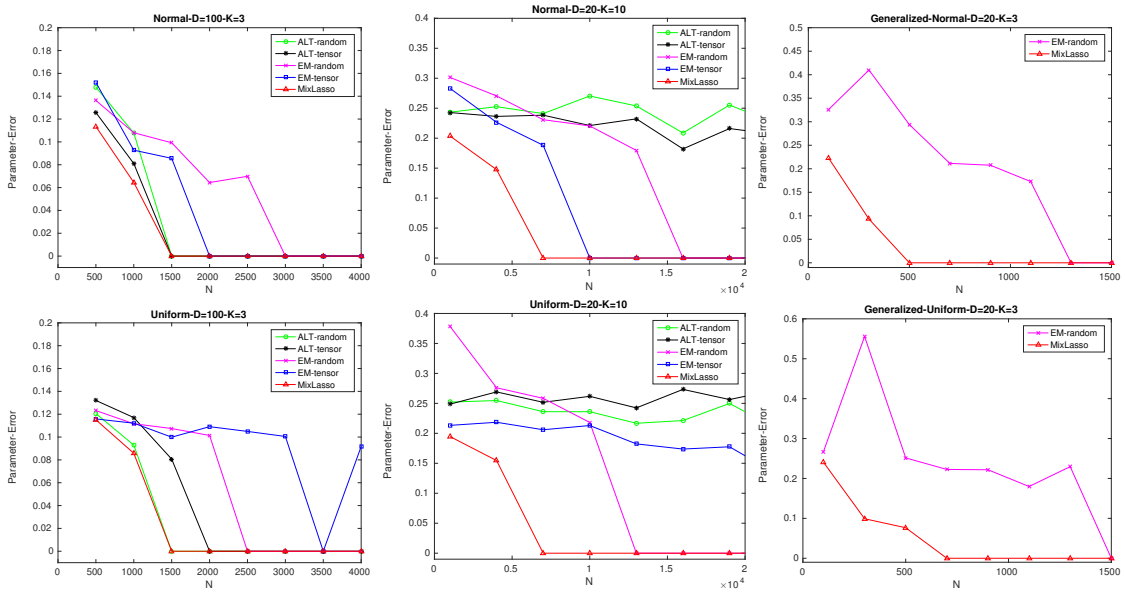

Figure 4.4: Results for Noiseless Mixture of Linear Regression with $N(0, I)$ input distribution (Top) and $U(-1, 1)$ input distribution (Bottom), where (Left) D=100, K=3, (Middle) D=20, K=10, and (Right) Generalized Mixture of Regression with D=20, K=3.

Note the output of Algorithm 13 has number of components $\hat{K} \leq T$. Therefore, Theorem 16 gives a trade-off between the suboptimality of risk $r(\hat{W}) - r(\bar{W}) \leq \epsilon$ and number of components $\hat{K} = O(K/\epsilon)$. Note the result of Theorem (16) is obtained without distributional assumption on the input/output (except boundedness), so it is in general not possible to guarantee convergence to an optimal risk with exactly K components, since finding such optimal solution is NP-hard even measured by the empirical risk [123]. It remains open if one can give a tighter result for the estimator (4.25) that achieves ϵ-suboptimal risk with number of components being a constant multiple of K, or derive a bound on the parameter estimation error, possibly with additional assumptions on the observations.

4.2.4 Experiments

In this section, we compare the proposed *MixLasso* method with other state-of-the-art approaches listed as follows:

- **EM-Random**: A standard EM algorithm that alternates between minimizing $\{\boldsymbol{z}_i\}_{i=1}^N$ and $\{f_k(\boldsymbol{x})\}_{k=1}^K$ until convergence, with random initialized $W \sim N(0, I)$ in the linear case and random initialized $Z \sim Multinoulli(1/K)$ in the nonlinear case. Each point in the figures is the best result out of 100 random trials.
- **EM-Tensor**: The EM algorithm initialized with Tensor Method proposed in [124]. The formula of Tensor Method is derived assuming $\boldsymbol{x}_i \sim N(0, I)$. We adopt implementation provided by the author of [127].
- **ALT-Random**: An Alternating Minimization algorithm proposed in [127] with the same initialization strategy and number of trails as EM-Random.

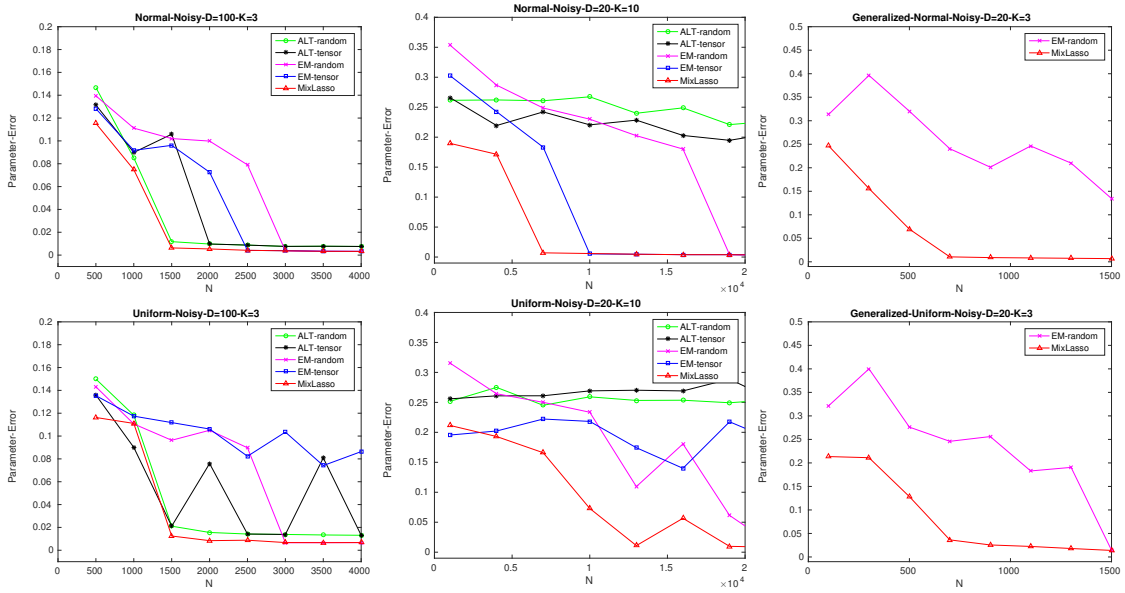

Figure 4.5: Results for Noisy ($\sigma = 0.1$) Mixture of Linear Regression with $N(0, I)$ input distribution (Top) and $U(-1, 1)$ input distribution (Bottom), where (Left) D=100, K=3, (Middle) D=20, K=10, and (Right) Generalized Mixture of Regression with D=20, K=3.

- **ALT-Tensor**: The Alternating Minimization algorithm initialized with Tensor Method proposed in [127]. The formula of Tensor Method is derived assuming $\boldsymbol{x}_i \sim N(0, I)$. We adopt implementation provided by the author of [127].
- **MixLasso**: The proposed estimator with Algorithm 13. We round our solution to exact K components according to the rounding procedure described in the last section for *generalized MR* and *standard MR* respectively. The rounded solution is further refined by EM iterates.

For the linear case, we compare methods using the root mean square error on the learned parameters W compared to the ground-truth parameters W^* of size $K \times D$: $\min_{S:|S|=K} \frac{\|W_{S,:} - W^*\|_F}{\sqrt{DK}}$, where S denotes a multiset that selects the best matched row in W for each row in W^*. For the nonlinear case, we compare methods using RMSE between the predicted value and the ground-truth function value: $\sqrt{\frac{1}{N}\sum_{i=1}^{N}(\sum_{k=1}^{K} z_{ik} f_k(\boldsymbol{x}_i) - \sum_{k=1}^{K} z_{ik}^* f_k^*(\boldsymbol{x}_i))^2}$.

Experiments on Synthetic Data We generate 14 synthetic data sets according to the model:

$$y_i = \sum_{k=1}^{K} z_{ik} f_k(\boldsymbol{x}) + \omega_i, \ i \in [N],$$

where *Syn1*∼*Syn12* are generated by D-dimensional linear models

$$f_k(\boldsymbol{x}) = \boldsymbol{w}_k^T \boldsymbol{x}$$

Data	D	K	$\boldsymbol{x}_i$	$\boldsymbol{z}_i$	ω_i
Syn1	100	3	N(0,I)	Multi(1/3)	0
Syn2	20	10	N(0,I)	Multi(1/10)	0
Syn3	20	3	N(0,I)	Ber(0.5)	0
Syn4	100	3	U(-1,1)	Multi(1/3)	0
Syn5	20	10	U(-1,1)	Multi(1/10)	0
Syn6	20	3	U(-1,1)	Ber(0.5)	0
Syn7	100	3	N(0,I)	Multi(1/3)	N(0,0.1)
Syn8	20	10	N(0,I)	Multi(1/10)	N(0,0.1)
Syn9	20	3	N(0,I)	Ber(0.5)	N(0,0.1)
Syn10	100	3	U(-1,1)	Multi(1/3)	N(0,0.1)
Syn11	20	10	U(-1,1)	Multi(1/10)	N(0,0.1)
Syn12	20	3	U(-1,1)	Ber(0.5)	N(0,0.1)
Data	deg	K	$\boldsymbol{x}_i$	$\boldsymbol{z}_i$	ω_i
Syn13	6	4	U(-1,1)	Multi(1/4)	0
Syn14	6	4	U(-1,1)	Multi(1/4)	N(0,0.1)

Table 4.2: Statistics of the synthetic data set, where Multi($1/K$) means $\boldsymbol{z}_i$ follows a Multinouli distribution with $p_k = 1/K, \forall k \in [K]$, and $Ber(0.5)$ means each component z_{ik} is an independent Bernoulli Random variable with $p = 0.5$.

and *Syn13*$\sim$*Syn14* are generated by 1-dimensional polynomial model of degree 6:

$$f_k(x) = \sum_{j=1}^{6} w_{kj} x^j.$$

We summarize the data statistics in Table 4.2.

Figure 4.4 and 4.5 give experimental results of the linear model in the noiseless and noisy case respectively.

We observe that, in the case of Normal input distribution (Syn1, Syn2, Syn7, Syn8) (top row), both the Tensor-initialized methods and MixLasso consistently improve upon random-initialized EM/ALT (even with 100 trials) in terms of the number of samples required to achieve a good performance, where *ALT* performs better than *EM* in higher dimensional case $(D = 100, K = 3)$ while*EM* performs better for cases of more components $(D = 20, K = 10)$; meanwhile, MixLasso leads to significant improvements in both cases.

On the other hand, when the input distribution becomes U(-1,1) (Syn4, Syn5, Syn10, Syn11), the tensor-initialized method becomes even worse than the random-initialized ones, presumably due to the model mis-specification, while MixLasso still consistently improve upon the random initialized EM/ALT. Note we are testing *Tensor Method derived based on the Normal assumption* on *data with Uniform input* on purpose. The goal is to see the effect of model misspecification on the Tensor approach, as in practice one would always have model misspecification to some degree.

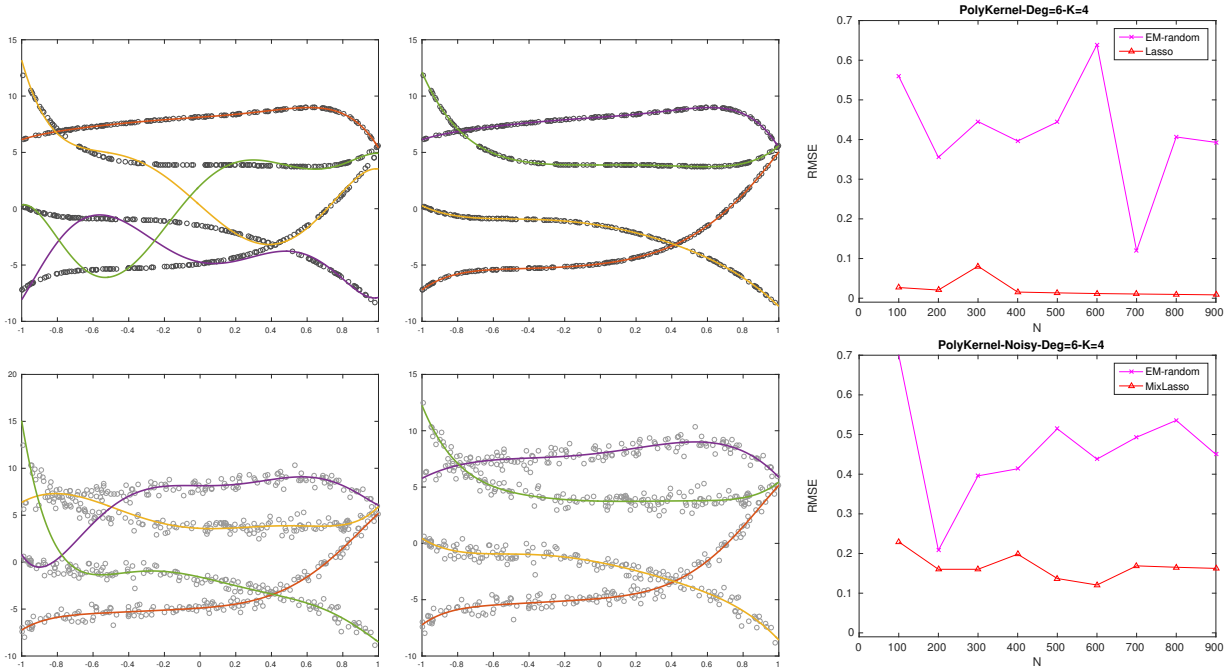

Figure 4.6: Results on Mixture of 6th-order Polynomial Regression of K=4 components with noise (Bottom) and without noise (Top). (Left) The best result of EM out of 100 random initialization. (Middle) Solution from MixLasso followed by fine-tuning EM iterates. (Right) Comparison in terms of RMSE.

The rightmost columns of Figure 4.4, 4.5 show the results on data generated from the *generalized mixed regression* model (Syn3, Syn6, Syn9, Syn12), where Tensor-based methods are not applicable, while MixLasso improves upon EM-Random by a large margin.

Figure 4.6 gives a comparison of EM-Random and MixLasso on Mixture of Kernel Regression with polynomial kernel $\mathcal{K}(\boldsymbol{x}_i, \boldsymbol{x}_j) = (a\boldsymbol{x}_i^\mathcal{T}\boldsymbol{x}_j + b)^d$ $(d = 6)$, where we generate K=4 random 6th-degree polynomial functions $\{f_k^*\}_{k=1}^K$ by uniform sampling their coefficients from $U(-4, 4)$. In this setting, we found EM-Random has a hard time converging to the ground-truth solution even with 100-restarts, while MixLasso obtains solution close to the ground truth with a small number of samples.

Experiments on Real Data In this section, we compare *MixLasso* and *EM* (with 100 restarts) for fitting a mixture of polynomial regression on a *Stock* data set that contains the mixed stock prices of *IBM*, *Facebook*, *Microsoft* and *Nvidia* of span 300 weeks till the Feb. of 2018. The task is to automatically recover the company label of each stock price, while fitting the stock price time series of each company as a polynomial curve. Both EM and MixLasso use a polynomial kernel of the parameters: $\mathcal{K}(\boldsymbol{x}_i, \boldsymbol{x}_j) = (2\boldsymbol{x}_i^\mathcal{T}\boldsymbol{x}_j + 2)^8$.

The results are shown in Figure 4.7. We can see that MixLasso almost recovers the pattern when all samples are given, except for a small number of samples generated by Nvidia's rapid growth recently. While MixLasso consistently achieving a lower RMSE over different sample sizes, the RMSE gap between MixLasso and EM increases as the number of samples grows.

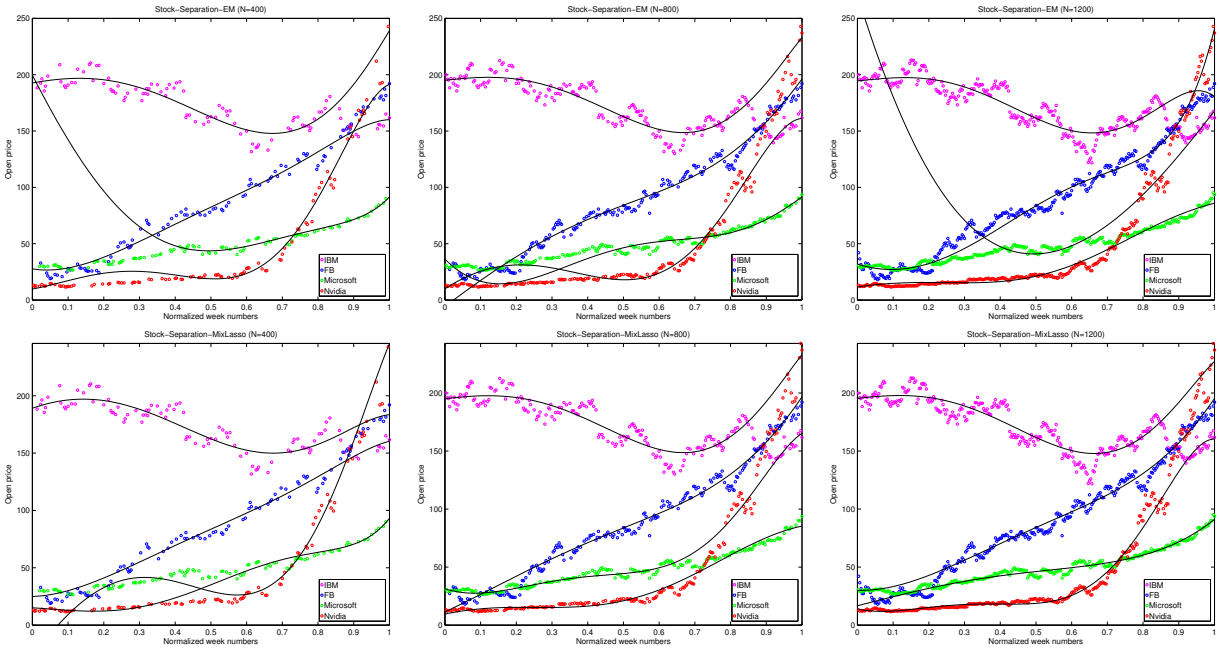

Figure 4.7: Results of fitting mixture of polynomial regressions on the *Stock* data set of increasing number of samples. The top row shows results fitted by EM, and the bottom row shows that from MixLasso. From left to right we have (left) 100 weeks, (middle) 200 weeks, and (right) 300 weeks. From left to right, the RMSE of **EM=**$(\mathbf{6.33}, \mathbf{6.04}, \mathbf{6.27})$ and the RMSE of **MixLasso=**$(\mathbf{6.29}, \mathbf{5.75}, \mathbf{5.58})$.

Chapter 5

Conclusion

Many high-dimensional problems have inherent low-dimensional structures such as the concentration of loss, the sparsity of messages between factors, and the compactness of representation when expressed in terms of an approximate atomic set. By exploiting such structures, one can design new *algorithms* and novel *estimators* of computational complexity sublinear to the domain size without sacrificing expressiveness of the solution. We demonstrate the power of such scheme through the development of state-of-the-art algorithms in Structured and Non-structured Classification of large output domains, and also extend the idea for the compression of Deep Neural Networks. Last but not the least, we design the first polynomial-time estimator for Latent-Feature Models and Generalized Mixed Regression that enjoy strong approximation guarantees without prior knowledge or restrictive assumptions on the data.

Appendix A

Appendix for Chapter 2

A.1 Proof for Theorem 1

The proof of Theorem 1 is similar to that in [59]. To be self-contained, we provide proofs in the following.

