Sublinear-Time Optimization for High-Dimensional Learning

Ian En-Hsu Yen

Thesis Proposal
Nov 2017

Thesis Committee:
Pradeep Ravikumar
Inderjit S. Dhillon
Geoffrey Gordon
Barnabas Poczos

Machine Learning Department
Carnegie Mellon University
Abstract

Across domains, the scale of data and complexity of machine learning models have both been increasing greatly in recent years. For many such large scale models of interest, tractable estimation without access to extensive computational infrastructure is still an open problem. In this thesis, we approach the tractability of large-scale estimation problems through the lens of leveraging sparse structure inherent in the learning objective, which allows us to develop algorithms sublinear in the size of domain by greedily searching for atoms comprising the optimal structure. We address the following three questions for each model of interest: (a) how to formulate model estimation as a high-dimensional optimization problem with tractable sparse structure, (b) how to efficiently i.e. in sublinear time, search for the optimal structure, (c) how to guarantee fast convergence of our optimization algorithm? By answering these questions, we develop state-of-the-art learning algorithms for varied domains as extreme classification, structured prediction with large output domains, and design new estimators for latent-variable models that enjoy polynomial computational and sample complexities without restrictive assumptions.
## Contents

1 Introduction .................................................. 4  
1.1 Extreme Classification ........................................ 4  
1.2 Structured Prediction with Large Output Domain .......... 5  
1.3 Estimation of Latent-Variable Models ......................... 5  

2 Sparse Optimization for Large Domain ......................... 7  
2.1 Extreme Classification: Primal & Dual Sparsity .............. 7  
2.1.1 Problem Setup ............................................ 7  
2.1.2 Algorithm .................................................. 10  
2.1.3 Experiments ............................................... 12  
2.2 Structured Prediction: Factor Decomposition ................. 13  
2.2.1 Problem Setup ............................................ 13  
2.2.2 Dual-Decomposed Learning ................................ 15  
2.2.3 Experiments ............................................... 18  
2.3 Proposed Works .............................................. 20  
2.3.1 Deep Extreme Classification via Loss Decomposition .... 20  
2.3.2 Learning Sparse Layers in Neural Network ............... 21  

3 Sparse Estimator for Latent-Variable Models .................. 22  
3.1 Latent-Feature Models ........................................ 22  
3.1.1 Problem Setup ............................................ 22  
3.1.2 Latent Feature Lasso ...................................... 23  
3.1.3 Algorithm .................................................. 23  
3.1.4 Experiments ............................................... 27  
3.2 Proposed Work ............................................... 29  
3.2.1 Matrix Factorization with Binary Components ............ 29  

4 Conclusion ..................................................... 30
Chapter 1

Introduction

In this thesis proposal, we approach the tractability of large-scale estimation problem by leveraging the structure inherent in the learning objective, which allows us to:

- Design sublinear-time algorithm for existing objective without sacrificing quality of the solution.
- 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 and Learning Structured Predictor as two main examples to demonstrate how to exploit sparse structures of the objective to design sublinear-time algorithms when the problem has large output domains. In the end of the first part, we explore how the techniques developed for Extreme and Structured Classification can be potentially extended to develop sublinear-time algorithms for non-convex deep architectures.

For the second part, we show that, under the sparse optimization scheme, we can formulate discrete latent-variable models such as Latent-Feature allocation, Community Detection and Binary Word Embedding as high-dimensional sparse estimation problems, which allows us to develop the first computationally tractable method with theoretical guarantees for the family of discrete latent-variable models, without strong distributional assumption on the observed data. We propose the approach as a general method to factorize any matrix into factors of binary components based on an arbitrary loss function.

1.1 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 the multiclass setting, we are given the fact that only one label is correct, while in the multilabel setting, multiple correct 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 [1]. There are a couple of approaches proposed in the literature to address this issues:

- **Structural Assumption:** One of the most popular approach to reduce complexity due to large number of classes is to impose structural relations among labels to reduce the model complexity such as low rank [2, 3, 4], tree-structure [5, 6, 7] and clusters [8]. However, in practice such structural assumptions do not often hold and thus enforcing the structures obliviously results in the decrease of model performance.

- **Sampling Approximation:** 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 [9, 10, 11]. The approach is effective when the loss has gradient evenly distributed on classes. However, a reasonable classification model often has its loss concentrated on a couple of confusing classes, which results in the slow convergence of those sampling approaches.

- **Candidate Search**: Yet another approach is to formulate the problem of finding confusing classes as a Maximum Inner-Product Search (MIPS) problem, which could be solved efficiently via pre-built data structure such as Hashing or Hierarchical Clustering [12, 13]. However, the MIPS problem itself has low recall/precision in high dimension, and thus results in much lower accuracy when employed in the training of classifiers.

In this thesis proposal, 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.

### 1.2 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 [14, 15, 16] 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 [17, 18]. For instance, many applications in NLP such as Machine Translation [19], Speech Recognition [20], and Semantic Parsing [21] have output domains as large as the size of vocabulary in the target language, giving an expensive cost of inference.

A line of research employs *approximate decoding* methods to retain efficiency. However, it is shown that standard training of structured predictor can be hurt a lot when combined with an approximate decoding method [22, 23, 24], since the negative prediction found by approximate inference can have actually a lower score than the ground truth, which results in training without improvement after certain iterations. Another approach to avoid inference during training is by introducing a loss function conditioned on the given labels of neighboring output variables, such as decomposed hinge loss [25] and maximizing pseudo-likelihood [26]. However, these approaches introduce much more variance to the estimation of model and could degrade testing accuracy significantly especially for large output domain.

In this proposal, we propose a new approach of learning structured predictor which, instead of employing inference as a subroutine in a learning algorithm, employs the *multiclass 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 each multiclass learning problems.

### 1.3 Estimation of Latent-Variable Models

We are interested in the estimation of models with binary latent-variables (i.e. variables taking 0/1 values), also called Latent-Feature Model (LFM) or Indian Buffet Process (IBP) [27, 28, 29] in the Nonparametric Bayesian’s literature. The estimation (inference) of such latent-variable models 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 such models is not a concave function of its parameters.

In practice, one often relies on local search methods similar to *Expectation Maximization* [30] to find an estimate of parameters, or employ Markov Chain Monte Carlo (MCMC) [31] or variational methods...
[32] to obtain an approximate posterior distribution. However, none of these approaches can provide guarantees on the quality of solution in polynomial time.

In the nonparametric setting with an IBP prior, [29] 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 LFM s 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.

In this thesis, we propose a novel approach to the estimation of latent-variable model by formulating it as a high-dimensional sparse optimization problem. Unlike Spectral method, our approach does not rely on any restrictive assumption on the distribution of data, while providing polynomial runtime and approximation guarantees in terms of a sparsity-accuracy trade-off.
Chapter 2

Sparse Optimization for Large Domain

In this chapter, we first briefly state our works on sublinear-time algorithms for Extreme Classification [33, 34] and the training of structured predictor of large domain [35] in section 2.1 and 2.2 respectively. Then we discuss two potential extensions of the methodologies to deep architectures in section 2.3.