The dual problem (2.14) has (generalized) Hessian for i-th block of variable $\boldsymbol{\alpha}^i$ being upper bounded by

$$\nabla^2_{\boldsymbol{\alpha}^i} G(\boldsymbol{\alpha}) \preceq Q_i I.$$

where $Q_i = \|\boldsymbol{x}_i\|^2$. Since the active set includes the most-violating pair (2.19) that defines the Frank-Wolfe direction $\boldsymbol{\alpha}^t_{FW}$ satisfying (2.18), the update given by solving the active-set subproblem (2.21) has

$$\begin{aligned}
&G(\boldsymbol{\alpha}^{t+1}) - G(\boldsymbol{\alpha}^t) \\
&\leq \gamma \langle \nabla_{\boldsymbol{\alpha}^i} G(\boldsymbol{\alpha}^t), \boldsymbol{\alpha}^{it}_{FW} - \boldsymbol{\alpha}^{it} \rangle + \frac{Q_i \gamma^2}{2} \|\boldsymbol{\alpha}^{it}_{FW} - \boldsymbol{\alpha}^{it}\|^2 \\
&\leq \gamma \langle \nabla_{\boldsymbol{\alpha}^i} G(\boldsymbol{\alpha}^t), \boldsymbol{\alpha}^{it}_{FW} - \boldsymbol{\alpha}^{it} \rangle + \frac{Q_i R^2 \gamma^2}{2}
\end{aligned}$$

for any $\gamma \in [0, 1]$, where $\|\boldsymbol{\alpha}^t_{FW} - \boldsymbol{\alpha}^{it}\|^2 \leq R^2 = 4C^2$ since both $\boldsymbol{\alpha}^t_{FW}$, $\boldsymbol{\alpha}^{it}$ lie within the domain (2.16). Taking expectation w.r.t. i (uniformly sampled from $[N]$), we have

$$\begin{aligned}
&E[G(\boldsymbol{\alpha}^{t+1})] - G(\boldsymbol{\alpha}^t) \\
&\leq \frac{\gamma}{N} \langle \nabla_{\boldsymbol{\alpha}} G(\boldsymbol{\alpha}^t), \boldsymbol{\alpha}^t_{FW} - \boldsymbol{\alpha}^t \rangle + \frac{Q R^2 \gamma^2}{2N}
\end{aligned} \tag{A.1}$$

where $Q = \sum_{i=1}^N Q_i$. Then denote $\boldsymbol{\alpha}^*$ as an optimal solution, by convexity and the definition of Frank-Wolfe direction we have

$$\begin{aligned}
\langle \nabla_{\boldsymbol{\alpha}} G(\boldsymbol{\alpha}^t), \boldsymbol{\alpha}^t_{FW} - \boldsymbol{\alpha}^t \rangle &\leq \langle \nabla_{\boldsymbol{\alpha}} G(\boldsymbol{\alpha}^t), \boldsymbol{\alpha}^* - \boldsymbol{\alpha}^t \rangle \\
&\leq G^* - G(\boldsymbol{\alpha}^t),
\end{aligned}$$

where $G^* := G(\boldsymbol{\alpha}^*)$. Together with (A.1), we have

$$\Delta G^{t+1} - \Delta G^t \leq \frac{-\gamma}{N} \Delta G^t + \frac{Q R^2 \gamma^2}{2N} \tag{A.2}$$

for any $\gamma \in [0, 1]$, where $\Delta G^t := E[G(\boldsymbol{\alpha}^t)] - G^*$. By choosing $\gamma = \frac{2N}{t+2N}$, the recurrence (A.2) leads to the result

$$\Delta G^t \leq \frac{2(QR^2 + \Delta G^0)}{t/N + 2},$$

which can be verified via induction as in the proof of Lemma C.2 of [59].

Appendix B

Appendix for Chapter 3

B.1 Proof of Theorem 8 and 9

Recall that the Augmented Lagrangian $\mathcal{L}(\boldsymbol{\alpha}, \boldsymbol{\lambda})$ is of the form

$$\mathcal{L}(\boldsymbol{\alpha}, \boldsymbol{\lambda}) := G(\boldsymbol{\alpha}) + \langle \boldsymbol{\lambda}, M\boldsymbol{\alpha} \rangle + \frac{\rho}{2}\|M\boldsymbol{\alpha}\|^2.$$

where M is "the number of consistency constraints" by "the number of variables" matrix and $M\boldsymbol{\alpha} = \mathbf{0}$ encodes all constraints of the form

$$M_{jf}\boldsymbol{\alpha}_f - \boldsymbol{\alpha}_j = \left[\begin{array}{cc} M_{jf} & -I_j \end{array} \right] \left[\begin{array}{c} \boldsymbol{\alpha}_f \\ \boldsymbol{\alpha}_j \end{array} \right] = \mathbf{0}.$$

The function

$$G(\boldsymbol{\alpha}) = \frac{1}{2}\sum_{F \in \mathcal{F}} \|\boldsymbol{w}_F(\boldsymbol{\alpha}_F)\|^2 - \sum_{j \in \mathcal{U}} \boldsymbol{\delta}_j^T \boldsymbol{\alpha}_j$$

can be written in a compact form as

$$\begin{aligned} G(\boldsymbol{\alpha}) &= \frac{1}{2}\|\boldsymbol{w}(\boldsymbol{\alpha})\|^2 + \boldsymbol{\delta}^T\boldsymbol{\alpha} \\ &= \frac{1}{2}\|\Phi^T\boldsymbol{\alpha}\|^2 + \boldsymbol{\delta}^T\boldsymbol{\alpha} \end{aligned} \tag{B.1}$$

where Φ is the "number of variables (in $\boldsymbol{\alpha}$)" by "number of parameters (in $\boldsymbol{w}$)" design matrix.

Now let $\boldsymbol{\alpha}$ be the "primal variables" and denote

$$\boldsymbol{\alpha}(\boldsymbol{\lambda}) := \{\boldsymbol{\alpha} | \boldsymbol{\alpha} = arg\min_{\boldsymbol{\alpha}} \ \mathcal{L}(\boldsymbol{\alpha}, \boldsymbol{\lambda})\} \tag{B.2}$$

with

$$\bar{\boldsymbol{\alpha}}^t := \underset{\bar{\boldsymbol{\alpha}} \in \boldsymbol{\alpha}(\boldsymbol{\lambda}^t)}{argmin} \|\bar{\boldsymbol{\alpha}} - \boldsymbol{\alpha}^t\|,$$

and let $\mathcal{M} = \{\boldsymbol{\alpha} \mid \boldsymbol{\alpha}_f \in \Delta^{|\mathcal{Y}_f|}, \forall f \in \mathcal{F}\}$. The dual objective of the augmented problem is

$$d(\boldsymbol{\lambda}) = \min_{\boldsymbol{\alpha} \in \mathcal{M}} \mathcal{L}(\boldsymbol{\alpha}, \boldsymbol{\lambda})$$

and

$$d^* = \max_{\boldsymbol{\lambda}} d(\boldsymbol{\lambda})$$

is the optimal dual objective value.

Then we measure the sub-optimality of iterates $\{(\boldsymbol{\alpha}^t, \boldsymbol{\lambda}^t)\}_{t=1}^T$ given by GDMM in terms of dual function difference

$$\Delta_d^t = d^* - d(\boldsymbol{\lambda}^t)$$

and the primal function difference for a given dual iterate $\boldsymbol{\lambda}^t$:

$$\Delta_p^t = \mathcal{L}(\boldsymbol{\alpha}^{t+1}, \boldsymbol{\lambda}^t) - d(\boldsymbol{\lambda}^t)$$

yielded by $\boldsymbol{\alpha}^{t+1}$ obtained from one pass of FC-BCFW algorithm on $\boldsymbol{\alpha}$. Then we have following lemma.

Lemma 2 (Dual Progress)**.** *Each iteration of GDMM (Algorithm 5) has*

$$\Delta_d^t - \Delta_d^{t-1} \leq -\eta (M\boldsymbol{\alpha}^t)^T (M\bar{\boldsymbol{\alpha}}^t). \tag{B.3}$$

Proof.

$$\begin{aligned}
\Delta_d^t - \Delta_d^{t-1} &= d^* - d(\boldsymbol{\lambda}^t) - d^* - d(\boldsymbol{\lambda}^{t-1}) \\
&= \mathcal{L}(\bar{\boldsymbol{\alpha}}^{t-1}, \boldsymbol{\lambda}^{t-1}) - \mathcal{L}(\bar{\boldsymbol{\alpha}}^t, \boldsymbol{\lambda}^t) \\
&\leq \mathcal{L}(\bar{\boldsymbol{\alpha}}^t, \boldsymbol{\lambda}^{t-1}) - \mathcal{L}(\bar{\boldsymbol{\alpha}}^t, \boldsymbol{\lambda}^t) \\
&= \langle \boldsymbol{\lambda}^{t-1} - \boldsymbol{\lambda}^t, M\bar{\boldsymbol{\alpha}}^t \rangle \\
&= -\eta \langle M\boldsymbol{\alpha}^t, M\bar{\boldsymbol{\alpha}}^t \rangle
\end{aligned}$$

where the first inequality follows the optimality of $\bar{\boldsymbol{\alpha}}^{t-1}$ for the function $\mathcal{L}(\boldsymbol{\alpha}, \boldsymbol{\lambda}^{t-1})$ defined by $\boldsymbol{\lambda}^{t-1}$, and the last equality follows the dual update in GDMM (3.14). □

On the other hand, the following lemma gives an expression on the primal progress that is independent of the algorithm used for minimizing Augmented Lagrangian

Lemma 3 (Primal Progress)**.** *Each iteration of GDMM (Algorithm 5) has*

$$\begin{aligned}
\Delta_p^t - \Delta_p^{t-1} \leq & \mathcal{L}(\boldsymbol{\alpha}^{t+1}, \boldsymbol{\lambda}^t) - \mathcal{L}(\boldsymbol{\alpha}^t, \boldsymbol{\lambda}^t) \\
& + \eta \|M\boldsymbol{\alpha}^t\|^2 - \eta \langle M\boldsymbol{\alpha}^t, M\bar{\boldsymbol{\alpha}}^t \rangle
\end{aligned}$$

Proof.

$$\begin{aligned}
& \Delta_p^t - \Delta_p^{t-1} \\
=& \mathcal{L}(\boldsymbol{\alpha}^{t+1}, \boldsymbol{\lambda}^t) - \mathcal{L}(\boldsymbol{\alpha}^t, \boldsymbol{\lambda}^{t-1}) - (d(\boldsymbol{\lambda}^t) - d(\boldsymbol{\lambda}^{t-1})) \\
\leq& \mathcal{L}(\boldsymbol{\alpha}^{t+1}, \boldsymbol{\lambda}^t) - \mathcal{L}(\boldsymbol{\alpha}^t, \boldsymbol{\lambda}^t) + \mathcal{L}(\boldsymbol{\alpha}^t, \boldsymbol{\lambda}^t) - \mathcal{L}(\boldsymbol{\alpha}^t, \boldsymbol{\lambda}^{t-1}) + (d(\boldsymbol{\lambda}^{t-1}) - d(\boldsymbol{\lambda}^t)) \\
\leq& \mathcal{L}(\boldsymbol{\alpha}^{t+1}, \boldsymbol{\lambda}^t) - \mathcal{L}(\boldsymbol{\alpha}^t, \boldsymbol{\lambda}^t) + \eta \|M\boldsymbol{\alpha}^t\|^2 - \eta \langle M\boldsymbol{\alpha}^t, M\bar{\boldsymbol{\alpha}}^t \rangle
\end{aligned}$$

where the last inequality uses Lemma 8 on $d(\boldsymbol{\lambda}^{t-1}) - d(\boldsymbol{\lambda}^t) = \Delta_d^t - \Delta_d^{t-1}$. □

By combining results of Lemma 8 and 9, we can obtain a joint progress of the form

$$\begin{aligned}&\Delta_d^t - \Delta_d^{t-1} + \Delta_p^t - \Delta_p^{t-1} \\ &\leq \mathcal{L}(\boldsymbol{\alpha}^{t+1}, \boldsymbol{\lambda}^t) - \mathcal{L}(\boldsymbol{\alpha}^t, \boldsymbol{\lambda}^t) + \eta\|M\boldsymbol{\alpha}^t - M\bar{\boldsymbol{\alpha}}^t\|^2 - \eta\|M\bar{\boldsymbol{\alpha}}^t\|^2\end{aligned} \tag{B.4}$$

Note the only positive term in (B.32) is the second one. To guarantee the descent of joint progress, we upper bound the three terms in (B.32) with the following lemmas.

Lemma 4.

$$\|M\boldsymbol{\alpha}^t - M\bar{\boldsymbol{\alpha}}^t\|^2 \leq \frac{2}{\rho}(\mathcal{L}(\boldsymbol{\alpha}^t, \boldsymbol{\lambda}^t) - \mathcal{L}(\bar{\boldsymbol{\alpha}}^t, \boldsymbol{\lambda}^t)) \tag{B.5}$$

Proof. Let

$$\tilde{\mathcal{L}}(\boldsymbol{\alpha}, \boldsymbol{\lambda}) = h(\boldsymbol{\alpha}) + \frac{\rho}{2}\|M\boldsymbol{\alpha}\|^2,$$

where

$$h(\boldsymbol{\alpha}) = G(\boldsymbol{\alpha}) + \langle \boldsymbol{\lambda}, M\boldsymbol{\alpha}\rangle + \boldsymbol{I}_{\boldsymbol{\alpha}\in\mathcal{M}}.$$

, $\boldsymbol{I}_{\boldsymbol{\alpha}\in\mathcal{M}} = 0$ if $\boldsymbol{\alpha} \in \mathcal{M}$ and $\boldsymbol{I}_{\boldsymbol{\alpha}\in\mathcal{M}} = \infty$ otherwise. Note we have $\tilde{\mathcal{L}}(\bar{\boldsymbol{\alpha}}^t, \boldsymbol{\lambda}^t) = \mathcal{L}(\bar{\boldsymbol{\alpha}}^t, \boldsymbol{\lambda}^t)$ and $\tilde{\mathcal{L}}(\boldsymbol{\alpha}^t, \boldsymbol{\lambda}^t) = \mathcal{L}(\boldsymbol{\alpha}^t, \boldsymbol{\lambda}^t)$ due to feasible iterates. Due to the optimality of $\bar{\boldsymbol{\alpha}}^t$, we have

$$\mathbf{0} = \boldsymbol{\sigma} + M^T M\bar{\boldsymbol{\alpha}}^t \in \partial_{\boldsymbol{\alpha}}\tilde{\mathcal{L}}(\bar{\boldsymbol{\alpha}}^t, \boldsymbol{\lambda})$$

for some $\boldsymbol{\sigma} \in \partial h(\bar{\boldsymbol{\alpha}}^t)$. And by the convexity of $h(\cdot)$ and the strong convexity of $\frac{\rho}{2}\|\cdot\|^2$, we have

$$h(\boldsymbol{\alpha}^t) - h(\bar{\boldsymbol{\alpha}}^t) \geq \langle \boldsymbol{\sigma}, \boldsymbol{\alpha}^t - \bar{\boldsymbol{\alpha}}^t\rangle$$

and

$$\frac{\rho}{2}\|M\boldsymbol{\alpha}^t\|^2 - \frac{\rho}{2}\|M\bar{\boldsymbol{\alpha}}^t\|^2 \geq \langle M^T M\bar{\boldsymbol{\alpha}}^t, \boldsymbol{\alpha}^t - \bar{\boldsymbol{\alpha}}^t\rangle + \frac{\rho}{2}\|M\boldsymbol{\alpha}^t - M\bar{\boldsymbol{\alpha}}^t\|^2$$

Then the above two together imply

$$\mathcal{L}(\boldsymbol{\alpha}^t, \boldsymbol{\lambda}^t) - \mathcal{L}(\bar{\boldsymbol{\alpha}}^t, \boldsymbol{\lambda}^t) \geq \frac{\rho}{2}\|M(\boldsymbol{\alpha}^t) - M(\bar{\boldsymbol{\alpha}}^t)\|^2$$

which leads to our conclusion. □

Lemma 5 (*Hong and Luo 2012*)**.** *There is a constant $\tau > 0$ such that*

$$\Delta_d(\boldsymbol{\lambda}) \leq \tau\|M\bar{\boldsymbol{\alpha}}(\boldsymbol{\lambda})\|^2. \tag{B.6}$$

for any $\boldsymbol{\lambda}$ and any minimizer $\bar{\boldsymbol{\alpha}}(\boldsymbol{\lambda})$ satisfying (B.2)*.*

Proof. This is a lemma adapted from [42]. Since our objective (3.13) satisfies the *assumptions A(a)—A(e)* and *A(g)* in [42]. Then Lemma 3.1 of [42] guarantees that, as long as $\|\nabla d(\boldsymbol{\lambda})\|$ is bounded, there is a constant $\tau > 0$ s.t.

$$\Delta_d(\boldsymbol{\lambda}) \leq \tau\|\nabla d(\boldsymbol{\lambda})\|^2 = \|M\bar{\boldsymbol{\alpha}}(\boldsymbol{\lambda})\|^2$$

for all $\boldsymbol{\lambda}$. Note our problem satisfies the condition of bounded gradient magnitude since

$$\|\nabla d(\boldsymbol{\lambda})\| = \|M\bar{\boldsymbol{\alpha}}(\boldsymbol{\lambda})\| \leq \|M\bar{\boldsymbol{\alpha}}(\boldsymbol{\lambda})\|_1 \leq \|M\|_1\|\bar{\boldsymbol{\alpha}}(\boldsymbol{\lambda})\|_1 \leq (\max_f |\mathcal{Y}_f|)|\mathcal{F}|$$

where the last inequality is because $\bar{\boldsymbol{\alpha}}(\boldsymbol{\lambda})$ lies in a simplex domain. □

The remaining thing is to show that one pass of AFW (Algorithm 6) or BGCD (Algorithm 7) suffices to give a descent amount $\mathcal{L}(\boldsymbol{\alpha}^{t+1}, \boldsymbol{\lambda}^t) - \mathcal{L}(\boldsymbol{\alpha}^t, \boldsymbol{\lambda}^t)$ lower bounded by some constant multiple of the primal sub-optimality $\mathcal{L}(\boldsymbol{\alpha}^t, \boldsymbol{\lambda}^t) - \mathcal{L}(\bar{\boldsymbol{\alpha}}^t, \boldsymbol{\lambda}^t)$. If it is true, then by selecting a small enough GDMM step size η, the RHS of (B.32) would be negative. For AFW (Algorithm 6), this can be achieved by leveraging recent results from [57], which shows linear convergence of AFW, even for non-strongly convex function of the form (B.43). We thus have the following lemma.

Lemma 6. *The descent amount of Augmented Lagrangian function produced by one pass of AFW (Algorithm 6) (and FMO parameter ν) has*

$$E[\mathcal{L}(\boldsymbol{\alpha}^{t+1}, \boldsymbol{\lambda}^t)] - \mathcal{L}(\boldsymbol{\alpha}^t, \boldsymbol{\lambda}^t) \leq -\frac{\mu_{\mathcal{M}}}{4(1+\nu)mQ}(\mathcal{L}(\boldsymbol{\alpha}^t, \boldsymbol{\lambda}^t) - \mathcal{L}(\bar{\boldsymbol{\alpha}}^t, \boldsymbol{\lambda}^t)) \tag{B.7}$$

where $\mu_{\mathcal{M}}$ is the generalized geometric strong convexity constant for function $\mathcal{L}(\boldsymbol{\alpha})$ in domain $\mathcal{M}$, Q is the Lipschitz-continuous constant of $\nabla_{\boldsymbol{\alpha}}\mathcal{L}(\boldsymbol{\alpha})$ and $m = |\mathcal{F}|$.

Proof. Note the Augmented Lagrangian is of the form

$$H(\boldsymbol{\alpha}) := \mathcal{L}(\boldsymbol{\alpha}, \boldsymbol{\lambda}^t) = g(B\boldsymbol{\alpha}) + \langle \boldsymbol{b}, \boldsymbol{\alpha} \rangle \tag{B.8}$$

where

$$B := \begin{bmatrix} \Phi^T \\ M \end{bmatrix}, \quad \boldsymbol{b} := \boldsymbol{\delta} + M^T \boldsymbol{\lambda}^t$$

and function $g(\begin{bmatrix} \boldsymbol{w} \\ \boldsymbol{v} \end{bmatrix}) = \frac{1}{2}\|\boldsymbol{w}\|^2 + \frac{\rho}{2}\|\boldsymbol{v}\|^2 + const.$ is strongly convex with parameter $\bar{\rho} = \min(1, \rho)$. Without loss of generality, assume $\rho \leq 1$ and thus $\bar{\rho} = \rho$. Since we are minimizing the function subject to a convex, polyhedral domain $\mathcal{M}$, by Theorem 10 of [57], we have the *generalized geometrical strong convexity* constant $\mu_{\mathcal{M}}$ of the form

$$\mu_{\mathcal{M}} := \mu(PWidth(\mathcal{M}))^2 \tag{B.9}$$

where $PWidth(\mathcal{M}) > 0$ is the pyramidal width of the simplex domain $\mathcal{M}$ and μ is the *generalized strong convexity* constant of function (B.43) (defined by Lemma 9 of [57]). By definition of the geometric strong convexity constant, we have

$$H(\boldsymbol{\alpha}^t) - H^* \leq \frac{g_t^2}{2\mu_{\mathcal{M}}} \tag{B.10}$$

from (23) in [57], where $g_t := \langle -\nabla H(\boldsymbol{\alpha}^t), \boldsymbol{v}^F - \boldsymbol{v}^A \rangle$, and $\boldsymbol{v}^F$ is the Frank-Wolfe direction

$$\boldsymbol{v}^F := arg\min_{\boldsymbol{v} \in \mathcal{M}} \langle \nabla H(\boldsymbol{\alpha}^t), \boldsymbol{v} \rangle,$$

$\boldsymbol{v}^A$ is the away direction

$$\boldsymbol{v}^A := arg\max_{\boldsymbol{v} \in \mathcal{A}^t} \langle \nabla H(\boldsymbol{\alpha}^t), \boldsymbol{v} \rangle$$

Then let $m = |\mathcal{F}|$ be the number of factors. The FMO returns $\boldsymbol{v}_f^+ = \boldsymbol{v}_f^F$ with probability at least $\frac{1}{\nu}$, and suppose we set $\boldsymbol{v}_f^+$ to $\boldsymbol{\alpha}_f^t$ whenever $\langle \nabla_{\boldsymbol{\alpha}_f} H, \boldsymbol{v}_f^+ - \boldsymbol{\alpha}_f^t \rangle \nleq 0$. We have $\langle \nabla H, \boldsymbol{d}_F \rangle \leq \frac{1}{\nu}\langle \nabla H, \boldsymbol{v}^F - \boldsymbol{\alpha}^t \rangle$ and thus

$$(1 + \frac{1}{\nu})\langle \nabla H, \boldsymbol{d}^t \rangle \leq \frac{1}{\nu}\langle \nabla H, \boldsymbol{v}^F - \boldsymbol{\alpha}^t \rangle + \frac{1}{\nu}\langle \nabla H, \boldsymbol{\alpha}^t - \boldsymbol{v}^A \rangle$$

and $\langle \nabla H, \boldsymbol{d}^t \rangle \leq -\frac{1}{1+\nu} g_t$. Therefore, for any $\forall \gamma \in [0, 1]$,

$$E[H(\boldsymbol{\alpha}^{t+1})] - H(\boldsymbol{\alpha}^t) \leq -\gamma \frac{g_t}{1+\nu} + \frac{Q}{2}\|\gamma(\boldsymbol{\alpha}^{t+1} - \boldsymbol{\alpha}^t)\|^2 \leq -\gamma \frac{g_t}{1+\nu} + \frac{2mQ}{2}\gamma^2 \tag{B.11}$$

where Q is an upper bound on the spectral norm of Hessian $\|\nabla^2 H(\boldsymbol{\alpha})\|$ and $2m$ is the square of the radius of domain $\mathcal{M}$. Now we need to consider two cases. When the greedy direction $\boldsymbol{d}_F$ in (3.18) is chosen, we have $\gamma^* = \min(\frac{g_s}{mQ_{\max}}, 1)$, which gives us

$$E[H(\boldsymbol{\alpha}^{t+1})] - H(\boldsymbol{\alpha}^t) \leq -\frac{g_t^2}{4(1+\nu)mQ}. \tag{B.12}$$

While in case $\boldsymbol{d}_A$ in (3.18) is chosen, we have $\gamma^* = \min(\frac{g_s}{mQ_{\max}}, c_{\boldsymbol{v}^-})$. When $\gamma^* = c_{\boldsymbol{v}^-}$, a basis $\boldsymbol{v}^-$ is removed from the active set and this is called a *drop step* [57] and it is hard to show sufficient descent in this case. Nevertheless, we can ignore those drop steps since the number of them is at most half of the iterates. For a non-drop step t, with the error bound (B.46), we have

$$E[H(\boldsymbol{\alpha}^{t+1})] - H(\boldsymbol{\alpha}^t) \leq -\frac{\mu_{\mathcal{M}}(H(\boldsymbol{\alpha}^t) - H^*)}{4(1+\nu)mQ}. \tag{B.13}$$

□

The above Lemma shows a significant progress made by the AFW algorithm. In the following, we provide a similar Lemma for minimizing AL subproblem with the Block-Greedy Coordinate Descent (BGCD) (Algorithm 7). Note that for problem of the form (B.43), the optimal solution is profiled by a polyhedral set $\mathcal{S} := \{\boldsymbol{\alpha} \mid B\boldsymbol{\alpha} = \boldsymbol{t}^*,\ \boldsymbol{b}^T\boldsymbol{\alpha} = s^*,\ \boldsymbol{\alpha} \in \mathcal{M}\}$. Therefore, let $\bar{\boldsymbol{\alpha}} := \Pi_{\mathcal{S}}(\boldsymbol{\alpha})$. We can bound the distance of any feasible point $\boldsymbol{\alpha} \in \mathcal{M}$ to its projection $\Pi_{\mathcal{S}}(\boldsymbol{\alpha})$ on $\mathcal{S}$ using the Hoffman's inequality [41]

$$\|\bar{\boldsymbol{\alpha}} - \boldsymbol{\alpha}\|_{2,1}^2 = \sum_{i=1}^{n}(\sum_{f \in \mathcal{F}_i} \|\bar{\boldsymbol{\alpha}}_f - \boldsymbol{\alpha}_f\|_2)^2 \leq \theta_1 \left(\|B\boldsymbol{\alpha} - \boldsymbol{t}^*\|^2 + \|\boldsymbol{b}^T\boldsymbol{\alpha} - s^*\|^2\right) \tag{B.14}$$

where θ_1 is a constant depending on the set $\mathcal{S}$. Then we can establish the following Lemma using the error bound (B.52).

Lemma 7. *The descent amount of Augmented Lagrangian function given by one pass of BGCD (Algorithm 7) with FMO multiplicative-approximation parameter ν has*

$$E[\mathcal{L}(\boldsymbol{\alpha}^{t+1}, \boldsymbol{\lambda}^t)] - \mathcal{L}(\boldsymbol{\alpha}^t, \boldsymbol{\lambda}^t) \leq \frac{-1}{1 + \nu Q_{\max}/\mu_1}(\mathcal{L}(\boldsymbol{\alpha}^t, \boldsymbol{\lambda}^t) - \mathcal{L}(\bar{\boldsymbol{\alpha}}^t, \boldsymbol{\lambda}^t)) \tag{B.15}$$

where

$$\mu_1 := \frac{1}{\max\{16\theta_1 \Delta\mathcal{L}^0, 2\theta_1(1+4L_g^2)\}}.$$

is the generalized strong convexity constant for function $\mathcal{L}(\boldsymbol{\alpha})$ with feasible domain $\mathcal{M}$, $\Delta\mathcal{L}^0$ is a bound on $\mathcal{L}(\boldsymbol{\alpha}^0) - \mathcal{L}(\bar{\boldsymbol{\alpha}}^0)$, L_g is the local Lipschitz-continuous constant of $g(.)$ and $Q_{\max} = \max_{f\in\mathcal{F}} Q_f$.

Proof. For each iteration s of Algorithm 7, let i be the chosen sample and suppose that out of ν partitions the one containing greedy factor satisfying (3.19) is chosen. We have

$$\begin{aligned}\mathcal{L}(\boldsymbol{\alpha}^{s+1}) - \mathcal{L}(\boldsymbol{\alpha}^s) &\leq \min_{\boldsymbol{\alpha}^s_{f^*}+\boldsymbol{d}_{f^*}\in\Delta^{|\mathcal{Y}_{f^*}|}} \langle \nabla_{\boldsymbol{\alpha}_{f^*}}\mathcal{L}, \boldsymbol{d}_{f^*}\rangle + \frac{Q_{\max}}{2}\|\boldsymbol{d}_{f^*}\|^2 \\ &= \min_{\boldsymbol{\alpha}^s_f+\boldsymbol{d}_f\in\Delta^{|\mathcal{Y}_f}} + \sum_{f\in\mathcal{F}_i}\langle \nabla_{\boldsymbol{\alpha}_f}\mathcal{L}, \boldsymbol{d}_f\rangle + \frac{Q_{\max}}{2}\left(\sum_{f\in\mathcal{F}_i}\|\boldsymbol{d}_f\|\right)^2\end{aligned} \tag{B.16}$$

where the second equality follows from the optimality of f^* w.r.t. (3.19). Then consider i being uniformly sampled from $[n]$, and consider the probability that the partition containing greedy factor f^* is chosen, the expected descent amount is

$$\begin{aligned}&E[\mathcal{L}(\boldsymbol{\alpha}^{s+1})] - \mathcal{L}(\boldsymbol{\alpha}^s) \\ &\leq \frac{1}{n\nu}\left(\min_{\boldsymbol{\alpha}^s_f+\boldsymbol{d}_f\in\Delta^{|\mathcal{Y}_f}} \sum_{f\in\mathcal{F}}\langle \nabla_{\boldsymbol{\alpha}_f}\mathcal{L}, \boldsymbol{d}_f\rangle + \frac{Q_{\max}}{2}\sum_{i=1}^{n}\left(\sum_{f\in\mathcal{F}_i}\|\boldsymbol{d}_f\|\right)^2\right) \\ &\leq \frac{1}{n\nu}\left(\min_{\boldsymbol{\alpha}^s_f+\boldsymbol{d}_f\in\Delta^{|\mathcal{Y}_f}} \mathcal{L}(\boldsymbol{\alpha}^s+\boldsymbol{d}) - \mathcal{L}(\boldsymbol{\alpha}^s) + \frac{Q_{\max}}{2}\sum_{i=1}^{n}\left(\sum_{f\in\mathcal{F}_i}\|\boldsymbol{d}_f\|\right)^2\right) \\ &\leq \frac{1}{n\nu}\left(\min_{\beta\in[0,1]} \mathcal{L}(\boldsymbol{\alpha}^s + \beta(\bar{\boldsymbol{\alpha}}^s - \boldsymbol{\alpha}^s)) - \mathcal{L}(\boldsymbol{\alpha}^s) + \frac{Q_{\max}\beta^2}{2}\sum_{i=1}^{n}\left(\sum_{f\in\mathcal{F}_i}\|\bar{\boldsymbol{\alpha}}^s_f - \boldsymbol{\alpha}^s_f\|\right)^2\right) \\ &\leq \frac{1}{n\nu}\left(\min_{\beta\in[0,1]} \beta(\mathcal{L}(\bar{\boldsymbol{\alpha}}^s) - \mathcal{L}(\boldsymbol{\alpha}^s)) + \frac{Q_{\max}\beta^2}{2}\|\bar{\boldsymbol{\alpha}}^s - \boldsymbol{\alpha}^s\|_{2,1}^2\right)\end{aligned} \tag{B.17}$$

where $\bar{\boldsymbol{\alpha}}^s = \Pi_{\mathcal{S}}(\boldsymbol{\alpha}^s)$ is the projection of $\bar{\boldsymbol{\alpha}}^s$ to the optimal solution set $\mathcal{S}$. The second and last inequality is due to convexity, and the third inequality is due to a confinement of optimization domain. Then we discuss two cases in the following.

Case 1: $4L_g^2\|B\boldsymbol{\alpha}^s - \boldsymbol{t}^*\|^2 < (\boldsymbol{b}^T\boldsymbol{\alpha}^s - s^*)^2$.

In this case, by the hoffman inequality (B.52), we have

$$\begin{aligned}\|\boldsymbol{\alpha}^s - \bar{\boldsymbol{\alpha}}^s\|_{2,1}^2 &\leq \theta_1(\|B\bar{\boldsymbol{\alpha}}^s - \boldsymbol{t}^*\|^2 + (\boldsymbol{b}^T\boldsymbol{\alpha}^s - s^*)^2) \\ &\leq \theta_1(\frac{1}{4L_g^2}+1)(\boldsymbol{b}^T\boldsymbol{\alpha}^s - s^*)^2 \\ &\leq 2\theta_1(\boldsymbol{b}^T\boldsymbol{\alpha}^s - s^*)^2,\end{aligned} \tag{B.18}$$

since $\frac{1}{4L_g^2} \leq 1$. Then

$$|\boldsymbol{b}^T\boldsymbol{\alpha}^s - s^*| \geq 2L_g\|B\boldsymbol{\alpha}^s - \boldsymbol{t}^*\| \geq 2|g(B\boldsymbol{\alpha}^s) - g(\boldsymbol{t}^*)|$$

by the definition of Lipschitz constant L_g. Note that $\boldsymbol{b}^T\boldsymbol{\alpha}^s - s^*$ is non-negative since otherwise we have contradiction $\mathcal{L}(\boldsymbol{\alpha}^s) - \mathcal{L}^* = g(B\boldsymbol{\alpha}^s) - g(\boldsymbol{t}^*) + (\boldsymbol{b}^T\boldsymbol{\alpha}^s - s^*) \leq |g(B\boldsymbol{\alpha}^s) - g(\boldsymbol{t}^*)| - |\boldsymbol{b}^T\boldsymbol{\alpha}^s - s^*| \leq -\frac{1}{2}|\boldsymbol{b}^T\boldsymbol{\alpha}^s - s^*| < 0$. Therefore, we have

$$\begin{aligned}\mathcal{L}(\boldsymbol{\alpha}^s) - \mathcal{L}^* &= g(B\boldsymbol{\alpha}^s) - g(\boldsymbol{t}^*) + (\boldsymbol{b}^T\boldsymbol{\alpha}^s - s^*) \\ &\geq -|g(B\boldsymbol{\alpha}^s) - g(\boldsymbol{t}^*)| + (\boldsymbol{b}^T\boldsymbol{\alpha}^s - s^*) \\ &\geq \frac{1}{2}(\boldsymbol{b}^T\boldsymbol{\alpha}^s - s^*).\end{aligned} \tag{B.19}$$

Combining (B.54), (B.55) and (B.56), we have

$$\begin{aligned}&\mathbb{E}[\mathcal{L}(\boldsymbol{\alpha}^{s+1})] - \mathcal{L}(\boldsymbol{\alpha}^s) \\ &\leq \frac{1}{n\nu}\left(\min_{\beta\in[0,1]} -\frac{\beta}{2}(\boldsymbol{b}^T\boldsymbol{\alpha}^s - s^*) + \frac{2\theta_1 Q_{\max}\beta^2}{2}(\boldsymbol{b}^T\boldsymbol{\alpha}^s - s^*)^2\right) \\ &= \begin{cases} -1/(16\theta_1 Q_{\max} n\nu) & ,\ 1/(4\theta_1 Q_{\max}(\boldsymbol{b}^T\boldsymbol{\alpha}^s - s^*)) \leq 1 \\ -\frac{1}{4n\nu}(\boldsymbol{b}^T\boldsymbol{\alpha}^s - s^*) & ,\ o.w. \end{cases}\end{aligned}$$

Furthermore, we have

$$-\frac{1}{16Q_{\max}\theta_1 n\nu} \leq -\frac{1}{16Q_{\max}\theta_1 n\nu(\mathcal{L}^0 - \mathcal{L}^*)}\left(\mathcal{L}(\boldsymbol{\alpha}^s) - \mathcal{L}^*\right) \tag{B.20}$$

where $\mathcal{L}^0 = \mathcal{L}(\boldsymbol{\alpha}^0)$, and

$$-\frac{1}{4n\nu}(\boldsymbol{b}^T\boldsymbol{\alpha}^s - s^*) \leq -\frac{1}{6n\nu}(\mathcal{L}(\boldsymbol{\alpha}^s) - \mathcal{L}^*) \tag{B.21}$$

since $\mathcal{L}(\boldsymbol{\alpha}^s) - \mathcal{L}^* \leq |g(B\boldsymbol{\alpha}^s) - g(\boldsymbol{t}^*)| + \boldsymbol{b}^T\boldsymbol{\alpha}^s - s^* \leq \frac{3}{2}(\boldsymbol{b}^T\boldsymbol{\alpha}^s - s^*)$. Since the bound (B.20) is much smaller than (B.21). For Case 1, we obtain

$$\mathbb{E}[\mathcal{L}(\boldsymbol{\alpha}^{s+1})] - \mathcal{L}^s \leq -\frac{\mu_1}{n\nu Q_{\max}}\left(\mathcal{L}(\boldsymbol{\alpha}^s) - \mathcal{L}^*\right) \tag{B.22}$$

where

$$\mu_1 = \frac{1}{16\theta(\mathcal{L}^0 - \mathcal{L}^*)}. \tag{B.23}$$

Case 2: $4L_g^2\|B\boldsymbol{\alpha}^s - \boldsymbol{t}^*\|^2 \geq (\boldsymbol{b}^T\boldsymbol{\alpha}^s - s^*)^2$.
In this case, we have

$$\|\boldsymbol{\alpha}^s - \bar{\boldsymbol{\alpha}}^s\|_{2,1}^2 \leq \theta_1\left(1 + 4L_g^2\right)\|B\boldsymbol{\alpha}^s - \boldsymbol{t}^*\|^2, \tag{B.24}$$

and by strong convexity of $g(.)$,

$$\mathcal{L}(\boldsymbol{\alpha}^s) - \mathcal{L}^* \geq \boldsymbol{b}^T(\boldsymbol{\alpha}^s - \boldsymbol{\alpha}^*) + \nabla g(t^*)^T B(\boldsymbol{\alpha}^s - \bar{\boldsymbol{\alpha}}^s) + \frac{\rho}{2}\|B\boldsymbol{\alpha}^s - \boldsymbol{t}^*\|^2.$$

Now let $h(\boldsymbol{\alpha})$ be a function that takes value 0 when $\boldsymbol{\alpha}$ is feasible and takes value ∞ otherwise. Adding inequality $0 = h(\boldsymbol{\alpha}^s) - h(\bar{\boldsymbol{\alpha}}^s) \geq \langle \boldsymbol{\sigma}^*, \boldsymbol{\alpha}^s - \bar{\boldsymbol{\alpha}}^s \rangle$ to the above gives

$$\mathcal{L}(\boldsymbol{\alpha}^s) - \mathcal{L}^* \geq \frac{\rho}{2} \|B\boldsymbol{\alpha}^s - \boldsymbol{t}^*\|^2 \tag{B.25}$$

because $\boldsymbol{\sigma}^* + \boldsymbol{b} + \nabla g(\boldsymbol{t}^*)^T B = \boldsymbol{\sigma}^* + \nabla \mathcal{L}(\bar{\boldsymbol{\alpha}}^s) = 0$ for some $\boldsymbol{\sigma}^* \in \partial h(\bar{\boldsymbol{\alpha}}^s)$. Combining (B.54), (B.59), and (B.60), we obtain

$$\begin{aligned}
&\mathbb{E}[\mathcal{L}(\boldsymbol{\alpha}^{s+1})] - \mathcal{L}(\boldsymbol{\alpha}^s) \\
&\leq \frac{1}{n\nu} \left(\min_{\beta \in [0,1]} -\beta(\mathcal{L}(\boldsymbol{\alpha}^s) - \mathcal{L}^*) + \frac{\theta_1(1+4L_g^2)Q_{\max}\beta^2}{\rho} (\mathcal{L}(\boldsymbol{\alpha}^s) - \mathcal{L}^*) \right) \\
&\leq -\frac{\rho}{n\nu\theta_1(1+4L_g^2)Q_{\max}} (\mathcal{L}(\boldsymbol{\alpha}^s) - \mathcal{L}^*)
\end{aligned} \tag{B.26}$$

Combining results of Case 1 (B.57) and Case 2 (B.61), and taking expectation on both sides w.r.t. the history, we have

$$E[\mathcal{L}(\boldsymbol{\alpha}^{s+1})] - \mathcal{L}(\boldsymbol{\alpha}^s) \leq -\frac{\mu_1}{Q_{\max} n\nu} (\mathcal{L}(\boldsymbol{\alpha}^s) - \mathcal{L}^*). \tag{B.27}$$

where

$$\mu_1 := \min\{\frac{1}{16\theta(\Delta\mathcal{L}^0)}, \frac{\rho}{\theta_1(1+4L_g^2)}\}.$$

Taking summation of (B.62) over iterates $s = 1...n$, we have

$$\begin{aligned}
E[\mathcal{L}(\boldsymbol{\alpha}^{t+1})] - \mathcal{L}(\boldsymbol{\alpha}^t) &\leq -\frac{\mu_1}{Q_{\max} n\nu} (\sum_{s=1}^{n} \mathcal{L}(\boldsymbol{\alpha}^s) - \mathcal{L}^*) \\
&\leq -\frac{\mu_1}{Q_{\max}\nu} (\mathcal{L}(\boldsymbol{\alpha}^{t+1}) - \mathcal{L}^*).
\end{aligned} \tag{B.28}$$

Rearranging terms gives the conclusion. □

Now we provide proof of Theorem 8 as follows.