2.1 Extreme Classification: Primal & Dual Sparsity

2.1.1 Problem Setup

Our formulation is based on the Empirical Risk Minimization (ERM) framework. Given a collection of training instances \( D = \{ (x_i, y_i) \}_{i=1}^N \) where \( x_i \in \mathbb{R}^D \) is \( D \)-dimensional (possibly sparse) feature vector of \( i \)-th instance and \( y_i \in \{0, 1\}^K \) 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 \( \mathcal{P}(y) = \{ k \in [K] \mid y_k = 1 \} \) to denote positive label indexes, while using \( \mathcal{N}(y) = \{ k \in [K] \mid y_k = 0 \} \) 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 \( \text{nnz}(y) \) is small and not growing linearly with \( K \). For example, in multiclass classification problem, we have \( \text{nnz}(y) = 1 \), and the assumption is also satisfied typically in multilabel problems. Denote \( X := (x_i^T)_{i=1}^N \) as the \( N \times D \) design matrix and \( Y := (y_i^T)_{i=1}^N \) as the \( N \) by \( K \) label matrix, our goal is to learn a classifier \( h : \mathbb{R}^D \rightarrow [K] \)

\[
h(x) := \arg\max_k \langle w_k, x \rangle,
\]

parameterized by a \( D \times K \) matrix \( W = (w_k)_{k=1}^K \).

Loss with Dual Sparsity

In this paper, we consider the separation ranking loss [36] that penalizes the prediction on an instance \( x \) by the highest response from the set of negative labels minus the lowest response from the set of positive labels

\[
L(z, y) = \max_{k_n \in \mathcal{N}(y)} \max_{k_p \in \mathcal{P}(y)} (1 + z_{k_n} - z_{k_p})_+
\]

(2.2)

where an instance has zero loss if all positive labels \( k_p \in \mathcal{P} \) have higher responses than that of negative labels \( k_n \in \mathcal{N} \) plus a margin. In the multiclass setting, let \( p(y) \) be the unique positive label. The loss (2.2) becomes the well-known multiclass SVM loss

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

(2.3)

proposed in [37] and widely-used in linear classification package such as LIBLINEAR [38]. 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
where $W = [w_1, w_2, ..., w_K]$. Let $\alpha_k := (\alpha_{ik})_{k \in [K]}$ and $\alpha^i := (\alpha_{ik})_{k \in [K]}$. The dual problem is of the form

$$
\min_{\alpha} \frac{1}{2} \sum_{k=1}^{K} \|w_k(\alpha)\|^2 + \sum_{i=1}^{N} e^T \alpha^i
$$

s.t. $\alpha^i \in \Delta^K_i$, $\forall i \in [N]$ (2.5)

where

$$
w_k(\alpha_k) = \sum_{i=1}^{N} \alpha_{ik} x_i = X^T \alpha_k,
$$

(2.6)

$e_i = 1 - y_i$, and $\Delta^K_i = \{\alpha \mid \sum_{k=1}^{K} \alpha_k = 0, \alpha_{p(y_i)} \leq C, \alpha_k \leq 0, \forall k \neq p(y_i)\}$ is a shifted simplex of $K$ corners. In particular, the optimal solution $\alpha$ of (2.5) satisfies: for $k \neq p(y_i)$, $\alpha^\ast_k \neq 0$ if and only if label $k$ has highest response $z_{ik} = \langle w_k, x_i \rangle$ that attains the maximum of (2.3). Therefore, to identify active variables that correspond to the confusing labels, [39] 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 $\ell_1$ penalty

$$
\lambda \sum_{k=1}^{K} \|w_k\|_1 + \sum_{i=1}^{N} L(W^T x_i, y_i).
$$

(2.7)

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

$$
\lambda \rho_k^\ast + \sum_{i=1}^{N} \alpha_{ik}^\ast x_i = \lambda \rho_k^\ast + X^T \alpha_k = 0, \forall k \in [K]
$$

(2.8)

for some subgradients $\rho_k^\ast \in \partial \|w_k^\ast\|_1$ and $\alpha^\ast \in \partial L(z_i, y_i)$ with $z_i = W^\ast x_i$. Recall that the subgradients $\alpha^\ast$ of loss (2.2) have $\alpha_{ik}^\ast \neq 0$ for some $k^\ast \neq k$ only if $k^\ast$ is the confusing label that satisfies

$$
k^\ast \in \arg \max_{k \neq k} \langle w_k, x_i \rangle.
$$

This means we have $n\text{nz}(\alpha^\ast) \ll K$ and $n\text{nz}(A) \ll NK$ 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 $\rho_k$ of $\ell_1$-norm satisfies

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

(2.9)
which means the set of non-zero primal variables $B_k^* = \{ j \mid w_{jk}^* \neq 0 \}$ at optimal satisfies
\begin{equation}
\lambda \text{sign}(|w_{jk}^*|)B_k^* = [X^T \alpha_k^*]B_k^*, \quad (2.10)
\end{equation}
which is a linear system of $|B_k^*|$ equality constraints and $\text{nnz}(\alpha_k)$ variables. However, for general design matrix $X$ that draws from any continuous probability distribution [40], the above cannot be satisfied unless
\begin{equation}
\text{nnz}(w_k^*) = |B_k^*| \leq \text{nnz}(\alpha_k^*), \forall k \in [K] \quad (2.11)
\end{equation}
and (2.11) further implies
\begin{equation}
\text{nnz}(W^*) \leq \text{nnz}(A^*) \quad (2.12)
\end{equation}
by summation over $K$, where $A^*$ an $N \times K$ matrix of stacked $(\alpha_k^*)_{k=1}^K$. 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 $\ell_1$ 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 $\ell_1$ penalty. The result is actually a simple extension to the fact that the number of non-zero weights at optimal under $\ell_1$ penalty is less or equal to the number of samples [40]. 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
\begin{equation}
\text{nnz}(W^*) \leq \text{nnz}(A^*)
\end{equation}
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 [41, 42], this issue can be resolved simply via adding an additional strongly convex regularizer to (2.7), the Elastic-Net-regularized problem
\begin{equation}
\sum_{k=1}^K \frac{1}{2} \|w_k\|^2 + \lambda \|w_k\|_1 + C \sum_{i=1}^N L(W^T x_i, y_i) \quad (2.13)
\end{equation}
has dual form
\begin{equation}
\min_{\alpha} \quad G(\alpha) := \frac{1}{2} \sum_{k=1}^K \|w_k(\alpha_k)\|^2 + \sum_{i=1}^N e_i^T \alpha^i \quad (2.14)
\end{equation}
s.t. $\alpha^i \in C_i$, $\forall i \in [N]$
that entangles variable of different samples $\alpha^i, \alpha^{i'}$ only through a smooth term $\sum_{k=1}^K \|w_k(\alpha_k)\|^2 / 2$, where
\begin{equation}
w_k(\alpha_k) := \text{prox}_{\lambda \| \cdot \|_1}(X^T \alpha_k). \quad (2.15)
\end{equation}

and
\begin{equation}
C_i := \left\{ \alpha \mid \sum_{k \in N_i} (-\alpha_k) = \sum_{k \in P_i} \alpha_k \in [0, C], \quad 0 \leq \alpha_k \leq 0, \forall k \in N_i \right\}. \quad (2.16)
\end{equation}

The proximal operator of $\ell_1$-norm $\text{prox}_{\lambda \| \cdot \|_1}(v)$ performs soft-thresholding to each single element $v_j$ as
\begin{equation}
\text{prox}_{\lambda \| \cdot \|_1}(v_j) := \begin{cases} 
0, & |v_j| \leq \lambda \\
v_j - \lambda, & v_j > \lambda \\
v_j + \lambda, & v_j < \lambda
\end{cases}
\end{equation}
The dual problem (2.14) has very similar form to that from purely $\ell_2$ regularized problem (2.5), with difference on the definition of $w_k$ (2.15), where the $\ell_1$-$\ell_2$-regularized problem has $w_k(\alpha_k)$ being a sparse
Algorithm 1 Fully-Corrective BCFW

[0:] Initialize $\alpha^0 = 0, \mathcal{A}^0 = \emptyset$.

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:] $A^t_{i+\frac{1}{2}} = A^t_i \cup \{k^*\}$.

[4:] Solving subproblem (2.21) w.r.t. active set $A^t_{i+\frac{1}{2}}$.