Proof. Let $\kappa = 4(1+\nu)mQ/\mu_{\mathcal{M}}$. By lemma 4, 12, 11 and (B.32), we have

$$\begin{aligned}
&\Delta_d^t - \Delta_d^{t-1} + E[\Delta_p^t] - \Delta_p^{t-1} \\
\leq &\frac{-1}{1+\kappa} \left(\mathcal{L}(\boldsymbol{\alpha}^t, \boldsymbol{\lambda}^t) - \mathcal{L}(\bar{\boldsymbol{\alpha}}^t, \boldsymbol{\lambda}^t) \right) + \frac{2\eta}{\rho} (\mathcal{L}(\boldsymbol{\alpha}^t, \boldsymbol{\lambda}^t) - \mathcal{L}(\bar{\boldsymbol{\alpha}}^t, \boldsymbol{\lambda}^t)) - \frac{\eta}{\tau} \Delta_d^t.
\end{aligned} \tag{B.29}$$

Then by choosing $\eta < \frac{\rho}{2(1+\kappa)}$, we have guaranteed descent on $\Delta_p + \Delta_d$ for each GDMM iteration. By choosing $\eta \leq \frac{\rho}{4(1+\kappa)}$, we have

$$\begin{aligned}
&(\Delta_d^t + E[\Delta_p^t]) - (\Delta_d^{t-1} + \Delta_p^{t-1}) \\
\leq &\frac{-1}{2(1+\kappa)} \left(\mathcal{L}(\boldsymbol{\alpha}^t, \boldsymbol{\lambda}^t) - \mathcal{L}(\bar{\boldsymbol{\alpha}}^t, \boldsymbol{\lambda}^t) \right) - \frac{\eta}{\tau} \Delta_d^t \\
\leq &- \min \left(\frac{1}{2(1+\kappa)}, \frac{\eta}{\tau} \right) (\Delta_p^t + \Delta_d^t)
\end{aligned}$$

which then leads to the conclusion. □

The proof for Theorem 9 follows the same line of above reasoning with step (B.38) replaced by application of Lemma 7 instead of Lemma 12.

B.2 Implementation details of FMO

Indicator Factor Here we assume $\delta(y_j, \bar{y}_j)$ is constant for $\forall y_j \neq \bar{y}_j$ as in the case of Hamming error. Then we find maximizers of the 4 cases as following

(i) Visit $\boldsymbol{y}_f$ in descending order of $v(.)$ to find the first $\boldsymbol{y}_f$:$m_{if}(y_i) = 0$, $m_{jf}(y_j) = 0$.

(ii) $\forall y_j$:$m_{jf}(y_j) \neq 0$, visit y_i in descending order of $v(.)$ to find the first y_i:$m_{if}(y_i) = 0$.

(iii) $\forall y_i$:$m_{if}(y_i) \neq 0$, visit y_j in descending order of $v(.)$ to find the first y_j:$m_{jf}(y_j) = 0$.

(iv) Evaluate the gradient of Augmented Lagrangian for $\forall (y_i, y_j)$:$m_{if}(y_i) \neq 0$, $m_{jf}(y_j) \neq 0$.

Then $\boldsymbol{y}_f^*$ is returned as label ($\neq \bar{\boldsymbol{y}}_f$) of maximum gradient among the 4 cases. One can verify the above procedure considers all labels that have potential to be $\boldsymbol{y}_f^*$. The complexities for (ii)-(iv) are bounded by $O(nnz(\boldsymbol{m}_{if})nnz(\boldsymbol{m}_{jf}))$, where $nnz(\boldsymbol{m}_{jf}) \leq |\hat{\mathcal{A}}_j^t|$. When BCFW adopts sampling without replacement, we have $|\hat{\mathcal{A}}_f^t| \leq t$. In practice, as t keeps increasing, $|\hat{\mathcal{A}}_f^t|$ converges to a constant that depends on the optimal $nnz(\boldsymbol{\alpha}_f^*)$. Note $nnz(\boldsymbol{\alpha}_f^*)$ is equivalent to the number of labels $\boldsymbol{y}_f$ that attains the maximum of hinge loss (3.9), which is small in general as long as there are few labels with larger responses than the others.

Define $\mathcal{Y}_{NZ} = \{\boldsymbol{y}_f | m_{if}(y_i) \neq 0 \ \vee \ m_{jf}(y_j) \neq 0\}$ as the set of labels with messages from one of the variables involved, and $\mathcal{Y}_{Inc} = \{\boldsymbol{y}_f | \boldsymbol{y}_f \in \mathcal{Y}_{NZ} \wedge v(\boldsymbol{y}_f, \boldsymbol{x}_f) > v(\boldsymbol{y}'_f, \boldsymbol{x}_f), \forall \boldsymbol{y}'_f \notin \mathcal{Y}_{NZ}\}$ as the subset being inconsistently ranked at the top in the multimap. The complexity of step (i) is $O(|\mathcal{Y}_{Inc}|)$, where

$$|\mathcal{Y}_{Inc}| \leq \max(|\mathcal{Y}_i||\hat{\mathcal{A}}_j^t|, |\mathcal{Y}_j||\hat{\mathcal{A}}_i^t|), \tag{B.30}$$

which is sublinear to the size of factor domain $|\mathcal{Y}_f| = |\mathcal{Y}_i||\mathcal{Y}_j|$. Although the bound (B.30) is already sublinear to $|\mathcal{Y}_f|$, it is a *very loose bound*. In our experiments, we observed the average number of elements being visited at stage (i) is no more than 5 for problems of $|\mathcal{Y}_f|$ up to 10^7, presumably because the inconsistency between factors is small in real applications.

Binary-Variable Interaction Factor Similarly, the procedure for finding active factors with largest gradient is as follows:

(i) Visit $\boldsymbol{y}_f$ in descending order of $v(.)$ to find the first $\boldsymbol{y}_f$:$i \notin \mathcal{A}$, $j \notin \mathcal{A}$.

(ii) $\forall j$:$j \notin \mathcal{A}$, visit i in descending order of $v(.)$ to find the first i:$i \notin \mathcal{A}$.

(iii) $\forall i$:$i \notin \mathcal{A}$, visit j in descending order of $v(.)$ to find the first j:$j \notin \mathcal{A}$.

(iv) Compute gradient for $\forall (i, j)$:$i \in \mathcal{A}$, $j \in \mathcal{A}$.

B.3 Proofs for Theorem 10 and 11

The key in proving Theorem 10 and 11 is to establish bounds on the primal-dual progress $\Delta_p^t + \Delta_d^t - \Delta_p^{t-1} - \Delta_d^{t-1}$. As intermediate steps, the two lemmas below bound the dual-progress

$\Delta_d^t - \Delta_d^{t-1}$ and the primal-progress $\Delta_p^t - \Delta_p^{t-1}$ with respect to the primal variables $\{\boldsymbol{z}^t\}$ and the optimal primal variables $\{\bar{\boldsymbol{z}}^t\}$ at each iteration.

Lemma 8 (Dual Progress)**.** *The dual progress is upper bounded as*

$$\Delta_d^t - \Delta_d^{t-1} \leq -\eta(M\boldsymbol{z}^t)^T(M\bar{\boldsymbol{z}}^t). \tag{B.31}$$

Lemma 9 (Primal Progress)**.** *The primal progress is upper bounded as*

$$\begin{aligned}\Delta_p^t - \Delta_p^{t-1} \leq\ & \mathcal{L}(\boldsymbol{z}^{t+1}, \boldsymbol{\mu}^t) - \mathcal{L}(\boldsymbol{z}^t, \boldsymbol{\mu}^t) \\ & + \eta\|M\boldsymbol{z}^t\|^2 - \eta\langle M\boldsymbol{z}^t, M\bar{\boldsymbol{z}}^t\rangle\end{aligned}$$

By combining results of Lemma 8 and 9, we obtain an intermediate upper bound on the primal-dual progress:

$$\begin{aligned}&\Delta_d^t - \Delta_d^{t-1} + \Delta_p^t - \Delta_p^{t-1} \\ &\leq \eta\|M\boldsymbol{z}^t - M\bar{\boldsymbol{z}}^t\|^2 - \eta\|M\bar{\boldsymbol{z}}^t\|^2 \\ &+ \mathcal{L}(\boldsymbol{z}^{t+1}, \boldsymbol{\mu}^t) - \mathcal{L}(\boldsymbol{z}^t, \boldsymbol{\mu}^t)\end{aligned} \tag{B.32}$$

The following four lemmas provide upper bounds on the three sub-terms in (B.32), i.e., $\|M\boldsymbol{z}^t - M\bar{\boldsymbol{z}}^t\|^2$, $-\eta\|M\bar{\boldsymbol{z}}^t\|^2$, and $\mathcal{L}(\boldsymbol{z}^{t+1}, \boldsymbol{\mu}^t) - \mathcal{L}(\boldsymbol{z}^t, \boldsymbol{\mu}^t)$, where the bounds on the last term are algorithm-dependent and therefore are tackled by Lemma 12 and Lemma B.36 for Algorithm 8 and Algorithm 9 respectively.

Lemma 10.

$$\|M\boldsymbol{z}^t - M\bar{\boldsymbol{z}}^t\|^2 \leq \frac{2}{\rho}(\mathcal{L}(\boldsymbol{z}^t, \boldsymbol{\mu}^t) - \mathcal{L}(\bar{\boldsymbol{z}}^t, \boldsymbol{\mu}^t)). \tag{B.33}$$

Lemma 11 (*Hong and Luo 2012*)**.** *There is a constant $\tau > 0$ such that*

$$\Delta_d(\boldsymbol{\mu}) \leq \tau\|M\bar{\boldsymbol{z}}(\boldsymbol{\mu})\|^2. \tag{B.34}$$

for any $\boldsymbol{\mu}$ in the dual domain and any primal minimizer $\bar{\boldsymbol{z}}(\boldsymbol{\mu})$ satisfying (B.2)*.*

Lemma 12. *The descent amount of Augmented Lagrangian function produced by one pass of FCFW (in Algorithm 8) has*

$$\begin{aligned}&\mathcal{L}(\boldsymbol{z}^{t+1}, \boldsymbol{\mu}^t) - \mathcal{L}(\boldsymbol{z}^t, \boldsymbol{\mu}^t) \\ &\leq -\frac{m_{\mathcal{M}}}{2|\mathcal{F}|Q}(\mathcal{L}(\boldsymbol{z}^t, \boldsymbol{\mu}^t) - \mathcal{L}(\bar{\boldsymbol{z}}^t, \boldsymbol{\mu}^t))\end{aligned} \tag{B.35}$$

where $Q = \rho\|M\|^2$.

Lemma 13. *The descent amount of Augmented Lagrangian function produced by iterations of Algorithm 9 has*

$$\begin{aligned}&\mathcal{L}(\boldsymbol{z}^{t+1}, \boldsymbol{\mu}^t) - \mathcal{L}(\boldsymbol{z}^t, \boldsymbol{\mu}^t) \\ &\leq \frac{-m_1}{Q_{max}}(\mathcal{L}(\boldsymbol{z}^t, \boldsymbol{\mu}^t) - \mathcal{L}(\bar{\boldsymbol{z}}^t, \boldsymbol{\mu}^t))\end{aligned} \tag{B.36}$$

where $Q_{max} = \max_{f\in\mathcal{F}} Q_f$ and

$$m_1 := \frac{1}{\max\{16\theta_1\Delta\mathcal{L}^0, 2\theta_1(1+4L_g^2)/\rho, 6\}} \tag{B.37}$$

is the generalized strong convexity constant *for function* $\mathcal{L}(., \boldsymbol{\mu})$. *Here* $\Delta\mathcal{L}^0$ *is a bound on* $\mathcal{L}(\boldsymbol{z}^0, \boldsymbol{\mu}^t) - \mathcal{L}(\bar{\boldsymbol{z}}^0, \boldsymbol{\mu}^t)$, L_g *is local Lipschitz-continuous constant of the function* $g(\boldsymbol{x}) := \|\boldsymbol{x}\|^2$, *and* θ_1 *is the Hoffman constant depending on the geometry of optimal solution set.*

Now we are ready to prove Theorem 10 and 11.

Proof of Theorem 10. Let $\kappa = m_{\mathcal{M}}/(|\mathcal{F}|Q)$. By lemma 12 and (B.32), we have

$$\begin{aligned}
&\Delta_d^t - \Delta_d^{t-1} + \Delta_p^t - \Delta_p^{t-1} \\
\leq &\frac{-\kappa}{1+\kappa}\left(\mathcal{L}(\boldsymbol{z}^t, \boldsymbol{\mu}^t) - \mathcal{L}(\bar{\boldsymbol{z}}^t, \boldsymbol{\mu}^t)\right) \\
&+ \frac{2\eta}{\rho}(\mathcal{L}(\boldsymbol{z}^t, \boldsymbol{\mu}^t) - \mathcal{L}(\bar{\boldsymbol{z}}^t, \boldsymbol{\mu}^t)) - \eta\|M\bar{\boldsymbol{z}}^t\|^2.
\end{aligned} \tag{B.38}$$

Then by choosing $\eta < \frac{\kappa\rho}{2(1+\kappa)}$, we have guaranteed descent on $\Delta_p + \Delta_d$ for each GDMM iteration. By choosing $\eta \leq \frac{\kappa\rho}{4(1+\kappa)}$, we have

$$\begin{aligned}
&(\Delta_d^t + \Delta_p^t) - (\Delta_d^{t-1} + \Delta_p^{t-1}) \\
\leq &\frac{-\kappa}{2(1+\kappa)}\left(\mathcal{L}(\boldsymbol{z}^t, \boldsymbol{\mu}^t) - \mathcal{L}(\bar{\boldsymbol{z}}^t, \boldsymbol{\mu}^t)\right) - \eta\|M\bar{\boldsymbol{z}}^t\|^2 \\
\leq &\frac{-\kappa}{2(1+\kappa)}\Delta_d^t - \frac{\eta}{\tau}\Delta_d^t \\
\leq &-\min\left(\frac{\kappa}{2(1+\kappa)}, \frac{\eta}{\tau}\right)\left(\Delta_p^t + \Delta_d^t\right)
\end{aligned}$$

where the second inequality is from Lemma 11. We thus obtain a recursion of the form

$$\Delta_d^t + \Delta_p^t \leq \frac{1}{1 + \min(\frac{\kappa}{2(1+\kappa)}, \frac{\eta}{\tau})}\left(\Delta_d^{t-1} + \Delta_p^{t-1}\right),$$

which then leads to the conclusion. □

The proof of Theorem 11 is the same as above except that the definition of κ is changed to m_1/Q_{max} and Lemma 12 is replaced by Lemma B.36.

We provide proofs to the lemmas used as follows.

Proof of Lemma 8.

$$\begin{aligned}
\Delta_d^t - \Delta_d^{t-1} &= \mathcal{L}(\bar{\boldsymbol{z}}^{t-1}, \boldsymbol{\mu}^{t-1}) - \mathcal{L}(\bar{\boldsymbol{z}}^t, \boldsymbol{\mu}^t) \\
&\leq \mathcal{L}(\bar{\boldsymbol{z}}^t, \boldsymbol{\mu}^{t-1}) - \mathcal{L}(\bar{\boldsymbol{z}}^t, \boldsymbol{\mu}^t) \\
&= \langle \boldsymbol{\mu}^{t-1} - \boldsymbol{\mu}^t, M\bar{\boldsymbol{z}}^t\rangle \\
&= -\eta\langle M\boldsymbol{z}^t, M\bar{\boldsymbol{z}}^t\rangle
\end{aligned}$$

where the first inequality follows from the optimality of $\bar{\boldsymbol{z}}^{t-1}$ for the function $\mathcal{L}(\boldsymbol{z}, \boldsymbol{\mu}^{t-1})$ defined by $\boldsymbol{\mu}^{t-1}$, and the last equality follows from the dual update (3.32). □

Proof of Lemma 9.

$$\begin{aligned}
&\Delta_p^t - \Delta_p^{t-1} \\
=&\mathcal{L}(\boldsymbol{z}^{t+1}, \boldsymbol{\mu}^t) - \mathcal{L}(\boldsymbol{z}^t, \boldsymbol{\mu}^{t-1}) - (d(\boldsymbol{\mu}^t) - d(\boldsymbol{\mu}^{t-1})) \\
\leq&\mathcal{L}(\boldsymbol{z}^{t+1}, \boldsymbol{\mu}^t) - \mathcal{L}(\boldsymbol{z}^t, \boldsymbol{\mu}^t) + \mathcal{L}(\boldsymbol{z}^t, \boldsymbol{\mu}^t) - \mathcal{L}(\boldsymbol{z}^t, \boldsymbol{\mu}^{t-1}) \\
&+ (d(\boldsymbol{\mu}^{t-1}) - d(\boldsymbol{\mu}^t)) \\
\leq&\mathcal{L}(\boldsymbol{z}^{t+1}, \boldsymbol{\mu}^t) - \mathcal{L}(\boldsymbol{z}^t, \boldsymbol{\mu}^t) + \eta\|M\boldsymbol{z}^t\|^2 - \eta\langle M\boldsymbol{z}^t, M\bar{\boldsymbol{z}}^t\rangle
\end{aligned}$$

where the last inequality uses Lemma 8 on $d(\boldsymbol{\mu}^{t-1}) - d(\boldsymbol{\mu}^t) = \Delta_d^t - \Delta_d^{t-1}$. □

Proof of Lemma 10. Introduce

$$\tilde{\mathcal{L}}(\boldsymbol{z}, \boldsymbol{\mu}) = h(\boldsymbol{z}) + G(M\boldsymbol{z}),$$

where

$$G(M\boldsymbol{z}) = \frac{\rho}{2}\|M\boldsymbol{z}\|^2,$$

and

$$h(\boldsymbol{z}) = \langle -\boldsymbol{\theta}, \boldsymbol{z}\rangle + \langle \boldsymbol{\mu}, M\boldsymbol{z}\rangle + \boldsymbol{I}_{\boldsymbol{z}\in\mathcal{M}}.$$

Here

$$\boldsymbol{I}_{\boldsymbol{z}\in\mathcal{M}} = \begin{cases} 0 & \boldsymbol{z} \in \mathcal{M}, \\ \infty & \text{otherwise.} \end{cases}$$