[5:] $A^{t+1}_i = A^t_{i+\frac{1}{2}} \setminus \{k \mid \alpha^t_{ik} = 0, k \notin \mathcal{P}_i\}$.

[6:] Maintain $w(\alpha), v(\alpha)$ via (2.22).

end for.

vector obtained from applying soft-thresholding operator to $X^T \alpha_k$. 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 $\ell_1$ regularizer, while the solution from Elastic-Net is often of higher prediction accuracy [43]. 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 [6].

2.1.2 Algorithm

The objective (2.14) comprises a smooth function subject to constraints $C_1, \ldots, C_N$ separable w.r.t. blocks of variables $\alpha_1, \alpha_2, \ldots, \alpha_N$. 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 $L_2$-regularization, a BCFW method that searches the greedy coordinate $\alpha^*_{ik}$ at each iterate is not better than a Block Coordinate Descent (BCD) algorithm that performs updates on the whole block of variable $\alpha^i$ [39, 38], 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

$$\nabla_{\alpha^i} G(\alpha) = W^T x_i - e_i. \quad (2.17)$$

If a primal-sparse $W$ can be maintained via (2.15), the gradient can be evaluated in time $O(nnz(x_i)nnz(w_j))$ (and $O(nnz(W))$ for dense $x_i$). In contrast, the update of the whole block of variable $\alpha^i$ would require maintaining relation (2.15) for $w_1...w_K$, which cannot exploit sparsity of $w_k$ 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 $A^t_i$ for each sample $i$. In each iteration, the BCFW method draws a block of variables $\alpha^i$ uniformly from $\{\alpha^i\}_{i=1}^N$, and finds greedy direction based on a local linear approximation

$$\alpha^{it}_{FW} := \arg\min_{\alpha^i \in \mathcal{C}_i} \langle \nabla_{\alpha^i} G(\alpha^i), \alpha^i \rangle. \quad (2.18)$$

1The coordinate descent method has global convergence only on problem where the non-smooth terms are separable w.r.t. the coordinates.
For \( C_t \) of structure (2.16), (2.18) is equivalent to finding the most violating pair of positive, negative labels:

\[
(k_n^*, k^*_p) := \argmin_{k_n \in \mathcal{N}_i, k_p \in \mathcal{P}_i} \langle \nabla \alpha^t G(\alpha^t), (\delta_{kp} - \delta_{kn}) \rangle,
\]

(2.19)

where \( \delta_k \) 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:

\[
k_n^* := \argmax_{k_n \in \mathcal{N}_i} \langle \nabla \alpha^t G(\alpha^t), \delta_{kn} \rangle,
\]

\[
= \argmax_{k_n \in \mathcal{N}_i} \langle w^t_{k_n}, x_i \rangle - 1
\]

(2.20)

which can be computed in \( O(nnz(x_i)nnz(w_j)) \), where \( j \) is the feature \( j \) of most non-zero labels in \( W \), among all nonzero features \( x_i \).

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

\[
\min_{\alpha_{Ai} \in C_t} \langle \nabla \alpha_{Ai} G, \alpha_{Ai} - \alpha^t_{Ai} \rangle + \frac{Q_i}{2} \| \alpha_{Ai} - \alpha^t_{Ai} \|^2
\]

(2.21)

where \( Q_i = \| x_i \|^2 \) and \( \mathcal{A}_i = \mathcal{A}_i \cup \{ k_n^* \} \). 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 \( C_t \) of the same complexity.

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

\[
v^t_k = X^T \alpha^t_k, \quad w^t_k = \text{prox}_{\lambda \| \cdot \|}(v^t_k).
\]

(2.22)

where maintaining the first relation costs \( O(nnz(x_i)|\mathcal{A}_i|) \) 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 \( \{ \alpha_i^t \}_{i=1}^N \) is bounded by \(|\mathcal{A}| \ll NK\), while the storage for maintaining primal variable is dominated by the space for \( \{ v_k \}_{k=1}^K \), which in the worst case, requires \( O(DK) \). However, by the definition of \( v_k \) (2.22), the number of non-zero elements in \( \{ v_k \}_{k=1}^K \) is bounded by \( O(nnz(X) \max_{\mathcal{A}_i}(|\mathcal{A}_i|)) \), with \( \max_{\mathcal{A}_i}(|\mathcal{A}_i|) \) bounded by the number of BCFW passes. This means the space requirement of the algorithm is only \( t \times \) 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 \( O(\text{nnz}(x_i)nnz(w^t_k) + \text{nnz}(x_i)|\mathcal{A}_i|) \). In case data matrix is dense the cost for one pass of FC-BCFW over all variables can be written as \( O(N\text{nnz}(W) + D\text{nnz}(A)) \), where \( A \) is the \( N \) by \( K \) matrix reshape of \( \alpha \). Let

\[
k_W := \frac{\text{nnz}(W)}{D}, \quad k_A := \frac{\text{nnz}(A)}{N}
\]

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

\[
O(N\text{nnz}(W) + D\text{nnz}(A)) = O(NDk_W + NDk_A).
\]

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 \( \ell_1 \)-regularized problem (which we observed empirically). We have \( \text{nnz}(W) \lesssim \text{nnz}(A) \) 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 $G(\alpha)$ be the dual objective (2.14). The iterates $\{\alpha^t\}_{t=1}^\infty$ given by the Fully-Corrective Block-Coordinate Frank-Wolfe (Algorithm 1) has

$$G(\alpha^t) - G^* \leq \frac{2(QR^2 + \Delta G^0)}{t/N + 2}, \quad t \geq 0$$ \hspace{1cm} (2.23)

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

Note our objective $G(\alpha^t)$ is $N$ times of the objective defined by average loss in for example [17, 44], 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 [38] one-versus-all logistic regression (1vsA-Logi).
- LibLinear one-vs-all SVM (1vsA-SVM).
- LibLinear multiclass SVM (Multi-SVM).
- LibLinear one-vs-all 1-regularized logistic regression solver (1vsA-L1-Logi).
- Vowpal-Wabbit (VW): A public fast learning system proposed in [6] for Extreme multiclass classification. We use the online trees (Tree) options provided by their solver.
- FastXML: An Extreme multilabel classification method [5] 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 [3]. 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 [45]. 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. EUR-Lex and bibtex are from Mulan multilabel data collections. $^3$ LSHTC1, Dmoz and LSHTC-wiki are from LSHTC2 competition described in [46]. RCV1-regions, aloi.bin and sector are from LIBSVM data collection $^4$, where aloi.bin uses Random Binning features [47, 48] approximating effect of RBF Laplacian kernel.

The statistics of data sets and results are shown in Table 2.1 and 2.2. 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.

\[\text{http://image-net.org/} \]
\[\text{mulan.sourceforge.net/datasets-mlc.html} \]
\[\text{www.csie.ntu.edu.tw/~cjlin/libsvmtools/datasets/multilabel.html} \]
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.

<table>
<thead>
<tr>
<th>Data</th>
<th>FastXML</th>
<th>LEML</th>
<th>L1vsA-Logi</th>
<th>L1vsA-SVM</th>
<th>Multi-SVM</th>
<th>L1vsA-L1-Logi</th>
<th>PD-Sparse</th>
<th>VW Tree</th>
<th>SLEEC</th>
</tr>
</thead>
<tbody>
<tr>
<td>LSHTC1</td>
<td>2131s</td>
<td>87950s</td>
<td>$\approx$ 6d</td>
<td>23744s</td>
<td>6411s</td>
<td>$\approx$ 14d</td>
<td>952s</td>
<td>1193s</td>
<td>10793s</td>
</tr>
<tr>
<td>N=83805</td>
<td>308M</td>
<td>7.7G</td>
<td>11G</td>
<td>4.5G</td>
<td>$\approx$ 57M</td>
<td>92M</td>
<td>744M</td>
<td>1.38G</td>
<td></td>
</tr>
<tr>
<td>D=347255</td>
<td>6.33s</td>
<td>189s</td>
<td>N/A</td>
<td>50.3s</td>
<td>49.0s</td>
<td>N/A</td>
<td>6.20s</td>
<td>6.84s</td>
<td>155s</td>
</tr>
<tr>
<td>K=12294</td>
<td>21.66s</td>
<td>16.52s</td>
<td>N/A</td>
<td>23.22</td>
<td>22.4</td>
<td>N/A</td>
<td>22.60s</td>
<td>10.56</td>
<td>12.8</td>
</tr>
<tr>
<td>Dmoz</td>
<td>6900s</td>
<td>97331s</td>
<td>$\approx$ 2d</td>
<td>160543s</td>
<td>N/A</td>
<td>$\approx$ 56d</td>
<td>2088.14s</td>
<td>7103s</td>
<td>113258s</td>
</tr>
<tr>
<td>N=345068</td>
<td>1.5G</td>
<td>3.6G</td>
<td>19G</td>
<td>N/A</td>
<td>$\approx$ 406M</td>
<td>40M</td>
<td>1.8G</td>
<td>3.23G</td>
<td></td>
</tr>
<tr>
<td>D=833484</td>
<td>57.1s</td>
<td>1298s</td>
<td>N/A</td>
<td>429.7s</td>
<td>N/A</td>
<td>N/A</td>
<td>6.74s</td>
<td>28s</td>
<td>3292s</td>
</tr>
<tr>
<td>K=11947</td>
<td>38.4s</td>
<td>31.28s</td>
<td>N/A</td>
<td>36.8</td>
<td>N/A</td>
<td>N/A</td>
<td>39.58s</td>
<td>21.27</td>
<td>32.49</td>
</tr>
<tr>
<td>imgNet</td>
<td>28440s</td>
<td>107490s</td>
<td>380971s</td>
<td>28640s</td>
<td>14510s</td>
<td>472611s</td>
<td>4958s</td>
<td>6492s</td>
<td>520570s</td>
</tr>
<tr>
<td>N=1261404</td>
<td>914M</td>
<td>13M</td>
<td>14M</td>
<td>23M</td>
<td>24M</td>
<td>1.8M</td>
<td>3.6M</td>
<td>35M</td>
<td>2.76G</td>
</tr>
<tr>
<td>D=1000</td>
<td>139s</td>
<td>554s</td>
<td>315.3s</td>
<td>136.8s</td>
<td>203.4s</td>
<td>390.5s</td>
<td>329.5s</td>
<td>37.7s</td>
<td>45372s</td>
</tr>
<tr>
<td>K=1000</td>
<td>6.48s</td>
<td>7.21</td>
<td>8.56</td>
<td>15.25</td>
<td>10.3</td>
<td>10.07</td>
<td>12.7</td>
<td>5.37</td>
<td>8.5</td>
</tr>
<tr>
<td>aloi.bin</td>
<td>2410s</td>
<td>62440s</td>
<td>42390s</td>
<td>9468s</td>
<td>449s</td>
<td>31770s</td>
<td>773.8s</td>
<td>334.3s</td>
<td>12200s</td>
</tr>
<tr>
<td>N=100000</td>
<td>992M</td>
<td>5.4G</td>
<td>15G</td>
<td>5.9G</td>
<td>612M</td>
<td>18M</td>
<td>7.1M</td>
<td>106M</td>
<td>1.96G</td>
</tr>
<tr>
<td>D=636911</td>
<td>10.99s</td>
<td>38.83s</td>
<td>16.61s</td>
<td>22.83s</td>
<td>12.42s</td>
<td>13.24s</td>
<td>1.42s</td>
<td>1.59s</td>
<td>191s</td>
</tr>
<tr>
<td>K=1000</td>
<td>95.5s</td>
<td>88.16</td>
<td>96.34</td>
<td>96.63</td>
<td>96.66</td>
<td>95.71</td>
<td>96.33</td>
<td>89.47</td>
<td>92.55</td>
</tr>
<tr>
<td>sector</td>
<td>100.77s</td>
<td>556.31s</td>
<td>107.12s</td>
<td>19.46s</td>
<td>11.46s</td>
<td>102.31s</td>
<td>14.12s</td>
<td>327.34s</td>
<td>164.3s</td>
</tr>
<tr>
<td>N=38658</td>
<td>7.0M</td>
<td>48.4M</td>
<td>129M</td>
<td>62M</td>
<td>57M</td>
<td>580K</td>
<td>1.6M</td>
<td>17M</td>
<td>223.5M</td>
</tr>
<tr>
<td>D=55197</td>
<td>0.25s</td>
<td>0.069s</td>
<td>0.114s</td>
<td>0.156s</td>
<td>0.169s</td>
<td>0.13s</td>
<td>0.09s</td>
<td>0.16s</td>
<td>1.59s</td>
</tr>
<tr>
<td>K=105</td>
<td>84.9</td>
<td>94.07</td>
<td>90.8</td>
<td>94.79</td>
<td>95.11</td>
<td>93.13</td>
<td>95.14</td>
<td>82.1</td>
<td>87.62</td>
</tr>
</tbody>
</table>

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.

### 2.2 Structured Prediction: Factor Decomposition

#### 2.2.1 Problem Setup

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

$$
\min_w \frac{1}{2} \|w\|^2 + C \sum_{i=1}^n L(w; x_i, y_i).
$$

(2.24)

In case of a Structural SVM [49, 50], we consider the structured hinge loss

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

(2.25)

where $\delta(y, y_i)$ is a task-dependent error function, for which the Hamming distance $\delta_H(y, y_i)$ is commonly used. Since the size of domain $|\mathcal{Y}(x)|$ typically grows exponentially with the number of output variables, the tractability of problem (2.24) lies in the decomposition of the responses $\langle w, \phi(x, y) \rangle$ 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 $f$
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.