As feasibility is strictly enforced during primal updates, we have

$$\tilde{\mathcal{L}}(\bar{\boldsymbol{z}}^t, \boldsymbol{\mu}^t) = \mathcal{L}(\bar{\boldsymbol{z}}^t, \boldsymbol{\mu}^t), \quad \tilde{\mathcal{L}}(\boldsymbol{z}^t, \boldsymbol{\mu}^t) = \mathcal{L}(\boldsymbol{z}^t, \boldsymbol{\mu}^t). \tag{B.39}$$

As $\bar{\boldsymbol{z}}^t$ is a critical point of $\mathcal{L}(\boldsymbol{z}, \boldsymbol{\mu}^t)$, and by definition, $\mathcal{L}(\boldsymbol{z}, \boldsymbol{\mu}^t) \le \tilde{\mathcal{L}}(\boldsymbol{z}, \boldsymbol{\mu}^t)$, we obtain,

$$0 \in \partial_{\boldsymbol{z}}\tilde{\mathcal{L}}(\bar{\boldsymbol{z}}^t, \boldsymbol{\mu}^t) = \partial h(\bar{\boldsymbol{z}}^t) + M^T\nabla G(M\bar{\boldsymbol{z}}^t).$$

Note that $h(\cdot)$ is convex, it follows that

$$h(\boldsymbol{z}^t) - h(\bar{\boldsymbol{z}}^t) \ge \langle \boldsymbol{v}, \boldsymbol{z}^t - \bar{\boldsymbol{z}}^t\rangle, \qquad \forall \boldsymbol{v} \in \partial h(\bar{\boldsymbol{z}}^t). \tag{B.40}$$

Moreover,

$$\begin{aligned} & G(M(\boldsymbol{z}^t)) - G(M(\bar{\boldsymbol{z}}^t)) \qquad\qquad (\text{B.41}) \\ &= \frac{\rho}{2}(\|M\boldsymbol{z}^t\|^2 - \|M\bar{\boldsymbol{z}}^t\|^2) \\ &= \frac{\rho}{2}(\boldsymbol{z}^t - \bar{\boldsymbol{z}}^t)^T M^T M(\boldsymbol{z}^t + \bar{\boldsymbol{z}}^t) \\ &= \rho(\boldsymbol{z}^t - \bar{\boldsymbol{z}}^t)^T M^T M\bar{\boldsymbol{z}}^t + \frac{\rho}{2}(\boldsymbol{z}^t - \bar{\boldsymbol{z}}^t)^T M^T M(\boldsymbol{z}^t - \bar{\boldsymbol{z}}^t) \\ &= \langle M^T\nabla G(M\bar{\boldsymbol{z}}^t), \boldsymbol{z}^t - \bar{\boldsymbol{z}}^t\rangle + \frac{\rho}{2}\|M\boldsymbol{z}^t - M\bar{\boldsymbol{z}}^t\|^2. \qquad (\text{B.42}) \end{aligned}$$

Combing (B.39), (B.40), and (B.42), we arrive at

$$\mathcal{L}(\boldsymbol{z}^t, \boldsymbol{\mu}^t) - \mathcal{L}(\bar{\boldsymbol{z}}^t, \boldsymbol{\mu}^t) \ge \frac{\rho}{2}\|M(\boldsymbol{z}^t) - M(\bar{\boldsymbol{z}}^t)\|^2.$$

□

Proof of Lemma 11. This is a lemma adapted from [42]. Since our primal objective (3.25) is a linear function with each block of primal variables $\boldsymbol{x}_i$ (or $\boldsymbol{y}_f$) constrained in a simplex domain, it satisfies the *assumptions A(a)—A(e)* and *A(g)* in [42]. Then Lemma 3.1 of [42] guarantees that, as long as $\|\nabla d(\boldsymbol{\mu})\|$ is always bounded, there is a constant $\tau > 0$ s.t.

$$\Delta_d(\boldsymbol{\mu}) \le \tau\|\nabla d(\boldsymbol{\mu})\|^2 = \|M\bar{\boldsymbol{z}}(\boldsymbol{\mu})\|^2$$

for all $\boldsymbol{\mu}$ in the dual domain. Note our problem satisfies the condition of bounded gradient magnitude since

$$\begin{aligned}\|\nabla d(\boldsymbol{\mu})\| &= \|M\bar{\boldsymbol{z}}(\boldsymbol{\mu})\| \leq \|M\bar{\boldsymbol{z}}(\boldsymbol{\mu})\|_1 \\ &\leq \|M\|_1\|\bar{\boldsymbol{z}}(\boldsymbol{\mu})\|_1 \leq (\max_f |\mathcal{Y}_f|)(|\mathcal{F}| + |\mathcal{V}|)\end{aligned}$$

where the last inequality is because each block of variables in $\bar{\boldsymbol{z}}(\boldsymbol{\mu})$ lie in a simplex domain. □

Proof of Lemma 12. Recall that the Augmented Lagrangian $\mathcal{L}(\boldsymbol{z}, \boldsymbol{\mu})$ is of the form

$$\mathcal{L}(\boldsymbol{z}, \boldsymbol{\mu}) = \langle -\boldsymbol{\theta} + M^T\boldsymbol{\mu}, \boldsymbol{z}\rangle + G(M\boldsymbol{z}) \ , \forall i \in \mathcal{V} \tag{B.43}$$

where M is the matrix that encodes all constraints of the form

$$M_{if}\boldsymbol{z}_f - \boldsymbol{z}_i = \left[\begin{array}{cc} M_{if} & -I_i \end{array}\right] \left[\begin{array}{c} \boldsymbol{z}_f \\ \boldsymbol{z}_i \end{array}\right] = \mathbf{0}.$$

and function $G(\boldsymbol{w}) = \frac{\rho}{2}\|\boldsymbol{w}\|^2$ is strongly convex with parameter ρ. Let

$$H(\boldsymbol{z}) := \mathcal{L}(\boldsymbol{z}, \boldsymbol{\mu}). \tag{B.44}$$

Since we are minimizing the function subject to a convex, polyhedral domain $\mathcal{M}$, by Theorem 10 of [57], we have the *generalized geometrical strong convexity* constant $m_{\mathcal{M}}$ of the form

$$m_{\mathcal{M}} := m(PWidth(\mathcal{M}))^2 \tag{B.45}$$

where $PWidth(\mathcal{M}) > 0$ is the pyramidal width of the simplex domain $\mathcal{M}$ and m is the *generalized strong convexity* constant of function (B.43) (defined by Lemma 9 of [57]). By definition of the geometric strong convexity constant, we have

$$H(\boldsymbol{z}) - H^* \leq \frac{g_{FW}^2}{2m_{\mathcal{M}}} \tag{B.46}$$

from (23) in [57], where $g_{FW} := \langle \nabla H(\boldsymbol{z}), \boldsymbol{v}_{FW} - \boldsymbol{v}_A\rangle$. $\boldsymbol{v}_{FW}$ is the greedy Frank-Wolfe (FW) direction

$$\boldsymbol{v}_{FW} := arg\min_{\boldsymbol{v}\in\mathcal{M}} \langle \nabla H(\boldsymbol{z}), \boldsymbol{v}\rangle \tag{B.47}$$

and $\boldsymbol{v}_A$ is the away direction

$$\boldsymbol{v}_A := arg\max_{\boldsymbol{v}\in\mathcal{M}} \langle \nabla \tilde{H}(\boldsymbol{z}), \boldsymbol{v}\rangle \tag{B.48}$$

where

$$\nabla_k \tilde{H}(\boldsymbol{z}) = \begin{cases} \nabla_k H(\boldsymbol{z}), & z_k \neq 0 \\ -\infty, & o.w. \end{cases}$$

Then let $m = |\mathcal{F}|$ be the number of factors. For each inner iteration s of the Fully-Corrective FW, by minimizing subproblem (3.28) w.r.t. an active set that contains the FW direction and also the away direction (by the definition (B.48)), we have, for any $\forall\gamma \in [0, 1]$,

$$H(\boldsymbol{z}^{t+1}) - H(\boldsymbol{z}^t) \leq \gamma g_{FW}^t + mQ\gamma^2. \tag{B.49}$$

Suppose the minimizer of (C.17) $\gamma^* = -\frac{g_{FW}^t}{2mQ}$ has $\gamma^* < 1$, we have

$$H(\boldsymbol{z}^{t+1}) - H(\boldsymbol{z}^t) \leq -\frac{g_{FW}^{t2}}{4mQ} \tag{B.50}$$

Otherwise, let $\gamma^* = 1$, we have

$$\begin{aligned} &H(\boldsymbol{z}^{t+1}) - H(\boldsymbol{z}^t) \\ &\leq g_{FW}^t + mQ \leq \frac{g_{FW}^t}{2} < -\frac{g_{FW}^{t2}}{2mQ} \leq -\frac{g_{FW}^{t2}}{4mQ}, \end{aligned}$$

where the second inequality holds since $-\frac{g_{FW}^t}{2Qm} \geq 1$.

Combining with the error bound (B.46), we have

$$H(\boldsymbol{z}^{t+1}) - H(\boldsymbol{z}^t) \leq -\frac{m_{\mathcal{M}}(H(\boldsymbol{z}^t) - H^*)}{2mQ}. \tag{B.51}$$

□

Proof of Lemma B.36.

For problem of the form (B.2), the optimal solution is profiled by the polyhedral set $\mathcal{S} := \{\boldsymbol{z} \mid M\boldsymbol{z} = \boldsymbol{t}^*,\ \boldsymbol{\Delta}^T\boldsymbol{z} = s^*,\ \boldsymbol{z} \in \mathcal{M}\}$ for some $\boldsymbol{t}^*$, s^*. Denoting $\bar{\boldsymbol{z}} := \Pi_{\mathcal{S}}(\boldsymbol{z})$, we can bound the distance of any feasible point $\boldsymbol{z}$ to its projection $\Pi_{\mathcal{S}}(\boldsymbol{z})$ to set $\mathcal{S}$ by

$$\begin{aligned} \|\bar{\boldsymbol{z}} - \boldsymbol{z}\|_{2,1}^2 &= (\sum_{f\in\mathcal{F}} \|\bar{\boldsymbol{z}}_f - \boldsymbol{z}_f\|_2)^2 \\ &\leq \theta_1 \left(\|M\boldsymbol{z} - \boldsymbol{t}^*\|^2 + \|\boldsymbol{\Delta}^T\boldsymbol{z} - s^*\|^2\right) \end{aligned} \tag{B.52}$$

where θ_1 is a constant depending on the set $\mathcal{S}$, using the Hoffman's inequality [41].

Then for each iteration t of the Algorithm 9, consider the descent amount produced by the update w.r.t. the selected factor satisfying (3.34). We have

$$\begin{aligned} &H(\boldsymbol{z}^{t+1}) - H(\boldsymbol{z}^t) \\ &\leq \min_{\boldsymbol{z}_{f^*}^t + \boldsymbol{d}_{f^*} \in \Delta_{f^*}} \langle \nabla_{\boldsymbol{z}_{f^*}} H, \boldsymbol{d}_{f^*} \rangle + \frac{Q_{\max}}{2} \|\boldsymbol{d}_{f^*}\|^2 \\ &= \min_{\boldsymbol{z}^t + \boldsymbol{d} \in \mathcal{M}} \sum_{f\in\mathcal{F}} \langle \nabla_{\boldsymbol{z}_f} H, \boldsymbol{d}_f \rangle + \frac{Q_{\max}}{2} \left(\sum_{f\in\mathcal{F}} \|\boldsymbol{d}_f\|\right)^2 \end{aligned} \tag{B.53}$$

where the second equality is from the definition (3.34) of f^*.

Then we have

$$
\begin{aligned}
&H(\boldsymbol{z}^{t+1})] - H(\boldsymbol{z}^t) \\
&\leq \min_{\boldsymbol{z}^t+\boldsymbol{d}\in\mathcal{M}} \left(\sum_{f\in\mathcal{F}} \langle \nabla_{\boldsymbol{z}_f} H, \boldsymbol{d}_f \rangle + \frac{Q_{\max}}{2} \left(\sum_{f\in\mathcal{F}} \|\boldsymbol{d}_f\| \right)^2 \right) \\
&\leq \min_{\boldsymbol{z}^t+\boldsymbol{d}\in\mathcal{M}} H(\boldsymbol{z}^t + \boldsymbol{d}) - H(\boldsymbol{z}^t) + \frac{Q_{\max}}{2} \left(\sum_{f\in\mathcal{F}} \|\boldsymbol{d}_f\| \right)^2 \\
&\leq \min_{\beta\in[0,1]} H(\boldsymbol{z}^t + \beta(\bar{\boldsymbol{z}}^t - \boldsymbol{z}^t)) - H(\boldsymbol{z}^t) \\
&\quad + \frac{Q_{\max}\beta^2}{2} \left(\sum_{f\in\mathcal{F}} \|\bar{\boldsymbol{z}}_f^t - \boldsymbol{z}_f^t\| \right)^2 \\
&\leq \min_{\beta\in[0,1]} \beta(H(\bar{\boldsymbol{z}}^t) - H(\boldsymbol{z}^t)) + \frac{Q_{\max}\beta^2}{2} \|\bar{\boldsymbol{z}}^t - \boldsymbol{z}^t\|_{2,1}^2
\end{aligned}
\tag{B.54}
$$

where $\bar{\boldsymbol{z}}^t = \Pi_{\mathcal{S}}(\boldsymbol{z}^t)$ is the projection of $\boldsymbol{z}^t$ to the optimal solution set $\mathcal{S}$. The second and last inequality is due to convexity, and the third inequality is due to a confinement of optimization domain. Then let L_g be the local Lipschitz-continuous constant of function $G(M\boldsymbol{z}) = \frac{\rho}{2}\|M\boldsymbol{z}\|^2$ in the bounded domain of $M\boldsymbol{z}$. We discuss two cases in the following.

Case 1: $4L_g^2\|M\boldsymbol{z}^t - \boldsymbol{t}^*\|^2 < (\boldsymbol{\Delta}^T\boldsymbol{z}^t - s^*)^2$.

In this case, we have

$$
\begin{aligned}
\|\boldsymbol{z}^t - \bar{\boldsymbol{z}}^t\|_{2,1}^2 &\leq \theta_1(\|M\boldsymbol{z}^t - \boldsymbol{t}^*\|^2 + (\boldsymbol{\Delta}^T\boldsymbol{z}^s - s^*)^2) \\
&\leq \theta_1(\frac{1}{L_g^2} + 1)(\boldsymbol{\Delta}^T\boldsymbol{z}^t - s^*)^2 \\
&\leq 2\theta_1(\boldsymbol{\Delta}^T\boldsymbol{z}^t - s^*)^2,
\end{aligned}
\tag{B.55}
$$

and

$$
|\boldsymbol{\Delta}^T\boldsymbol{z}^t - s^*| \geq 2L_g\|M\boldsymbol{z}^t - \boldsymbol{t}^*\| \geq 2|G(M\boldsymbol{z}^t) - G(\boldsymbol{t}^*)|
$$

by the definition of Lipschitz constant L_g. Note $\boldsymbol{\Delta}^T\boldsymbol{z}^t - s^*$ is non-negative since otherwise, $H(\boldsymbol{z}^t) - H^* = G(M\boldsymbol{z}^t) - G(\boldsymbol{t}^*) + (\boldsymbol{\Delta}^T\boldsymbol{z}^t - s^*) \leq |G(M\boldsymbol{z}^t) - G(\boldsymbol{t}^*)| - |\boldsymbol{\Delta}^T\boldsymbol{z}^t - s^*| \leq -\frac{1}{2}|\boldsymbol{\Delta}^T\boldsymbol{z}^t - s^*| < 0$, which leads to contradiction. Therefore, we have

$$
\begin{aligned}
&H(\boldsymbol{z}^t) - H^* \\
&= G(M\boldsymbol{z}^t) - G(\boldsymbol{t}^*) + (\boldsymbol{\Delta}^T\boldsymbol{z}^t - s^*) \\
&\geq -|G(M\boldsymbol{z}^t) - G(\boldsymbol{t}^*)| + (\boldsymbol{\Delta}^T\boldsymbol{z}^t - s^*) \\
&\geq \frac{1}{2}(\boldsymbol{\Delta}^T\boldsymbol{z}^t - s^*).
\end{aligned}
\tag{B.56}
$$

Combining (B.54), (B.55) and (B.56), we have

$$\begin{aligned}
&H(\boldsymbol{z}^{t+1}) - H(\boldsymbol{z}^t) \\
&\leq \min_{\beta\in[0,1]} -\frac{\beta}{2}(\boldsymbol{\Delta}^T\boldsymbol{z}^t - s^*) + \frac{2Q_{max}\theta_1\beta^2}{2}(\boldsymbol{\Delta}^T\boldsymbol{z}^t - s^*)^2 \\
&= \begin{cases} -1/(16Q_{\max}\theta_1) & , \ 1/(4\rho\theta_1(\boldsymbol{\Delta}^T\boldsymbol{z}^t - s^*)) \leq 1 \\ -\frac{1}{4}(\boldsymbol{\Delta}^T\boldsymbol{\alpha}^s - s^*) & , \ o.w. \end{cases}
\end{aligned}$$

Furthermore, we have

$$-\frac{1}{16Q_{max}\theta_1} \leq -\frac{1}{16Q_{max}\theta_1(H^0 - H^*)}\left(H(\boldsymbol{z}^t) - H^*\right)$$

where $H^0 = H(\boldsymbol{z}^0)$, and

$$-\frac{1}{4}(\boldsymbol{\Delta}^T\boldsymbol{z}^t - s^*) \leq -\frac{1}{6}(H(\boldsymbol{z}^t) - H^*)$$

since $H(\boldsymbol{z}^t) - H^* \leq |G(M\boldsymbol{z}^t) - G(\boldsymbol{t}^*)| + \boldsymbol{\Delta}^T\boldsymbol{z}^t - s^* \leq \frac{3}{2}(\boldsymbol{\Delta}^T\boldsymbol{z}^t - s^*)$. In summary, for Case 1 we obtain

$$H(\boldsymbol{z}^{t+1})] - H^* \leq (1 - \frac{m_0}{Q_{max}})\left(H(\boldsymbol{z}^t) - H^*\right) \tag{B.57}$$

where

$$m_0 = \frac{1}{\max\left\{16\theta_1(H^0 - H^*)\ ,\ 6\right\}}. \tag{B.58}$$

Case 2: $4L_g^2\|M\boldsymbol{z}^t - \boldsymbol{t}^*\|^2 \geq (\boldsymbol{\Delta}^T\boldsymbol{z}^t - s^*)^2$.
In this case, we have

$$\|\bar{\boldsymbol{z}}^t - \boldsymbol{z}^t\|^2 \leq \theta_1\left(1 + 4L_g^2\right)\|M\boldsymbol{z}^t - \boldsymbol{t}^*\|^2, \tag{B.59}$$

and by strong convexity of $G(.)$,

$$\begin{aligned}
&H(\boldsymbol{z}^t) - H^* \geq \\
&\boldsymbol{\Delta}^T(\boldsymbol{z}^t - \boldsymbol{z}^*) + \nabla G(t^*)^T M(\bar{\boldsymbol{z}}^t - \boldsymbol{z}^t) + \frac{\rho}{2}\|M\boldsymbol{z}^t - \boldsymbol{t}^*\|^2.
\end{aligned}$$

Now let $h(\boldsymbol{\alpha})$ be a function that takes value 0 when $\boldsymbol{z}$ is feasible and takes value ∞ otherwise. Adding inequality $0 = h(\boldsymbol{z}^t) - h(\bar{\boldsymbol{z}}^t) \geq \langle\boldsymbol{\sigma}^*, \boldsymbol{z}^t - \bar{\boldsymbol{z}}^t\rangle$ for some $\boldsymbol{\sigma}^* \in \partial h(\bar{\boldsymbol{z}}^t)$ to the above gives

$$H(\boldsymbol{z}^t) - H^* \geq \frac{\rho}{2}\|M\boldsymbol{z}^t - \boldsymbol{t}^*\|^2 \tag{B.60}$$

since $\boldsymbol{\sigma}^* + \boldsymbol{\Delta} + \nabla G(\boldsymbol{t}^*)^T M = \boldsymbol{\sigma}^* + \nabla H(\boldsymbol{z}^t) = 0$. Combining (B.54), (B.59), and (B.60), we obtain

$$\begin{aligned}
&H(\boldsymbol{z}^{t+1}) - H(\boldsymbol{z}^t) \\
&\leq \min_{\beta\in[0,1]} -\beta(H(\boldsymbol{z}^t) - H^*) + \frac{\theta_1(1 + 4L_g^2)Q_{max}\beta^2}{2\rho}\left(H(\boldsymbol{z}^t) - H^*\right) \\
&= -\frac{\rho}{2\theta_1(1 + 4L_g^2)Q_{max}}\left(H(\boldsymbol{z}^t) - H^*\right)
\end{aligned} \tag{B.61}$$

Combining results of Case 1 (B.57) and Case 2 (B.61), we have

$$H(\boldsymbol{z}^{t+1}) - H(\boldsymbol{z}^t) \leq -\frac{m_1}{Q_{\max}}(H(\boldsymbol{z}^t) - H^*), \tag{B.62}$$

where

$$m_1 = \frac{1}{\max\{16\theta_1\Delta\mathcal{L}^0, 2\theta_1(1+4L_g^2)/\rho, 6\}}$$

This leads to the conclusion.