<table>
<thead>
<tr>
<th>Data</th>
<th>FastXML</th>
<th>LEML</th>
<th>IvsA-Logi</th>
<th>IvsA-SVM</th>
<th>1vsA-Logi</th>
<th>PD-Sparse</th>
<th>SLEEC</th>
</tr>
</thead>
<tbody>
<tr>
<td>LSHTC-wiki</td>
<td>10442s</td>
<td>217190s</td>
<td>&gt;10y</td>
<td>&gt;96d</td>
<td>&gt;10y</td>
<td>12486s/7s</td>
<td>2224000s</td>
</tr>
<tr>
<td>N=2355436</td>
<td>8.9G</td>
<td>10.4G</td>
<td>≈ 426G</td>
<td>≈ 870G</td>
<td>≈ 358M</td>
<td>685M</td>
<td>12.6G</td>
</tr>
<tr>
<td>D=2085167</td>
<td>164.8s</td>
<td>2896s</td>
<td>N/A</td>
<td>N/A</td>
<td>N/A</td>
<td>15.56s</td>
<td>8906s</td>
</tr>
<tr>
<td>K=320338</td>
<td>78.28</td>
<td>28.46</td>
<td>N/A</td>
<td>N/A</td>
<td>N/A</td>
<td>89.3</td>
<td>73.44</td>
</tr>
<tr>
<td>EUR-Lex</td>
<td>317s</td>
<td>17471s</td>
<td>22551s</td>
<td>32275s</td>
<td>32531s</td>
<td>434.9s</td>
<td>2443s</td>
</tr>
<tr>
<td>N=15643</td>
<td>324.5M</td>
<td>78M</td>
<td>257M</td>
<td>118M</td>
<td>14M</td>
<td>8.0M</td>
<td>80.8M</td>
</tr>
<tr>
<td>D=5000</td>
<td>0.996s</td>
<td>42.24s</td>
<td>7.93s</td>
<td>7.23s</td>
<td>1.39s</td>
<td>1.089s</td>
<td>4.89s</td>
</tr>
<tr>
<td>K=3956</td>
<td>67.3</td>
<td>67.82</td>
<td>77.3</td>
<td>64.5</td>
<td>73.8</td>
<td>76.3</td>
<td>74.2</td>
</tr>
<tr>
<td>RCV1-regions</td>
<td>94.06s</td>
<td>2247s</td>
<td>79.27s</td>
<td>14.73s</td>
<td>84.74s</td>
<td>8.82s</td>
<td>1129s</td>
</tr>
<tr>
<td>N=20835</td>
<td>14.61M</td>
<td>205M</td>
<td>129M</td>
<td>39M</td>
<td>504K</td>
<td>1.7M</td>
<td>204M</td>
</tr>
<tr>
<td>D=47237</td>
<td>0.824s</td>
<td>2.515s</td>
<td>0.486s</td>
<td>0.392s</td>
<td>0.174s</td>
<td>0.115s</td>
<td>15.8s</td>
</tr>
<tr>
<td>K=225</td>
<td>93.28</td>
<td>96.28</td>
<td>90.96</td>
<td>95.98</td>
<td>94.7</td>
<td>96.54</td>
<td>91</td>
</tr>
<tr>
<td>bibtex</td>
<td>18.35s</td>
<td>157.9s</td>
<td>8.944s</td>
<td>3.24s</td>
<td>13.97s</td>
<td>5.044s</td>
<td>298s</td>
</tr>
<tr>
<td>N=5991</td>
<td>27M</td>
<td>8.6M</td>
<td>3.7M</td>
<td>3.3M</td>
<td>412K</td>
<td>68K</td>
<td>26.7M</td>
</tr>
<tr>
<td>D=1837</td>
<td>0.09s</td>
<td>0.221s</td>
<td>0.038s</td>
<td>0.079s</td>
<td>0.0238s</td>
<td>0.0059s</td>
<td>0.94s</td>
</tr>
<tr>
<td>K=159</td>
<td>64.14</td>
<td>64.01</td>
<td>62.65</td>
<td>58.46</td>
<td>61.16</td>
<td>64.55</td>
<td>65.09</td>
</tr>
</tbody>
</table>

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

<table>
<thead>
<tr>
<th>Data sets</th>
<th>$k_A$</th>
<th>$k_W$</th>
</tr>
</thead>
<tbody>
<tr>
<td>EUR-Lex (K=3956)</td>
<td>20.73</td>
<td>45.24</td>
</tr>
<tr>
<td>LSHTC-wiki (K=320338)</td>
<td>18.24</td>
<td>20.95</td>
</tr>
<tr>
<td>LSHTC (K=12294)</td>
<td>7.15</td>
<td>4.88</td>
</tr>
<tr>
<td>aloi.bin (K=1000)</td>
<td>3.24</td>
<td>0.31</td>
</tr>
<tr>
<td>bibtex (K=159)</td>
<td>18.17</td>
<td>1.94</td>
</tr>
<tr>
<td>Dmoz (K=11947)</td>
<td>5.87</td>
<td>0.116</td>
</tr>
</tbody>
</table>

involves the variable $j$. Typically, a set of factor templates $T$ exists so that factors of the same template $F \in T$ share the same feature map $\phi_F(.)$ and parameter vector $w_F$. Then the response on input-output pair $(x, y)$ is given by

$$
\langle w, \phi(x, y) \rangle = \sum_{F \in T} \sum_{f \in F(x)} \langle w_F, \phi_F(x_f, y_f) \rangle,
$$

(2.26)

where $F(x)$ denotes the set of factors on $x$ that share a template $F$, and $y_f$ denotes output variables relevant to factor $f$ of domain $Y_f = Y_F$. We will use $F(x)$ to denote the union of factors of different templates $\{F(x)\}_{F \in T}$. Figure 2.1 shows two examples that both have two factor templates (i.e. unigram and bigram) for which the responses have decomposition

$$
\sum_{f \in u(x)} \langle w_u, \phi_u(x_f, y_f) \rangle + \sum_{f \in b(x)} \langle w_b, \phi_b(y_f) \rangle.
$$

Unfortunately, even with such decomposition, the maximization in (2.25) 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 (2.25) obtained from relaxation [14, 51]. 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 $|Y_f|$ times the number of factors $|F|$. This results in a prohibitive computational cost for problems with large output domain. As in Figure 2.1, one example has a factor domain $|Y_b|$ which grows quadratically with the size of output domain; the other has the number of factors $|F|$ which grows quadratically with the number of outputs. A key observation of this paper is, in contrast to the structural maximization (2.25) that requires larger extent of exploration on locally suboptimal assignments in order to achieve global optimality, the Factorwise Maximization Oracle (FMO)

$$
y_f^* := \arg \max_{y_f} \langle w_F, \phi(x_f, y_f) \rangle
$$

(2.27)

can be realized in a more efficient way by maintaining data structures on the factor parameters $w_F$. In the next section, we develop globally-convergent algorithms that rely only on FMO, and provide realizations
We consider an upper bound of the loss (2.25) 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 [52, 51]:

$$L^{LP}(w; x, \tilde{y}) = \max_{(q, p) \in M_L} \sum_{f \in F(x)} \langle \theta_f(w), q_f \rangle$$

(2.28)

where $\theta_f(w) := (\langle w_F, \phi_F(x, y_f) - \phi_F(x, \tilde{y}_f) \rangle + \delta_f(y_f, \tilde{y}_f))_{y_f \in Y}$, $M_L$ is a polytope that constrains $q_f$ in a $|Y_f|$-dimensional simplex $\Delta^{|Y_f|}$ and also enforces local consistency:

$$M_L := \left\{ q = (q_f)_{f \in F(x)}, p = (p_f)_{f \in V(x)} \mid q_f \in \Delta^{|Y_f|}, \forall f \in F(x), \forall F \in T \right\}$$

where $M_{j,f}$ is a $|Y_f|$ by $|Y_f|$ matrix that has $M_{j,f}(y_j, y_f) = 1$ if $y_j$ is consistent with $y_f$ (i.e. $y_j = [y_f]_j$) and $M_{j,f}(y_j, y_f) = 0$ otherwise. For a tree-structured graph $G(F, V, E)$, the LP relaxation is tight and thus loss (2.28) is equivalent to (2.25). For a general factor graph, (2.28) is an upper bound on the original loss (2.25). It is observed that parameters $w$ learned from the upper bound (2.28) tend to tightening the LP relaxation and thus in practice lead to tight LP in the testing phase [14]. Instead of solving LP (2.28) as a subroutine, a recent attempt formulates (2.24) as a problem that optimizes $(p, q)$ and $w$ jointly via dual decomposition [52, 53]. We denote $\lambda_{j,f}$ as dual variables associated with constraint $M_{j,f}q_f = p_f$, and $\lambda_f := (\lambda_{j,f})_{j \in N(f)}$ where $N(f) = \{ j \mid (j, f) \in E \}$. We have

$$L^{LP}(w; x, \tilde{y}) = \min_{q, p} \sum_{f \in F(x)} \langle \theta_f(w), q_f \rangle + \sum_{j \in N(f)} \langle \lambda_{j,f} , M_{j,f}q_f - p_f \rangle$$

(2.29)

$$= \min_{\lambda \in \Lambda} \sum_{f \in F(x)} \max_{q_f \in \Delta^{|Y_f|}} \langle \theta_f(w) + \sum_{j \in N(f)} \lambda_{j,f} [y_f]_j \rangle$$

(2.30)

$$= \min_{\lambda \in \Lambda} \sum_{f \in F(x)} L_f(w; x_f, \tilde{y}_f, \lambda_f)$$

(2.31)

where (2.30) follows the strong duality, and the domain $\Lambda = \{ \lambda \mid \sum_{(j,f) \in E(x)} \lambda_{j,f} = 0, \forall j \in V(x) \}$ follows the maximization w.r.t. $p$ in (2.29). The result (2.32) is a loss function $L_f(.)$ that penalizes the response of each factor separately given $\lambda_f$. The ERM problem (2.24) can then be expressed as

$$\min_{w, \lambda \in \Lambda} \sum_{F \in T} \left( \frac{1}{2} \| w_F \|^2 + C \sum_{f \in F} L_f(w_F; x_f, \tilde{y}_f, \lambda_f) \right).$$

(2.33)
We discuss two greedy algorithms that suit two different cases in the following.