□

Appendix C

Appendix for Chapter 4

C.1 Comparison of Runtime Complexity

The proposed LatentLasso algorithm runs significantly faster than other methods in our experiments. For example, on the Syn1 dataset (N=1000, D=1000, K=35), the runtime of LatentLasso is 398s, while MCMC, Variational, MF-Binary and BP-Means all take more than 10000s to obtain their best results reported in the Figures (and the implementation of Spectral Method we obtained from the authors has memory requirement that restricts $K < 14$). On the real data sets, we report only up to K=50 because most of the compared methods already took one day to train.

Table C.1: Comparison of Time Complexity. (T denotes number of iterations)

MCMC	Variational	MF-Binary
$(NK^2D)T$	$(NK^2D)T$	$(NK)2^K$
BP-Means	Spectral	LatentLasso
$(NK^3D)T$	$ND + K^5log(K)$	$(ND + K^2D)T$

The complexity of each algorithm can be summarized in Table C.1. The reason for the smaller runtime of LatentLasso is due to the decoupling of factor ND from the factor related to K, where the factor $O(ND)$ comes from the cost of solving a MAX-CUT-like problem using the method of [10] or [102], while the factor $O(K^2D)$ comes from the cost of solving a least-square problem given by (4.11) with the maintenance cost of Z^TZ amortized.

C.2 Proof for Theorem 14

Let $L(M)$ be a smooth function such that $\nabla L(M)$ is Lipschitz-continuous with parameter β, that is,

$$L(M') - L(M) - \langle \nabla L(M), M' - M \rangle \leq \frac{\beta}{2}\|M' - M\|_F^2.$$

Then

$$\nabla_j f(c) = z_j^T \nabla L(M) z_j$$

is Lipschitz-continuous with parameter γ, which is of order $O(1)$ when loss function $L(.)$ is an empirical average normalized by ND.

Let $\mathcal{A}$ be the active set before adding $\hat{j}$. Consider the descent amount produced by minimizing $F(c)$ w.r.t. the $c_{\hat{j}}$ given that $0 \in \partial_j F(c)$ for all $j \in \mathcal{A}$ due to the subproblem solved in the previous iteration. Let $j = \hat{j}$, for any η_j we have

$$\begin{aligned} F(c + \eta_j e_j) - F(c) &\leq \nabla_j f(c)\eta_j + \lambda|\eta_j| + \frac{\gamma}{2}\eta_j^2 \\ &\leq \mu\nabla_{j^*} f(c)\eta_j + \lambda|\eta_j| + \frac{\gamma}{2}\eta_j^2 \end{aligned}$$

Minimize w.r.t η_j gives

$$\begin{aligned} &\min_{\eta_j} \; F(c + \eta_j e_j) - F(c) \\ &\leq \min_{\eta_j} \; \mu\nabla_{j^*} f(c)\eta_j + \lambda|\eta_j| + \frac{\gamma}{2}\eta_j^2 \\ &= \min_{\eta_k : k \notin \mathcal{A}} \sum_{k \notin \mathcal{A}} \left(\mu\nabla_k f(c)\eta_k + \lambda|\eta_k| \right) + \frac{\gamma}{2} \left(\sum_{k \notin \mathcal{A}} |\eta_k| \right)^2 \\ &\leq \min_{\eta_k : k \notin \mathcal{A}} \; \mu \sum_{k \notin \mathcal{A}} \left(\nabla_k f(c)\eta_k + \lambda|\eta_k| \right) + \frac{\gamma}{2} \left(\sum_{k \notin \mathcal{A}} |\eta_k| \right)^2 \\ &+ (1 - \mu)\lambda \sum_{k \notin \mathcal{A}} |\eta_k| \end{aligned}$$

where the last equality is justified later in Lemma 14. For $k \in \mathcal{A}$, we have

$$0 = \min_{\eta_k : k \in \mathcal{A}} \; \mu \sum_{k \in \mathcal{A}} \left(\nabla_k f(c)\eta_k + \lambda|c_k + \eta_k| - \lambda|c_k| \right)$$

Combining cases for $k \notin \mathcal{A}$ and $k \in \mathcal{A}$, we can obtain a global estimate of descent amount

compared to some optimal solution x^* as follows

$$\begin{aligned}
&\min_{\eta_{\hat{j}}} \; F(c+\eta_{\hat{j}}e_{\hat{j}}) - F(c) \\
&\leq \min_{\eta} \; \mu\Big(\langle \nabla f(c), \eta\rangle + \lambda\|c+\eta\|_1 - \lambda\|c\|_1\Big) \\
&+ \frac{\gamma}{2}\left(\sum_{k\notin\mathcal{A}}|\eta_k|\right)^2 + (1-\mu)\lambda\sum_{k\notin\mathcal{A}}|\eta_k| \\
&\leq \min_{\eta} \; \mu\Big(F(c+\eta) - F(c)\Big) + \frac{\gamma}{2}\left(\sum_{k\notin\mathcal{A}}|\eta_k|\right)^2 \\
&+ (1-\mu)\lambda\sum_{k\notin\mathcal{A}}|\eta_k| \\
&\leq \min_{\alpha\in[0,1]} \; \mu\Big(F(c+\alpha(c^*-c)) - F(c)\Big) + \frac{\alpha\gamma}{2}\|c^*\|_1^2 \\
&+ \alpha(1-\mu)\lambda\|c^*\|_1 \\
&\leq \min_{\alpha\in[0,1]} \; -\alpha\mu\Big(F(c) - F(c^*)\Big) + \frac{\alpha^2\gamma}{2}\|c^*\|_1^2 \\
&+ \alpha(1-\mu)\lambda\|c^*\|_1.
\end{aligned}$$

It means we can always choose an α small enough to guarantee descent if

$$F(c) - F(c^*) > \frac{(1-\mu)}{\mu}\lambda\|c^*\|_1. \tag{C.1}$$

In addition, for

$$F(c) - F(c^*) \geq \frac{2(1-\mu)}{\mu}\lambda\|c^*\|_1, \tag{C.2}$$

we have

$$\begin{aligned}
&\min_{\eta_{\hat{j}}} \; F(c+\eta_{\hat{j}}e_{\hat{j}}) - F(c) \\
&\leq \min_{\alpha\in[0,1]} \; -\frac{\alpha\mu}{2}\Big(F(c) - F(c^*)\Big) + \frac{\alpha^2\gamma}{2}\|c^*\|_1^2.
\end{aligned}$$

Minimizing w.r.t. to α gives the convergence guarantee

$$F(c^t) - F(c^*) \leq \frac{2\gamma\|c^*\|_1^2}{\mu^2}\frac{1}{t}.$$

for any iterate with $F(c^t) - F(c^*) \geq \frac{2(1-\mu)}{\mu}\lambda\|c^*\|_1$.

Lemma 14.

$$\min_{\eta_j} \; \mu\nabla_{j^*} f(c)\eta_j + \lambda|\eta_j| + \frac{\gamma}{2}\eta_j^2 \tag{C.3}$$

$$= \min_{\eta_k : k\notin\mathcal{A}} \sum_{k\notin\mathcal{A}}\Big(\mu\nabla_k f(c)\eta_k + \lambda|\eta_k|\Big) + \frac{\gamma}{2}\left(\sum_{k\notin\mathcal{A}}|\eta_k|\right)^2 \tag{C.4}$$

Proof. The minimization (C.18) is equivalent to

$$\begin{aligned} \min_{\eta_k:k\notin\mathcal{A}} \quad & \sum_{k\notin\mathcal{A}}\Big(\mu\nabla_k f(c)\eta_k\Big) \\ s.t. \quad & \left(\sum_{k\notin\mathcal{A}}|\eta_k|\right)^2 \le C_1 \\ & \sum_{k\notin\mathcal{A}}|\eta_k| \le C_2 \end{aligned}$$

and therefore is equivalent to

$$\begin{aligned} \min_{\eta_k:k\notin\mathcal{A}} \quad & \mu\sum_{k\notin\mathcal{A}}\nabla_k f(c)\eta_k \\ s.t. \quad & \sum_{k\notin\mathcal{A}}|\eta_k| \le \min\{\sqrt{C_1}, C_2\} \end{aligned}$$

which is a linear objective subject to a convex set and thus always has solution that lies on the corner point with only one non-zero coordinate η_{j^*}, which then gives the same minimum as (C.17). □

C.3 Proof of Theorem 15

Lemma 15. *Let $\mathcal{A}^* \in [\bar{K}]$ be a support set and $c^* := arg\min_{c:supp(c)=\mathcal{A}^*} F(c)$. Suppose $F(c)$ is strongly convex on $\mathcal{A}^*$ with parameter β. We have*

$$\|c^*\|_1 \le \sqrt{\frac{2\|c^*\|_0(F(0)-F(c^*))}{\beta}}. \tag{C.5}$$

Proof. Since $supp(c^*) = \mathcal{A}^*$, and c^* is optimal when restricted on the support, we have $\langle \boldsymbol{\eta}, c^*\rangle = 0$ for some $\boldsymbol{\eta} \in \partial F(c^*)$. And since $F(c)$ is strongly convex on the support $\mathcal{A}^*$ with parameter β, we have

$$\begin{aligned} F(0) - F(c^*) &= F(0) - F(c^*) - \langle \boldsymbol{\eta}, 0 - c^*\rangle \\ &\ge \frac{\beta}{2}\|c^* - 0\|_2^2, \end{aligned}$$

which gives us

$$\|c^*\|_2^2 \le \frac{2(F(0)-F(c^*))}{\beta}.$$

Combining above with the fact for any c, $\|c\|_1^2 \le \|c\|_0\|c\|_2^2$, we obtain the result. □

Since $F(0) - F(c^*) \le \frac{1}{2N}\sum_{i=1}^N y_i^2 \le 1$, from Theorem (14) and (C.5), we have

$$F(c^T) - F(c^*) \le \frac{4\gamma\|c^*\|_0}{\beta\mu^2}\left(\frac{1}{T}\right) + \frac{2(1-\mu)\lambda}{\mu}\sqrt{\frac{2\|c^*\|_0}{\beta}}. \tag{C.6}$$

for any $c^* := arg\min_{c:supp(c)=\mathcal{A}^*} F(c)$.

C.4 Proof of Theorem 16

Before delving into the analysis of the *Latent Feature Lasso* method, we first investigate what one can achieve in terms of the risk defined in (4.1) if the *combinatorial version of objective* is solved. Let

$$f(x;W) := \min_{z\in\{0,1\}^K} \frac{1}{2}\|x - W^T z\|^2.$$

Suppose we can obtain solution $\hat{W}$ to the following empirical risk minimization problem:

$$\hat{W} := \underset{W\in\mathbb{R}^{K\times D}:\|W\|_F\le R}{argmin} \quad \frac{1}{N}\sum_{i=1}^{N} f(x_i;W). \tag{C.7}$$

Then the following theorem holds.

Theorem 19. *Let W^* be the minimizer of risk* (4.1) *and $\hat{W}$ be the empirical risk minimizer* (C.7)*. Then*

$$\begin{aligned}&\mathcal{E}[f(x;\hat{W})] - \mathcal{E}[f(x;W^*)]\\ &\le \frac{3}{N} + \sqrt{\frac{DK\log(4R^2KN)}{2N} + \frac{1}{2N}\log(\frac{1}{\rho})}\end{aligned}$$

with probability $1-\rho$.

Proof Sketch. Let $\mathcal{E}_N[f(x,W)]$ denote the empirical risk. We have

$$\begin{aligned}&\mathcal{E}[f(x;\hat{W})] - \mathcal{E}[f(x;W^*)]\\ &\le 2\left(\sup_{W\in\mathbb{R}^{K\times D}:\|W\|_F\le R} |\mathcal{E}[f(x;W)] - \mathcal{E}_N[f(x;W)]|\right)\end{aligned} \tag{C.8}$$

from error decomposition and $\mathcal{E}_N[f(x,\hat{W})] \le \mathcal{E}_N[f(x,W^*)]$. Then by introducing a δ-net $\mathcal{N}(\delta)$ with covering number $|\mathcal{N}(\delta)| = \left(\frac{4R}{\delta}\right)^{DK}$, we have $\|\tilde{W} - W\|_F \le \delta$ for some $\tilde{W}\in\mathcal{N}(\delta)$ and

$$\begin{aligned}&P\left(\sup_{\tilde{W}\in\mathcal{N}(\delta)}\left|\mathcal{E}[f(x;\tilde{W})] - \mathcal{E}_N[f(x;\tilde{W})]\right| \le \epsilon\right)\\ &\ge 1 - \left(\frac{4R}{\delta}\right)^{DK}\exp(-2N\epsilon^2).\end{aligned} \tag{C.9}$$

Then since

$$\begin{aligned}&2(f(x,\tilde{W}) - f(x,W)) \le \|x - \tilde{W}^T z^*\|^2 - \|x - W^T z^*\|^2\\ &= z^{*T}(W - \tilde{W})x + \langle \tilde{W}\tilde{W}^T - WW^T, z^* z^{*T}\rangle\\ &\le \|z^*\|_2\|W - \tilde{W}\|_F + 2R\|\tilde{W} - W\|_F\|z^*\|_2^2\\ &\le 3RK\|\tilde{W} - W\|_F,\end{aligned}$$

we have

$$\begin{aligned}
&\sup_{W:\|W\|_F\leq R}\left|\mathcal{E}[f(x;W)]-\mathcal{E}_N[f(x;W)]\right| \\
&\leq (3RK\delta)+\sup_{\tilde{W}\in\mathcal{N}(\delta)}\left|\mathcal{E}[f(x;\tilde{W})]-\mathcal{E}_N[f(x;\tilde{W})]\right| \\
&\leq 3RK\delta+\sqrt{\frac{DK}{2N}\log(\frac{4R}{\delta})+\frac{1}{2N}\log(\frac{1}{\rho})}
\end{aligned} \tag{C.10}$$

with probability $1-\rho$. Choosing $\delta = 1/(RKN)$ yields the result. □

Now we establish the proof of Theorem (16) for bounding risk of the *Latent Feature Lasso* estimator.

Proof. Let $Z^* \in arg\min_{Z\in\{0,1\}^{NK}} \frac{1}{N}\|X - ZW^*\|_F^2$ and $\mathcal{S}^*$ be the set of column index of $\mathbf{Z}$ with the same 0-1 patterns to columns in Z^*. Let c^* be indicator vector with $c_k^* = 1, k \in \mathcal{S}^*$ and $c_k^* = 0, k \notin \mathcal{S}^*$. We have

$$F(\bar{c}) \leq F(c^*) \leq \mathcal{E}_N[f(x;W^*)] + \frac{\tau}{2}\|W^*\|_F^2 + \lambda\|c^*\|_1 \tag{C.11}$$

where $\bar{c} \in \underset{c:supp(c)=\mathcal{S}^*}{argmin} F(c)$. Then let (c, W) with $supp(c) = \hat{\mathcal{S}}$ be the output obtained from running T iterations of the greedy algorithm, we have

$$\begin{aligned}
&\mathcal{E}_N[f(x, D_cW)] + \frac{\tau}{2}\|W\|_F^2 + \lambda\|c\|_1 \\
&= \frac{1}{2N}\sum_{i=1}^{N}\min_{z\in\{0,1\}^{\|c\|_0}}\|x_i - W^TD_c^Tz\|^2 + \frac{\tau}{2}\|W\|_F^2 + \lambda\|c\|_1 \\
&\leq F(c)
\end{aligned} \tag{C.12}$$

Combining (C.17), (C.18) and (C.6), we obtain a bound on the *bias* and *optimization error* of the Latent Feature Lasso estimator

$$\begin{aligned}
&\mathcal{E}_N[f(x, D_cW)] \leq F(c) \leq \mathcal{E}_N[f(x;W^*)] \\
&+ \underbrace{\frac{\tau}{2}\|W^*\|^2 + \lambda K}_{\text{regularize bias}} + \underbrace{\frac{2\gamma K}{\beta}\left(\frac{1}{T}\right) + \sqrt{\frac{2(1-\mu)K}{\mu\beta}}\lambda}_{\text{optimization error}}
\end{aligned} \tag{C.13}$$

To bound the estimation error, notice that the matrix $\hat{W} := D_cW$ is $\hat{K} \times D$ with $\hat{K} \leq T$. Furthermore, the descent condition $F(c) \leq F(0)$ guarantees that

$$\frac{\tau}{2}\|W\|_F^2 + \lambda\|c\|_1 \leq \frac{1}{N}\|X - 0\|^2 \leq 1$$

and thus $\|W\|_F^2 \leq 1/\tau$, $\|c\|_1 \leq 1/\lambda$.

Let $\mathcal{W}(T,\lambda,\tau) := \{\hat{W} \in (\mathbb{R}^{T\times D}) \mid \|\hat{W}\|_F \le \sqrt{1/(\lambda\tau)}\}$. We have

$$\sup_{(c,W)\in\mathcal{W}(T,\lambda,\tau)} \mathcal{E}[f(x;\hat{W})] - \mathcal{E}_N[f(x,\hat{W})] \le \sqrt{\frac{DT\log(4TN/(\tau\lambda))}{2N} + \frac{1}{2N}\log(\frac{1}{\rho})}$$

with probability $1-\rho$ through the same argument as in the case of combinatorial objective (C.10). Combining the above *estimation error* with the *bias* and *optimization error* in (C.13), we have

$$\begin{aligned}&\mathcal{E}[f(x;W)] - \mathcal{E}[f(x;W^*)]\\ &\le \frac{\tau}{2}R^2 + \lambda K + \frac{2\gamma K}{\beta T} + \sqrt{\frac{2(1-\mu)K}{\mu\beta}}\lambda\\ &\quad + \sqrt{\frac{DT\log(4TN/(\tau\lambda))}{2N} + \frac{1}{2N}\log(\frac{1}{\rho})}\end{aligned}$$

Choosing $T = \frac{2\gamma K}{\beta}(\frac{1}{\epsilon})$, $\lambda = \tau = \frac{1}{\sqrt{N}}$ and $N \gtrsim \frac{DT}{\epsilon^2} = \frac{DK}{\epsilon^3}$ gives the result. □

C.5 Proof for Lemma 15

Since $supp(\boldsymbol{c}^*) = \mathcal{A}^*$, and $\boldsymbol{c}^*$ is optimal when restricted on the support, we have $\langle \boldsymbol{\eta}, \boldsymbol{c}^*\rangle = 0$ for some $\boldsymbol{\eta} \in \partial F(\boldsymbol{c}^*)$. And since $F(\boldsymbol{c})$ is strongly convex on the support $\mathcal{A}^*$ with parameter β, we have

$$\begin{aligned}F(0) - F(\boldsymbol{c}^*) &= F(0) - F(\boldsymbol{c}^*) - \langle \boldsymbol{\eta}, 0 - \boldsymbol{c}^*\rangle\\ &\ge \frac{\beta}{2}\|\boldsymbol{c}^* - 0\|_2^2,\end{aligned}$$

which gives us

$$\|\boldsymbol{c}^*\|_2^2 \le \frac{2(F(0) - F(\boldsymbol{c}^*))}{\beta}.$$

Combining above with the fact for any $\boldsymbol{c}$, $\|\boldsymbol{c}\|_1^2 \le \|\boldsymbol{c}\|_0\|\boldsymbol{c}\|_2^2$, we obtain the result.

C.6 Proof for Theorem 18

Lemma 16. *Let $r(W)$ and $r_N(W)$ be the risk* (4.21) *and the empirical risk respectively, we have*

$$\sup_{W\in\mathbb{R}^{K\times D}:\|W\|_F\le R} |r(W) - r_N(W)| \le \sqrt{\frac{2DK\log(4RKN)}{N} + \frac{1}{N}\log(\frac{1}{\rho})}$$

with probability $1-\rho$.

Proof. Since $\min_{\boldsymbol{z}\in\{0,1\}^N} \frac{1}{2}(y - \boldsymbol{z}^\mathcal{T} W\boldsymbol{x})^2 \le |y|^2 \le 1$ for a given W, by Hoeffding inequality,

$$\begin{aligned} &P\left(|r_N(W) - r(W)| \ge \epsilon\right) \\ &\le \exp(-2N\epsilon^2). \end{aligned}$$

Let $\mathcal{N}(\delta)$ be a δ-covering of the set $\mathcal{W} := \{W \in \mathbb{R}^{K\times D} \mid \|W\|_F \le R\}$ with $|\mathcal{N}(\delta)| \le \left(\frac{4R}{\delta}\right)^{DK}$. Then for any $W \in \mathcal{W}$, we have $\tilde{W} \in \mathcal{N}(\delta)$ with $\|W - \tilde{W}\| \le \delta$. Applying a union bound, we have

$$\begin{aligned} &P\left(\sup_{\tilde{W}\in\mathcal{N}(\delta)} |r_N(\tilde{W}) - r(\tilde{W})| \ge \epsilon\right) \\ &\le \left(\frac{4R}{\delta}\right)^{DK} \exp(-2N\epsilon^2). \end{aligned} \tag{C.14}$$

Then for $\Delta W := W - \tilde{W}$ satisfying $\|\Delta W\| \le \delta$, we can bound the difference of square loss of W and $\tilde{W}$ by

$$\begin{aligned} &\min_{\boldsymbol{z}\in\{0,1\}^K} \frac{1}{2}(y - \boldsymbol{z}^\mathcal{T} W\boldsymbol{x})^2 - \min_{\boldsymbol{z}\in\{0,1\}^K} \frac{1}{2}(y - \boldsymbol{z}^\mathcal{T} \tilde{W}\boldsymbol{x})^2 \\ &\le \frac{1}{2}(y - \tilde{\boldsymbol{z}}^\mathcal{T} W\boldsymbol{x})^2 - \frac{1}{2}(y - \tilde{\boldsymbol{z}}^\mathcal{T} \tilde{W}\boldsymbol{x})^2 \\ &\le \|\Delta W\|_F\|\tilde{\boldsymbol{z}}\| + 2R\|\tilde{\boldsymbol{z}}\|^2\|\Delta W\|_F \le 3RK\epsilon \end{aligned} \tag{C.15}$$

where $\tilde{\boldsymbol{z}} = arg\min_{\boldsymbol{z}\in\{0,1\}^K} \frac{1}{2}(y - \boldsymbol{z}^\mathcal{T} W\boldsymbol{x})^2$ and we used the fact that $\|\boldsymbol{x}\| \le 1$ and $|y| \le 1$. By symmetry, we have

$$\left|\min_{\boldsymbol{z}\in\{0,1\}^K} \frac{1}{2}(y - \boldsymbol{z}^\mathcal{T} \tilde{W}\boldsymbol{x})^2 - \min_{\boldsymbol{z}\in\{0,1\}^K} \frac{1}{2}(y - \boldsymbol{z}^\mathcal{T} W\boldsymbol{x})^2\right| \le 3RK\epsilon$$

. Combining (C.14) with (C.15), we have

$$\begin{aligned} &\sup_{W\in\mathcal{W}} |r_N(W) - r_N(W)| \\ &\le 6RK\delta + \sqrt{\frac{DK}{2N}\log(\frac{4R}{\delta}) + \frac{1}{2N}\log(\frac{1}{\rho})}. \end{aligned} \tag{C.16}$$

with probability $1 - \rho$. Setting $\delta = 1/(6RK\sqrt{N})$ and apply Jennen's inequality gives the result. □

Then the following gives the proof for Theorem 18.

Proof. Let $\bar{\boldsymbol{z}}_i = arg\min_{\boldsymbol{z}_i\in\{0,1\}^K} (y_i - \boldsymbol{z}_i^\mathcal{T} \bar{W}\boldsymbol{x}_i)^2$ for $i \in [N]$. Denote $\bar{Z}$ as the $N \times K$ matrix stacked from $(\bar{\boldsymbol{z}}_i^\mathcal{T})_{i=1}^N$. Let $\{\bar{\boldsymbol{z}}^k\}_{k=1}^K$ be the columns of $\bar{Z}$ and $\bar{\mathcal{A}}$ be the indexes of atoms in the atomic set (4.24) that have the same 0-1 patterns to those columns. Denote $\bar{\boldsymbol{c}}$ as the coefficient vector with $\bar{c}_k = 1$ for $k \in \bar{\mathcal{A}}$ and $\bar{c}_k = 0$ for $k \notin \bar{\mathcal{A}}$. By the definition of $F(\boldsymbol{c})$, we have

$$F(\bar{\boldsymbol{c}}) \le r_N(\bar{W}) + \frac{\tau}{2}\|\bar{W}\|_F^2 + \lambda\|\bar{\boldsymbol{c}}\|_1. \tag{C.17}$$

where $r_N(\bar{W}) := \frac{1}{2N}\sum_{i=1}^{N}\min_{\boldsymbol{z}\in\{0,1\}^K}(y_i - \boldsymbol{z}^\mathcal{T}\bar{W}\boldsymbol{x}_i)^2$ is the empirical risk of $\bar{W}$. Let $\boldsymbol{c}^* := arg\min_{\boldsymbol{c}:supp(\boldsymbol{c})=\bar{\mathcal{A}}} F(\boldsymbol{c})$. We have $F(\boldsymbol{c}^*) \leq F(\bar{\boldsymbol{c}})$. Then from (C.6),

$$F(\hat{\boldsymbol{c}}) - F(\bar{\boldsymbol{c}}) \leq F(\hat{\boldsymbol{c}}) - F(\boldsymbol{c}^*) \leq \frac{4\gamma K}{\beta\mu^2}\left(\frac{1}{T}\right) + \frac{2(1-\mu)\lambda}{\mu}\sqrt{\frac{2K}{\beta}}. \tag{C.18}$$

In addition, the risk of $\hat{W}$ satisfies

$$r_N(\hat{W}) + \frac{\tau}{2}\|\hat{W}\|_F^2 + \lambda\|\hat{\boldsymbol{c}}\|_1 \leq F(\hat{\boldsymbol{c}}) \tag{C.19}$$

by the definition of the empirical risk $r_N(.)$ (since it is minimized w.r.t. the hidden assignments). Combining (C.17), (C.18) and (C.19), we obtain a bound on the difference of empirical risk

$$\begin{aligned} &r_N(\hat{W}) - r_N(\bar{W}) \\ &\leq \underbrace{\frac{\tau}{2}\|\bar{W}\|^2 + \lambda K}_{\text{bias of regularization}} + \underbrace{\frac{4\gamma K}{\beta\mu^2}\left(\frac{1}{T}\right) + \frac{2(1-\mu)\lambda}{\mu}\sqrt{\frac{2K}{\beta}}}_{\text{optimization error}} \end{aligned} \tag{C.20}$$

The remaining task is to bound the estimation error $|r(W) - r_N(W)|$. Since Algorithm 13 is a descent algorithm w.r.t. $F(\boldsymbol{c})$ and in the beginning $F(0) \leq 1/2$, we have $\|\boldsymbol{c}\|_1 \leq 1/\lambda$ and $\|W\|^2 \leq 1/\tau$ at any iterate. Then we can bound the estimation error by Lemma 16 for $\hat{W}$ belonging to the set $\mathcal{W}(T) := \{\hat{W} \in \mathbb{R}^{T\times D} \mid \|\hat{W}\|_F \leq \sqrt{1/\lambda\tau}\}$, giving

$$|r(\hat{W}) - r_N(\hat{W})| \leq \sqrt{\frac{2DT\log(4TN/\sqrt{\lambda\tau})}{N} + \frac{1}{N}\log(\frac{1}{\rho})}. \tag{C.21}$$

Combining (C.20) and (C.21), and choosing $\lambda = 1/(NK)$, $\tau = 1/(NR^2)$, we obtain $r(\hat{W}) - r(\bar{W}) \leq \epsilon$ with $T \geq \frac{4\gamma}{\mu^2\beta}(\frac{K}{\epsilon})$, and $N \geq \frac{DT}{\epsilon^2}(2\log(\frac{4RKT}{\epsilon}) + \log(\frac{1}{\rho}))$ for any $0 < \epsilon < 1$. □

Bibliography

[1] Alireza Aghasi, Afshin Abdi, Nam Nguyen, and Justin Romberg. Net-trim: Convex pruning of deep neural networks with performance guarantee. In *Advances in Neural Information Processing Systems*, pages 3180–3189, 2017. 1.2

[2] Edoardo M Airoldi, David Blei, Elena A Erosheva, and Stephen E Fienberg. *Handbook of Mixed Membership Models and Their Applications*. Chapman and Hall/CRC, 2014. 4.1

[3] Bjoern Andres, Thorsten Beier, and Jörg H Kappes. Opengm: A c++ library for discrete graphical models. *arXiv preprint arXiv:1206.0111*, 2012. 3.2.3

[4] Björn Andres, Jörg H Kappes, Ullrich Köthe, Christoph Schnörr, and Fred A Hamprecht. An empirical comparison of inference algorithms for graphical models with higher order factors using opengm. In *Joint Pattern Recognition Symposium*, pages 353–362. Springer, 2010. 3.2.3

[5] Alex Auvolat, Sarath Chandar, Pascal Vincent, Hugo Larochelle, and Yoshua Bengio. Clustering is efficient for approximate maximum inner product search. *arXiv preprint arXiv:1507.05910*, 2015. 3.3.3, 3.3.4

[6] R. Babbar and B. Schölkopf. Dismec: Distributed sparse machines for extreme multi-label classification. In *Proceedings of the Tenth ACM International Conference on Web Search and Data Mining (WSDM 2017)*, 2017. 2.2, 2.2.1, 2.2.3

[7] Amir Beck and Marc Teboulle. A fast iterative shrinkage-thresholding algorithm for linear inverse problems. *SIAM journal on imaging sciences*, 2(1):183–202, 2009. 3.2.2

[8] Kush Bhatia, Himanshu Jain, Purushottam Kar, Manik Varma, and Prateek Jain. Sparse local embeddings for extreme multi-label classification. In *Advances in Neural Information Processing Systems*, pages 730–738, 2015. 2.1.3, 2.2.3

[9] Christopher M Bishop and Markus Svenskn. Bayesian hierarchical mixtures of experts. In *Proceedings of the Nineteenth conference on Uncertainty in Artificial Intelligence*, pages 57–64. Morgan Kaufmann Publishers Inc., 2002. 4.2

[10] Nicolas Boumal, Vlad Voroninski, and Afonso Bandeira. The non-convex burer-monteiro approach works on smooth semidefinite programs. In *Advances in Neural Information Processing Systems*, pages 2757–2765, 2016. 4.1.3, 4.1.3, 4.2.3, 4.2.3, C.1

[11] Tamara Broderick, Brian Kulis, and Michael I Jordan. Mad-bayes: Map-based asymptotic derivations from bayes. In *ICML (3)*, pages 226–234, 2013. 1.2, 4.1, 4.1.1, 4.1.2, 4.1.2, 4.1.4

[12] Arun T Chaganty and Percy Liang. Spectral experts for estimating mixtures of linear regressions. In *Proceedings of the 30th International Conference on Machine Learning (ICML-13)*, pages 1040–1048, 2013. 1.2, 4.2

[13] Venkat Chandrasekaran, Benjamin Recht, Pablo A Parrilo, and Alan S Willsky. The convex geometry of linear inverse problems. *Foundations of Computational mathematics*, 12 (6):805–849, 2012. 4.1.2, 4.2.2

[14] Welin Chen, David Grangier, and Michael Auli. Strategies for training large vocabulary neural language models. *arXiv preprint arXiv:1512.04906*, 2015. 3.3.4

[15] Yao-Nan Chen and Hsuan-Tien Lin. Feature-aware label space dimension reduction for multi-label classification. In *Advances in Neural Information Processing Systems*, pages 1529–1537, 2012. 1.2, 2.1

[16] Yudong Chen, Xinyang Yi, and Constantine Caramanis. A convex formulation for mixed regression with two components: Minimax optimal rates. In *COLT*, pages 560–604, 2014. 4.2

[17] Anna Choromanska, Alekh Agarwal, and John Langford. Extreme multi class classification. In *NIPS Workshop: eXtreme Classification, submitted*, 2013. 1.2, 2.1

[18] Anna E Choromanska and John Langford. Logarithmic time online multiclass prediction. In *Advances in Neural Information Processing Systems*, pages 55–63, 2015. 1.2, 2.1, 2.1.1, 2.1.3, 2.2.3

[19] Koby Crammer and Yoram Singer. On the algorithmic implementation of multiclass kernel-based vector machines. *The Journal of Machine Learning Research*, 2:265–292, 2002. 2.1.1

[20] Koby Crammer and Yoram Singer. A family of additive online algorithms for category ranking. *The Journal of Machine Learning Research*, 3:1025–1058, 2003. 2.1.1

[21] Dipanjan Das, Desai Chen, André FT Martins, Nathan Schneider, and Noah A Smith. Frame-semantic parsing. *Computational linguistics*, 40(1):9–56, 2014. 1.2, 3.1

[22] Alexandre d'Aspremont, Laurent El Ghaoui, Michael I Jordan, and Gert RG Lanckriet. A direct formulation for sparse pca using semidefinite programming. *SIAM review*, 49(3): 434–448, 2007. 4.1

[23] Jia Deng, Alexander C Berg, Kai Li, and Li Fei-Fei. What does classifying more than 10,000 image categories tell us? In *Computer Vision–ECCV 2010*, pages 71–84. Springer, 2010. 1.2, 2.1

[24] Finale Doshi-Velez and Zoubin Ghahramani. Accelerated sampling for the indian buffet process. In *Proceedings of the 26th annual international conference on machine learning*, pages 273–280. ACM, 2009. 1.2, 4.1, 4.1.1, 4.1.4

[25] Finale Doshi-Velez, Kurt T Miller, Jurgen Van Gael, Yee Whye Teh, and Gatsby Unit. Variational inference for the indian buffet process. In *Proceedings of the Intl. Conf. on Artificial Intelligence and Statistics*, volume 12, pages 137–144, 2009. 1.2, 4.1, 4.1.1, 4.1.4

[26] John Duchi, Shai Shalev-Shwartz, Yoram Singer, and Tushar Chandra. Efficient pro-

jections onto the l 1-ball for learning in high dimensions. In *Proceedings of the 25th international conference on Machine learning*, pages 272–279. ACM, 2008. 3.2.2

[27] Gal Elidan, Amir Globerson, and Uri Heinemann. Pascal 2011 probabilistic inference challenge, 2012. 3.2.3

[28] Rong-En Fan, Kai-Wei Chang, Cho-Jui Hsieh, Xiang-Rui Wang, and Chih-Jen Lin. Liblinear: A library for large linear classification. *The Journal of Machine Learning Research*, 9:1871–1874, 2008. 2.1.1, 2.1.2, 2.1.3, 2.2.3

[29] Kevin Gimpel and Noah A Smith. Structured ramp loss minimization for machine translation. In *NAACL*, pages 221–231. Association for Computational Linguistics, 2012. 1.2, 3.1

[30] Amir Globerson and Tommi S Jaakkola. Fixing max-product: Convergent message passing algorithms for map lp-relaxations. In *Advances in neural information processing systems*, pages 553–560, 2008. 3.2.3

[31] Jacob Goldberger, Geoffrey E Hinton, Sam T Roweis, and Ruslan R Salakhutdinov. Neighbourhood components analysis. In *Advances in neural information processing systems*, pages 513–520, 2005. 2

[32] Thomas L Griffiths and Zoubin Ghahramani. Infinite latent feature models and the indian buffet process. In *NIPS*, volume 18, pages 475–482, 2005. 1.2, 4.1, 4.1.4

[33] Thomas L Griffiths and Zoubin Ghahramani. The indian buffet process: An introduction and review. *Journal of Machine Learning Research*, 12(Apr):1185–1224, 2011. 4.1, 4.1.1

[34] Yiwen Guo, Anbang Yao, and Yurong Chen. Dynamic network surgery for efficient dnns. In *Advances In Neural Information Processing Systems*, pages 1379–1387, 2016. 1.2

[35] Michael Gutmann and Aapo Hyvärinen. Noise-contrastive estimation: A new estimation principle for unnormalized statistical models. In *Proceedings of the Thirteenth International Conference on Artificial Intelligence and Statistics*, pages 297–304, 2010. 1.2

[36] Song Han, Huizi Mao, and William J Dally. Deep compression: Compressing deep neural networks with pruning, trained quantization and huffman coding. *arXiv preprint arXiv:1510.00149*, 2015. 1.2

[37] Song Han, Jeff Pool, John Tran, and William Dally. Learning both weights and connections for efficient neural network. In *Advances in neural information processing systems*, pages 1135–1143, 2015. 1.2

[38] Paul Hand and Babhru Joshi. A convex program for mixed linear regression with a recovery guarantee for well-separated data. *arXiv preprint arXiv:1612.06067*, 2016. 4.2

[39] Kohei Hayashi and Ryohei Fujimaki. Factorized asymptotic bayesian inference for latent feature models. In *Advances in Neural Information Processing Systems*, pages 1214–1222, 2013. 4.1.1

[40] Tom Heskes, Kees Albers, and Bert Kappen. Approximate inference and constrained optimization. In *Proceedings of the Nineteenth conference on Uncertainty in Artificial Intelligence*, pages 313–320. Morgan Kaufmann Publishers Inc., 2002. 3.2.3

[41] A.J. Hoffman. On approximate solutions of systems of linear inequalities. *Journal of Research of the National Bureau of Standards*, 1952. B.1, B.3

[42] Mingyi Hong and Zhi-Quan Luo. On the linear convergence of the alternating direction method of multipliers. *arXiv preprint arXiv:1208.3922*, 2012. 3.1.2, 3.2.2, B.1, B.3

[43] Cho-Jui Hsieh, Kai-Wei Chang, Chih-Jen Lin, S Sathiya Keerthi, and Sellamanickam Sundararajan. A dual coordinate descent method for large-scale linear svm. In *Proceedings of the 25th international conference on Machine learning*, pages 408–415. ACM, 2008. 2.2.2

[44] Xiangru Huang, Qixing Huang, Ian EH Yen, Pradeep Ravikumar, Ruohan Zhang, and Inderjit S Dhillon. Greedy direction method of multiplier for map inference of large output domain. In *JMLR workshop and conference proceedings*, volume 54, page 1550. NIH Public Access, 2017. 3

[45] Himanshu Jain, Yashoteja Prabhu, and Manik Varma. Extreme multi-label loss functions for recommendation, tagging, ranking & other missing label applications. In *Proceedings of the 22nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*. ACM, 2016. 2.2.3