Algorithm 2 Greedy Direction Method of Multiplier

0. Initialize \( t = 0 \), \( \alpha^0 = 0 \), \( \lambda^0 = 0 \) and \( \mathcal{A}^0 = \mathcal{A}^{\text{init}} \).

for \( t = 0, 1, \ldots \) do

1. Compute \((\alpha^{t+1}, \mathcal{A}^{t+1})\) via one pass of Algorithm 3, 4, or 1.

2. \( \lambda_{jf}^{t+1} = \lambda_{jf}^t + \eta (M_{jf} \alpha_{jf}^{t+1} - \alpha_{jf}^t), \quad j \in \mathcal{N}(f), \forall f \in \mathcal{F}. \)

end for

where \( \mathcal{F} = \bigcup_{i=1}^{N} F(x_i) \) and \( \mathcal{F} = \bigcup_{f \in \mathcal{T}} F \). The formulation (2.33) has an insightful interpretation: each factor template \( F \) learns a multiclass SVM given by parameters \( w_F \) from factors \( f \in F \), while each factor is augmented with messages \( \lambda_f \) passed from all variables related to \( f \).

Greedy Direction Method of Multiplier Let \( \alpha_f(y_f) \) be dual variables for the factor responses \( z_f(y_f) = \langle w, \phi(x_f, y_f) \rangle \) and \( \{ \alpha_j \}_{j \in \mathcal{V}} \) be that for constraints in \( \Lambda \). The dual problem of (2.33) can be expressed as

\[
\min_{\alpha_f \in \Delta^{[\mathcal{Y}_f]}} G(\alpha) := \frac{1}{2} \sum_{F \in \mathcal{T}} \| w_F(\alpha) \|^2 - \sum_{j \in \mathcal{V}} \delta_j^T \alpha_j \\
\text{s.t.} \quad M_{jf} \alpha_f = \alpha_j, \quad j \in \mathcal{N}(f), \quad f \in \mathcal{F}. \tag{2.34}
\]

where \( \alpha_f \) lie in the shifted simplex

\[
\Delta^{[\mathcal{Y}_f]} := \left\{ \alpha_f \mid \alpha_f(y_f) \leq C, \quad \alpha_f(y_f) \leq 0, \quad \forall y_f \neq \bar{y}_f, \quad \sum_{y_f \in \mathcal{Y}_f} \alpha_f(y_f) = 0 \right\}. \tag{2.35}
\]

Problem (2.34) can be interpreted as a summation of the dual objectives of \(|T|\) multiclass SVMs (each per factor template), connected with consistency constraints. To minimize (2.34) one factor at a time, we adopt a Greedy Direction Method of Multiplier (GDMM) algorithm that alternates between minimizing the Augmented Lagrangian function

\[
\min_{\alpha_f \in \Delta^{[\mathcal{Y}_f)}} L(\alpha, \lambda^t) := G(\alpha) + \frac{\rho}{2} \sum_{j \in \mathcal{N}(f), f \in \mathcal{F}} \| m_{jf}(\alpha, \lambda^t) \|^2 - \| \lambda_{jf}^t \|^2 \tag{2.36}
\]

and updating the Lagrangian Multipliers (of consistency constraints)

\[
\lambda_{jf}^{t+1} = \lambda_{jf}^t + \eta (M_{jf} \alpha_f - \alpha_j), \quad \forall j \in \mathcal{N}(f), \quad f \in \mathcal{F}, \tag{2.37}
\]

where \( m_{jf}(\alpha, \lambda^t) = M_{jf} \alpha_f - \alpha_j + \lambda_{jf}^t \) plays the role of messages between \(|T|\) multiclass problems, and \( \eta \) is a constant step size. The procedure is outlined in Algorithm 2. The minimization (2.36) 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.

Factor of Large Domain For problems with large factor domains, we minimize (2.36) via a variant of Frank-Wolfe algorithm with away steps (AFW) [54], outlined in Algorithm 3. The AFW algorithm maintains the iterate \( \alpha^t \) as a linear combination of bases constructed during iterates

\[
\alpha^t = \sum_{v \in \mathcal{A}^t} c_v^t v, \quad \mathcal{A}^t := \{ v \mid c_v^t \neq 0 \} \tag{2.38}
\]

\( \alpha_f \) is also dual variables for responses on unigram factors. We define \( \mathcal{U} := \mathcal{V} \) and \( \alpha_f := \alpha_j, \forall f \in \mathcal{U}. \)
Frank-Wolfe

We minimize (2.36) via one pass of

where we choose between two descent directions

d

which corresponds to the basis that leads to the most descent amount when being removed. Then the

y

where

Q

version in Appendix-A (Algorithm 1).

version since it turns out to be much more efficient. We include a detailed description on the BCFW

domains, for which Algorithm 3 does not yield any computational advantage. For this type of problem,

Many structured prediction problems, such as alignment, segmentation, and

Large Number of Factors

difficult [55], which prohibits its use in our analysis. In our implementation, however, we adopt the

f

steps (2-5) in Algorithm 3 can be computed in

O

, which can be much smaller than

|Y

nnz

when output domain is large. In practice, a

Block-Greedy Coordinate Descent (BGCD)

Algorithm 4 Block-Greedy Coordinate Descent

repeat

1. Find \(v^+\) satisfying (2.39).
2. Find \(v^-\) satisfying (2.40).
3. Compute \(\alpha^{t+1}\) according to (2.41).
4. Maintain active set \(\mathcal{A}^t\) by (2.38).
5. Maintain \(w_F(\alpha)\) according to (2.34).
until a non-drop step is performed.

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

\[
v_f^+ := \arg\min_{v_f \in \Delta |y_f|} \langle \nabla_{\alpha_f} \mathcal{L}(\alpha^t, \lambda^t), v_f \rangle = C(e_{y_f} - e_{\bar{y}_f}), \quad \forall f \in F
\]

(2.39)

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

\[
v^- := \arg\max_{v \in \mathcal{A}^t} \langle \nabla_{\alpha} \mathcal{L}(\alpha^t, \lambda^t), v \rangle,
\]

(2.40)

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

\[
\alpha^{t+1} := \begin{cases} 
\alpha^t + \gamma_F d_F, & \langle \nabla_{\alpha} \mathcal{L}, d_F \rangle < \langle \nabla_{\alpha} \mathcal{L}, d_A \rangle \\
\alpha^t + \gamma_A d_A, & \text{otherwise.}
\end{cases}
\]

(2.41)

where we choose between two descent directions \(d_F := v^+ - \alpha^t\) and \(d_A := \alpha^t - v^-\). The step size of each direction \(\gamma_F := \arg\min_{\gamma \in [0,1]} \mathcal{L}(\alpha^t + \gamma d_F)\) and \(\gamma_A := \arg\min_{\gamma \in [0, c_{v^-}] \setminus \{0\}} \mathcal{L}(\alpha^t + \gamma d_A)\) can be computed exactly due to the quadratic nature of (2.36). A step is called drop step if a step size \(\gamma = c_{v^-}\) is chosen, which leads to the removal of a basis \(v^-\) 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 3 stops only if a non-drop step is performed. Note Algorithm 3 requires only a factorwise greedy search (2.39) instead of a structural maximization (2.25). 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 3 can be computed in \(O(|\mathcal{A}_f| \text{nnz}(\phi_f))\), where \(|\mathcal{A}_f|\) is the number of active states in factor \(f\), which can be much smaller than \(|Y_f|\) when output domain is large.

In practice, a Block-Greedy Coordinate Frank-Wolfe (BCFW) method has much faster convergence than Frank-Wolfe method (Algorithm 3) [55, 17], but proving linear convergence for BCFW is also much more difficult [55], 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. 2.1, right), comprise binary variables and large number of factors with small domains, for which Algorithm 3 does not yield any computational advantage. For this type of problem, we minimize (2.36) via one pass of Block-Greedy Coordinate Descent (BGCD) (Algorithm 4) instead. Let \(Q_{\max}\) be an upper bound on the eigenvalue of Hessian matrix of each block \(\nabla^2_{\alpha_f} \mathcal{L}(\alpha)\). For binary variables of pairwise factor, we have \(Q_{\max} = 4(\max_{f \in F} \|\phi_f\|^2 + 1)\). Each iteration of BGCD finds a

\[
\text{Algorithm 3 Away-step Frank-Wolfe (AFW)}
\]

\[
\text{Algorithm 4 Block-Greedy Coordinate Descent}
\]

repeat

1. Find \(v^+\) satisfying (2.39).
2. Find \(v^-\) satisfying (2.40).
3. Compute \(\alpha^{t+1}\) according to (2.41).
4. Maintain active set \(\mathcal{A}^t\) by (2.38).
5. Maintain \(w_F(\alpha)\) according to (2.34).
until a non-drop step is performed.

for \(i \in [n]\) do

1. Find \(f^+\) satisfying (2.42).
2. \(\mathcal{A}_f^{t+1} = \mathcal{A}_f^t \cup \{f^+\}\).
3.1 Update \(\alpha_f\) according to (2.43).
3.2 Maintain \(w_{F'}(\alpha)\) via (2.34).
end for

end for
factor that leads to the most progress
\[ f^* := \arg\min_{f \in \mathcal{F}(x_i)} \left( \min_{\alpha_j + d \in \Delta_j} \langle \nabla_{\alpha_j} \mathcal{L}(\alpha^t, \lambda^t), d \rangle + \frac{Q_{\max}}{2} ||d||^2 \right). \] (2.42)
for each instance \( x_i \), adds them into the set of active factors \( \mathcal{A}_i \), and performs updates by solving block subproblems
\[ d^*_f = \arg\min_{\alpha_j + d \in \Delta_j} \langle \nabla_{\alpha_j} \mathcal{L}(\alpha^t, \lambda^t), d \rangle + \frac{Q_{\max}}{2} ||d||^2 \] (2.43)
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 \(|\mathcal{F}(x_i)|\) in practice. We address in the next section how a joint FMO can be performed to compute (2.42) in time sublinear to \(|\mathcal{F}(x_i)|\) in the binary-variable case.

**Convergence Analysis** The analysis leverages recent analysis on the global linear convergence of Frank-Wolfe variants [54] for function of the form (2.36) with a polyhedral domain, and also the analysis in [56] for Augmented Lagrangian based method. This type of greedy Augmented Lagrangian Method was also analyzed previously under different context [57, 58, 59].

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 [17], which outperforms other competitors such as Cutting-Plane, FW, and online-EG methods in [17]. (ii) SSG: an implementation of the Stochastic Subgradient method [60]. (iii) Soft-BCFW: Algorithm proposed in ([53]), which avoids structural oracle by minimizing an approximate objective, where a parameter \( \rho \) 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 [17] into C++, which is an order of magnitude faster. All other

\[ \Delta^t_d := d^* - d^t, \quad \Delta^t_p := \mathcal{L}(\alpha^t, \lambda^t) - d^t \] (2.44)
be the dual and primal suboptimality of problem (2.34) respectively. We have the following theorems.

**Theorem 2** (Convergence of GDMM with AFW). The iterates \( \{ (\alpha^t, \lambda^t) \}_{t=1}^\infty \) produced by Algorithm 2 with step 1 performed by Algorithm 3 has
\[ E[|\Delta^t_p + \Delta^t_d|] \leq \epsilon \text{ for } t \geq \omega \log\left( \frac{1}{\epsilon} \right) \] (2.45)
for any \( 0 < \eta \leq \frac{\rho}{4 + 10(1 + \nu_m)\mu_M} \) with \( \omega = \max \left\{ 2(1 + \frac{mQ(1 + \nu)}{\mu M}), \frac{\tau}{\eta} \right\} \), where \( \mu_M \) is the generalized geometric strong convexity constant of (2.36), \( Q \) is the Lipschitz-continuous constant for the gradient of objective (2.36), and \( \tau > 0 \) is a constant depending on optimal solution set.

**Theorem 3** (Convergence of GDMM with BGCD). The iterates \( \{ (\alpha^t, \lambda^t) \}_{t=1}^\infty \) produced by Algorithm 2 with step 1 performed by Algorithm 4 has
\[ E[|\Delta^t_p + \Delta^t_d|] \leq \epsilon \text{ for } t \geq \omega_1 \log\left( \frac{1}{\epsilon} \right) \] (2.46)
for any \( 0 < \eta \leq \frac{\rho}{4(1 + \nu_m)\mu_1} \) with \( \omega_1 = \max \left\{ 2(1 + \frac{Q_{\max}}{\mu_1}), \frac{\tau}{\eta} \right\} \), where \( \mu_1 \) is the generalized strong convexity constant of objective (2.36) and \( Q_{\max} = \max_{f \in \mathcal{F}} Q_f \) is the factorwise Lipschitz-continuous constant on the gradient.

### 2.2.3 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 [17], which outperforms other competitors such as Cutting-Plane, FW, and online-EG methods in [17]. (ii) SSG: an implementation of the Stochastic Subgradient method [60]. (iii) Soft-BCFW: Algorithm proposed in ([53]), which avoids structural oracle by minimizing an approximate objective, where a parameter \( \rho \) 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 [17] 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 $w$) 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 treebank⁶ that contains 3,808 sentences, 196,223 words, and 45 POS labels. The **HIT-MW⁷** **ChineseOCR** dataset is a hand-written Chinese character dataset from [61]. 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-regions⁸** and **EUR-Lex** (directory codes)⁹. 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 [52, 53]. **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 2.2. Due to the approximation (controlled by $\rho$), Soft-BCFW can converge to a suboptimal primal objective value. While the gap decreases as $\rho$ 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 2.2 (right). More general experiments are showed in Figure 2.3. 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. Therefore, GDMM-subFMO achieves

---

⁶https://catalog.ldc.upenn.edu/LDC99T42
⁷https://sites.google.com/site/hitmwbdb/
⁸http://www.csie.ntu.edu.tw/~cjlin/libsvmtools/datasets/multilabel.html
⁹mulan.sourceforge.net/datasets-mlc.html
orders-of-magnitude speedup over other methods. 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.

2.3 Proposed Works

2.3.1 Deep Extreme Classification via Loss Decomposition

![Figure 2.4: Decomposing loss on a high-dimensional product into losses on several low-dimension products that are amenable to MIPS.](image)

As discussed in section 2.1, the successfulness of the Primal-Dual Sparse technique requires $DK \gg N$ (or equivalently, $D \gg N/K$), which is typically true if $D \gg 100$. However, many deep learning models have embeddings of size $128 \sim 512$ as the input of last layer. This is an awkward range of dimensions that neither high enough for an efficient primal-dual sparse representation nor low enough for an accurate Maximum Inner Product Search (MIPS).