[46] Kalina Jasinska, Krzysztof Dembczynski, Róbert Busa-Fekete, Karlson Pfannschmidt, Timo Klerx, and E Hüllermeier. Extreme f-measure maximization using sparse probability estimates. In *Proceedings of the 33nd International Conference on Machine Learning*, pages 1435–1444, 2016. 2.2.3

[47] Sébastien Jean, Kyunghyun Cho, Roland Memisevic, and Yoshua Bengio. On using very large target vocabulary for neural machine translation. *arXiv preprint arXiv:1412.2007*, 2014. 1.2, 3.3.4

[48] Thorsten Joachims, Thomas Finley, and Chun-Nam John Yu. Cutting-plane training of structural svms. *Machine Learning*, 77(1):27–59, 2009. 1.2, 3.1

[49] Ian T Jolliffe, Nickolay T Trendafilov, and Mudassir Uddin. A modified principal component technique based on the lasso. *Journal of computational and Graphical Statistics*, 12 (3):531–547, 2003. 4.1

[50] Sham Kakade, Shai Shalev-Shwartz, and Ambuj Tewari. On the duality of strong convexity and strong smoothness: Learning applications and matrix regularization. *Unpublished Manuscript, http://ttic. uchicago. edu/shai/papers/KakadeShalevTewari09. pdf*, 2009. 2.1.1

[51] Ashish Kapoor, Raajay Viswanathan, and Prateek Jain. Multilabel classification using bayesian compressed sensing. In *Advances in Neural Information Processing Systems*, pages 2645–2653, 2012. 1.2, 2.1

[52] S Sathiya Keerthi, Sellamanickam Sundararajan, Kai-Wei Chang, Cho-Jui Hsieh, and Chih-Jen Lin. A sequential dual method for large scale multi-class linear svms. In *Proceedings of the 14th ACM SIGKDD international conference on Knowledge discovery and data mining*, pages 408–416. ACM, 2008. 2.1.1, 2.1.1, 2.1.2

[53] Vladimir Kolmogorov. Convergent tree-reweighted message passing for energy minimiza-

tion. *IEEE transactions on pattern analysis and machine intelligence*, 28(10):1568–1583, 2006. 3.2.3

[54] Nikos Komodakis, Nikos Paragios, and Georgios Tziritas. Mrf optimization via dual decomposition: Message-passing revisited. In *2007 IEEE 11th International Conference on Computer Vision*, pages 1–8. IEEE, 2007. 3.2.3

[55] Frank R Kschischang, Brendan J Frey, and H-A Loeliger. Factor graphs and the sum-product algorithm. *IEEE Transactions on information theory*, 47(2):498–519, 2001. 3.2.3

[56] M Pawan Kumar, Vladimir Kolmogorov, and Philip HS Torr. An analysis of convex relaxations for map estimation. *Advances in Neural Information Processing Systems*, 20: 1041–1048, 2007. 1.2, 3.1

[57] Simon Lacoste-Julien and Martin Jaggi. On the global linear convergence of frank-wolfe optimization variants. In *Advances in Neural Information Processing Systems*, pages 496–504, 2015. 3.1.2, 3.1.2, 3.2.2, B.1, B.1, B.1, B.1, B.1, B.3, B.3, B.3

[58] Simon Lacoste-Julien and Martin Jaggi. On the global linear convergence of frank-wolfe optimization variants. In *Advances in Neural Information Processing Systems*, pages 496–504, 2015. 3.2.2

[59] Simon Lacoste-Julien, Martin Jaggi, Mark Schmidt, and Patrick Pletscher. Block-coordinate frank-wolfe optimization for structural svms. In *ICML 2013 International Conference on Machine Learning*, pages 53–61, 2013. (document), 1.2, 2.1.2, 3.1, 3.1.2, 3.1.3, 3.3, A.1, A.1

[60] Ke Li and Jitendra Malik. Fast k-nearest neighbour search via prioritized dci. In *International Conference on Machine Learning*, pages 2081–2090, 2017. 3.3.1, 3.3.2

[61] André FT Martins, Mário AT Figueiredo, Pedro MQ Aguiar, Noah A Smith, and Eric P Xing. Ad3: Alternating directions dual decomposition for map inference in graphical models. *Journal of Machine Learning Research*, 16:495–545, 2015. 3.2.3

[62] Ofer Meshi, David Sontag, Tommi Jaakkola, and Amir Globerson. Learning efficiently with approximate inference via dual losses. 2010. 3.1, 3.1.2, 3.1.2, 3.1.3

[63] Ofer Meshi, Mehrdad Mahdavi, and Alex Schwing. Smooth and strong: Map inference with linear convergence. In *Advances in Neural Information Processing Systems*, pages 298–306, 2015. 2.1.1, 3.2.3

[64] Ofer Meshi, Mehrdad Mahdavi, and David Sontag. On the tightness of lp relaxations for structured prediction. *arXiv preprint arXiv:1511.01419*, 2015. 1.2, 3.1, 3.1.1, 3.1.2

[65] Ofer Meshi, Nathan Srebro, and Tamir Hazan. Efficient training of structured svms via soft constraints. In *AISTAT*, 2015. 3.1, 3.1.2, 3.1.3

[66] Tomas Mikolov, Ilya Sutskever, Kai Chen, Greg S Corrado, and Jeff Dean. Distributed representations of words and phrases and their compositionality. In *Advances in neural information processing systems*, pages 3111–3119, 2013. 1.2, 3, 13, 3.3.4

[67] Kurt Miller, Michael I Jordan, and Thomas L Griffiths. Nonparametric latent feature models for link prediction. In *Advances in neural information processing systems*, pages 1276–1284, 2009. 1.2

[68] Andriy Mnih and Yee Whye Teh. A fast and simple algorithm for training neural probabilistic language models. *arXiv preprint arXiv:1206.6426*, 2012. 1.2

[69] Dmitry Molchanov, Arsenii Ashukha, and Dmitry Vetrov. Variational dropout sparsifies deep neural networks. *arXiv preprint arXiv:1701.05369*, 2017. 1.2

[70] Joris M. Mooij. libDAI: A free and open source C++ library for discrete approximate inference in graphical models. *Journal of Machine Learning Research*, 11:2169–2173, August 2010. URL `http://www.jmlr.org/papers/volume11/mooij10a/mooij10a.pdf`. 3.2.3

[71] Yair Movshovitz-Attias, Alexander Toshev, Thomas K Leung, Sergey Ioffe, and Saurabh Singh. No fuss distance metric learning using proxies. *arXiv preprint arXiv:1703.07464*, 2017. 2

[72] Stephen Mussmann and Stefano Ermon. Learning and inference via maximum inner product search. In *International Conference on Machine Learning*, pages 2587–2596, 2016. 3

[73] Yurii Nesterov et al. *Quality of semidefinite relaxation for nonconvex quadratic optimization*. Université Catholique de Louvain. Center for Operations Research and Econometrics [CORE], 1997. 4.1.3, 4.2.3, 4.2.3

[74] Julie Nutini, Mark Schmidt, Issam H Laradji, Michael Friedlander, and Hoyt Koepke. Coordinate descent converges faster with the gauss-southwell rule than random selection. In *Proceedings of the 32nd International Conference on Machine Learning (ICML-15)*, pages 1632–1641, 2015. 3.2.2

[75] Anton Osokin, Jean-Baptiste Alayrac, Isabella Lukasewitz, Puneet K Dokania, and Simon Lacoste-Julien. Minding the gaps for block frank-wolfe optimization of structured svms. *arXiv preprint arXiv:1605.09346*, 2016. 3.1.2

[76] Ioannis Partalas, Aris Kosmopoulos, Nicolas Baskiotis, Thierry Artieres, George Paliouras, Eric Gaussier, Ion Androutsopoulos, Massih-Reza Amini, and Patrick Galinari. Lshtc: A benchmark for large-scale text classification. *arXiv preprint arXiv:1503.08581*, 2015. 2.1.3, 3.3.4

[77] Yashoteja Prabhu and Manik Varma. Fastxml: a fast, accurate and stable tree-classifier for extreme multi-label learning. In *Proceedings of the 20th ACM SIGKDD international conference on Knowledge discovery and data mining*, pages 263–272. ACM, 2014. 1.2, 2.1, 2.1.3, 2.2.3

[78] Yuan Qi and TP Minka. Tree-structured approximations by expectation propagation. *Advances in Neural Information Processing Systems (NIPS)*, 16:193, 2004. 3.2.3

[79] Ali Rahimi and Benjamin Recht. Random features for large-scale kernel machines. In *Advances in neural information processing systems*, pages 1177–1184, 2007. 2.1.3

[80] Pradeep Ravikumar and John Lafferty. Quadratic programming relaxations for metric labeling and markov random field map estimation. In *ICML*, 2006. 1.2, 3.1

[81] P. Richtárik and M. Takáč. Iteration complexity of randomized block-coordinate descent methods for minimizing a composite function. *Mathematical Programming*, 2014. 2.2.2

[82] Rajhans Samdani and Dan Roth. Efficient decomposed learning for structured prediction. *ICML*, 2012. 3.1

[83] Florian Schroff, Dmitry Kalenichenko, and James Philbin. Facenet: A unified embedding for face recognition and clustering. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 815–823, 2015. 3.3.4

[84] Hanie Sedghi, Majid Janzamin, and Anima Anandkumar. Provable tensor methods for learning mixtures of generalized linear models. In *Artificial Intelligence and Statistics*, pages 1223–1231, 2016. 1.2, 4.2

[85] Shai Shalev-Shwartz and Tong Zhang. Stochastic dual coordinate ascent methods for regularized loss. *The Journal of Machine Learning Research*, 14(1):567–599, 2013. 2.1.2

[86] Shai Shalev-Shwartz, Yoram Singer, Nathan Srebro, and Andrew Cotter. Pegasos: Primal estimated sub-gradient solver for svm. *Mathematical programming*, 2011. 3.1.3

[87] Anshumali Shrivastava and Ping Li. Asymmetric lsh (alsh) for sublinear time maximum inner product search (mips). In *Advances in Neural Information Processing Systems*, pages 2321–2329, 2014. 3.3.1, 3.3.2

[88] Martin Slawski, Matthias Hein, and Pavlo Lutsik. Matrix factorization with binary components. In *Advances in Neural Information Processing Systems*, pages 3210–3218, 2013. 4.1, 4.1.1, 4.1.2, 4.1.4

[89] David Sontag, Do Kook Choe, and Yitao Li. Efficiently searching for frustrated cycles in map inference. In *28th Conference on Uncertainty in Artificial Intelligence, UAI 2012*, 2012. 3.2.3

[90] David Sontag, Talya Meltzer, Amir Globerson, Tommi S Jaakkola, and Yair Weiss. Tightening lp relaxations for map using message passing. *arXiv preprint arXiv:1206.3288*, 2012. 3.2.3

[91] Tonghua Su, Tianwen Zhang, and Dejun Guan. Corpus-based hit-mw database for offline recognition of general-purpose chinese handwritten text. *IJDAR*, 10(1):27–38, 2007. 3.1.3

[92] Ben Taskar, Carlos Guestrin, and Daphne Koller. Max-margin markov networks. In *Advances in neural information processing systems*, volume 16, 2003. 3.1.1

[93] Ben Taskar, Vassil Chatalbashev, Daphne Koller, and Carlos Guestrin. Learning structured prediction models: A large margin approach. In *ICML*, 2005. 3.1.1, 3.1.2

[94] Robert Tibshirani. Regression shrinkage and selection via the lasso. *Journal of the Royal Statistical Society. Series B (Methodological)*, pages 267–288, 1996. 4.2.2

[95] Ryan J Tibshirani et al. The lasso problem and uniqueness. *Electronic Journal of Statistics*, 7:1456–1490, 2013. 2.1.1, 2.1.1, 2.2.1, 2.3.2

[96] Ioannis Tsochantaridis, Thomas Hofmann, Thorsten Joachims, and Yasemin Altun. Support vector machine learning for interdependent and structured output spaces. In *ICML*, 2004. 3.1.1

[97] Hsiao-Yu Tung and Alexander J Smola. Spectral methods for indian buffet process inference. In *Advances in Neural Information Processing Systems*, pages 1484–1492, 2014.

1.2, 4.1, 4.1.1, 4.1.3, 4.1.4

[98] Naonori Ueda and Kazumi Saito. Parametric mixture models for multi-labeled text. *Advances in neural information processing systems*, pages 737–744, 2003. 4.1

[99] Sudheendra Vijayanarasimhan, Jonathon Shlens, Rajat Monga, and Jay Yagnik. Deep networks with large output spaces. *arXiv preprint arXiv:1412.7479*, 2014. 3

[100] Martin J Wainwright, Tommi S Jaakkola, and Alan S Willsky. Tree-reweighted belief propagation algorithms and approximate ml estimation by pseudo-moment matching. In *AISTATS*, 2003. 3.2.3

[101] Martin J Wainwright, Tommi S Jaakkola, and Alan S Willsky. Map estimation via agreement on trees: message-passing and linear programming. *IEEE transactions on information theory*, 51(11):3697–3717, 2005. 3.2.3

[102] Po-Wei Wang and J Zico Kolter. The mixing method for maxcut-sdp problem. *NIPS LHDS Workshop.*, 2016. 4.1.3, 4.1.3, 4.2.3, 4.2.3, C.1

[103] Kilian Q Weinberger and Lawrence K Saul. Distance metric learning for large margin nearest neighbor classification. *Journal of Machine Learning Research*, 10(Feb):207–244, 2009. 2

[104] Tomáš Werner. Revisiting the decomposition approach to inference in exponential families and graphical models. *Center for Machine Perception, Czech Technical University Prague, Research Report, CTU-CMP-2009-06, ftp://cmp. felk. cvut. cz/pub/cmp/articles/werner/Werner-TR-2009-06. pdf*, 2009. 3.2.3

[105] Jason Weston, Ameesh Makadia, and Hector Yee. Label partitioning for sublinear ranking. In *ICML (2)*, pages 181–189, 2013. 2.2.3

[106] PC Woodland and Daniel Povey. Large scale discriminative training for speech recognition. In *ASR2000-Automatic Speech Recognition Workshop (ITRW)*, 2000. 1.2, 3.1

[107] Chang Xu, Dacheng Tao, and Chao Xu. Robust extreme multi-label learning. In *Proceedings of the 22nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, San Francisco, CA, USA August*, pages 13–17, 2016. 2.2.3

[108] Lei Xu, Michael I Jordan, and Geoffrey E Hinton. An alternative model for mixtures of experts. In *Advances in neural information processing systems*, pages 633–640, 1995. 4.2

[109] Chen Yanover, Ora Schueler-Furman, and Yair Weiss. Minimizing and learning energy functions for side-chain prediction. *Journal of Computational Biology*, 15(7):899–911, 2008. 3.2.3

[110] Jonathan S Yedidia, William T Freeman, and Yair Weiss. Constructing free-energy approximations and generalized belief propagation algorithms. *IEEE Transactions on Information Theory*, 51(7):2282–2312, 2005. 3.2.3

[111] Ian EH Yen, Xin Lin, EDU Jiong Zhang, EDU Pradeep Ravikumar, and Inderjit S Dhillon. A convex atomic-norm approach to multiple sequence alignment and motif discovery. 2016. 3.1.2

[112] Ian EH Yen, Dmitry Malioutov, and Abhishek Kumar. Scalable exemplar clustering and

facility location via augmented block coordinate descent with column generation. In *AISTAT*, 2016. 3.1.2

[113] Ian EH Yen, Xiangru Huang, Wei Dai, Pradeep Ravikumar, Inderjit Dhillon, and Eric Xing. Ppdsparse: A parallel primal-dual sparse method for extreme classification. In *Proceedings of the 23rd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, pages 545–553. ACM, 2017. 2, 2.3.1

[114] Ian En-Hsu Yen, Cho-Jui Hsieh, Pradeep K Ravikumar, and Inderjit S Dhillon. Constant nullspace strong convexity and fast convergence of proximal methods under high-dimensional settings. In *Advances in Neural Information Processing Systems*, pages 1008–1016, 2014. 3.2.2

[115] Ian En-Hsu Yen, Ting-Wei Lin, Shou-De Lin, Pradeep K Ravikumar, and Inderjit S Dhillon. Sparse random feature algorithm as coordinate descent in hilbert space. In *Advances in Neural Information Processing Systems*, pages 2456–2464, 2014. 2.1.3

[116] Ian En-Hsu Yen, Xin Lin, Kai Zhong, Pradeep Ravikumar, and Inderjit Dhillon. A convex exemplar-based approach to mad-bayes dirichlet process mixture models. In *International Conference on Machine Learning*, pages 2418–2426, 2015. 4.2

[117] Ian En-Hsu Yen, Kai Zhong, Cho-Jui Hsieh, Pradeep K Ravikumar, and Inderjit S Dhillon. Sparse linear programming via primal and dual augmented coordinate descent. In *NIPS*, 2015. 3.1.2, 3.2.3

[118] Ian En-Hsu Yen, Xiangru Huang, Pradeep Ravikumar, Kai Zhong, and Inderjit Dhillon. Pd-sparse: A primal and dual sparse approach to extreme multiclass and multilabel classification. In *International Conference on Machine Learning*, pages 3069–3077, 2016. 2, 2.2, 2.2.1, 2.2.1, 2.2.2, 2.2.3, 2.3.1, 3.3.4

[119] Ian En-Hsu Yen, Xiangru Huang, Kai Zhong, Ruohan Zhang, Pradeep K Ravikumar, and Inderjit S Dhillon. Dual decomposed learning with factorwise oracle for structural svm of large output domain. In *Advances in Neural Information Processing Systems*, pages 5030–5038, 2016. 3

[120] Ian En-Hsu Yen, Xin Lin, Jiong Zhang, Pradeep Ravikumar, and Inderjit Dhillon. A convex atomic-norm approach to multiple sequence alignment and motif discovery. In *International Conference on Machine Learning*, pages 2272–2280, 2016. 4.2

[121] Ian En-Hsu Yen, Wei-Chen Chang, Sung-En Chang, Arun Sai Suggula, Shou-De Lin, and Pradeep. Ravikumar. Latent feature lasso. *International Conference on Machine Learning*, 2017. 4.2

[122] Ian En-Hsu Yen, Wei-Cheng Lee, Sung-En Chang, Arun Sai Suggala, Shou-De Lin, and Pradeep Ravikumar. Latent feature lasso. In *International Conference on Machine Learning*, pages 3949–3957, 2017. 4

[123] Xinyang Yi, Constantine Caramanis, and Sujay Sanghavi. Alternating minimization for mixed linear regression. In *ICML*, pages 613–621, 2014. 4.2, 4.2.2, 4.2.3

[124] Xinyang Yi, Constantine Caramanis, and Sujay Sanghavi. Solving a mixture of many random linear equations by tensor decomposition and alternating minimization. *arXiv*

preprint arXiv:1608.05749, 2016. 1.2, 4.2, 4.2.4

[125] Hsiang-Fu Yu, Prateek Jain, Purushottam Kar, and Inderjit S Dhillon. Large-scale multi-label learning with missing labels. *arXiv preprint arXiv:1307.5101*, 2013. 1.2, 2.1, 2.1.3, 2.2.3

[126] Kai Zhong, Ian En-Hsu Yen, Inderjit S Dhillon, and Pradeep K Ravikumar. Proximal quasi-newton for computationally intensive l1-regularized m-estimators. In *Advances in Neural Information Processing Systems*, pages 2375–2383, 2014. 2.3.2

[127] Kai Zhong, Prateek Jain, and Inderjit S Dhillon. Mixed linear regression with multiple components. In *Advances in Neural Information Processing Systems*, pages 2190–2198, 2016. 1.2, 4.2, 4.2.4

[128] Hui Zou and Trevor Hastie. Regularization and variable selection via the elastic net. *Journal of the Royal Statistical Society: Series B (Statistical Methodology)*, 67(2):301–320, 2005. 2.1.1

[129] Ghahramani Zoubin. Scaling the indian buffet process via submodular maximization. In *Proceedings of the 30th International Conference on Machine Learning (ICML-13)*, pages 1013–1021, 2013. 4.1.1