Here we propose the use of Dual-Decomposed Loss introduced in section 2.2 as a complimentary technique to MIPS when applied to the training of Extreme Classifier. In particular, to allow efficient MIPS, we decompose an high-dimensional inner product into the summation of several low-dimensional inner products:

$$\theta := \langle W, x \rangle = \sum_{f=1}^{F} \langle W_f, x_f \rangle = \sum_{f=1}^{F} \theta_f.$$  

Then through the convex-conjugate representation of a loss function, we can decompose a classification loss as follows:

$$L_y(\theta) := \max_{\alpha} \left( \sum_{f=1}^{F} \theta_f, \alpha \right) - L_y^*(\alpha)$$

$$= \max_{\alpha_f, \alpha_f = \alpha} \left( \sum_{f=1}^{F} \left( \langle \theta_f, \alpha_f \rangle - \frac{1}{F} L_y^*(\alpha_f) \right) \right)$$

$$= \min_{\lambda, \sum_f \lambda_f = 0} \left( \sum_{f=1}^{F} \max_{\alpha_f} \left( \langle \theta_f + \lambda_f, \alpha_f \rangle - \frac{1}{F} L_y^*(\alpha_f) \right) \right)$$

$$= \min_{\lambda, \sum_f \lambda_f = 0} 1/F \sum_{f=1}^{F} L_y((\theta_f + \lambda_f)F)$$  

As shown in Figure 2.4, this results in several loss functions of low-dimensional matrix vector products, and thus are amenable to MIPS with high precision and recall. Furthermore, due to the simple factor decomposition, the minimization w.r.t. the message variables in (2.47) has closed-form solution. Therefore, we can perform a Greedy Block Coordinate Descent w.r.t. the message variables $\lambda$. The algorithm retrieves top candidates incrementally until the (projected) gradient w.r.t. the message variable varnishes, which allows us to evaluate loss only only classes retrieved by MIPS on each low-dimensional losses. In our
preliminary experiments, we found even a single iteration of message passing results in solutions of 15% higher accuracy than that obtained by directly applying MIPS to the training.

2.3.2 Learning Sparse Layers in Neural Network

With the success of sparse optimization techniques in Extreme Classification, it is natural to ask: if one can extend such technique from the last layer of Neural Network to the intermediate hidden layers. In particular, each layer of a neural network can be treated as a multilabel predictor that takes input from the output of previous layer, and the number of neurons in the layer can be interpreted as the number of output classes. The only difference is that the correct output of each layer is not given as labels. Instead, they are given in the form of \textit{back-propagated gradients} when applying chain rules to the network. Then one can have significant speedup if:

- The output of each layer is sparse, which corresponds to the \textit{dual sparsity} in the classification problem.
- Given that the back-propagated gradient of an output is sparse, there exists a \textit{sparse weight matrix} mapping from input to output that fits the direction of the back-propagated gradient.

Let $D, K$ denote the input and output dimensions of a layer. In addition, let $d_A, k_A$ be the number of sparse outputs of previous layer and this layer respectively, and $d_W, k_W$ be the number of non-zero weights on each column and row of the weight matrix respectively. If the above two conditions hold, one can:

1. Reduce the cost of forward propagation in a particular layer from $O(DK)$ to $O(d_A k_W)$, and
2. Reduce the cost of back propagation from $O(DK)$ to $O(d_W k_A + d_A k_A)$.

Note the reduction of costs in these cases are even more than that in the case of classification, since the sparsification is happening on both input and output neurons.

To induce a (approximately) sparse output of each layer, a potential candidate method is to use the multiclass sigmoid activation function $\sigma : \mathbb{R}^K \rightarrow \mathbb{R}^K$

$$[\sigma(z)]_k := \frac{\exp(z_k)}{\sum_{j=1}^{K} \exp(z_j)}$$

which generalizes the binary sigmoid function $\sigma(z) = 1/(1 + \exp(-z))$ to encourage concentration of activations to a small number of most active neurons. Another candidate activation function is the top-$R$ ranking function:

$$[\sigma_R(z)]_k := \begin{cases} z_k & z_k \in \text{top}_R(z) \\ 0 & \text{o.w.} \end{cases}$$

which can be seen as a generalization of the Relu function from single neuron to multiple neurons.
Chapter 3

Sparse Estimator for Latent-Variable Models

In this chapter, we first briefly describe our work [62] in section 3.1, which designs the first polynomial-time estimator with approximation guarantees for Latent-Feature Model by formulating it as a high-dimensional sparse optimization problem. Then we propose extending the methodology to handle any Matrix Factorization of binary components with arbitrary loss functions in section 3.2.

3.1 Latent-Feature Models

3.1.1 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 \( e \). We thus obtained the vectorized form of the model as \( X = ZW + e \).

Most existing works on LFM make two strong assumptions. The first is that the model has zero bias \( E[e|z] = 0 \) [29, 30, 63, 64, 31, 32, 66]. The second common but strong class of assumptions is distributional [66, 29]:

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

where \( Bern(\pi) \) denotes the distribution of \( K \) independent Bernoulli with \( z_k \sim Bern(\pi_k) \). In the Nonparametric Bayesian setting [63, 65, 32, 31, 30], one replaces \( Bern(\pi) \) with an Indian Buffet Process IBP(\( \alpha \)) 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 [63, 30, 65, 31, 66], the goal is to infer the posterior distribution \( P(Z, W|X) \) given \( X \). The line of work using Spectral Methods [29] on the other hand aim to estimate \( p(z), p(x|z) \) in turn by estimating parameters \( (\pi, W) \). In some other work [64], 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

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

(3.1)

where the expectation is over the random observation $x$.

### 3.1.2 Latent Feature Lasso

We first consider the non-convex formulation that was also previously studied in [30] as asymptotics of

$$x$$

where the expectation is over the random observation $x$ with

$$S$$

where $M$ is a convex minimization problem. Applying Lagrangian duality to (3.3) results in the following

$$L$$

and

$$A$$

The estimation problem in [64] could also be cast in the above form with

$$\{ \text{value} \}

$$

3.1.2 Latent Feature Lasso

We first consider the non-convex formulation that was also previously studied in [30] as asymptotics of the MAP estimator of IBP Linear-Gaussian model:

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

(3.2)

The estimation problem in [64] could also be cast in the above form with $\lambda = 0$ and $K$ treated as a fixed hyper-parameter, while [30] treats $K$ as a variable and controls it through $\lambda$. (3.2) is a combinatorial optimization of $N \times K + 1$ integer variables. In the following we develop a tight convex approximation to (3.2) with $\ell_2$ regularization on $W$, by introducing a type of atomic norm [67].

For a fixed $K$, $Z$, consider the minimization over $W$ of the $\ell_2$ regularized version of (3.2)

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

(3.3)

which is a convex minimization problem. Applying Lagrangian duality to (3.3) results in the following dual form

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

(3.4)

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 (3.3) and (3.4), 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 (3.4) 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 (3.2) 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

$$\|M\|_S := \min_{c \geq 0} \sum_{a \in S} c_a \text{ s.t. } M = \sum_{a \in S} c_a a.$$

(3.5)

with $S := \{ zz^T | z \in \{0,1\}^N \}$. Note $\|M\|_S = \sum_{a \in S} c_a = K$ when $c_a$ in (3.5) are constrained at integer value $\{0,1\}$, and it serves a convex approximation to $K$ similar to the $\ell_1$-norm used in Lasso for the approximation of cardinality. This results in the following Latent Feature Lasso estimator

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

(3.6)

### 3.1.3 Algorithm

The estimator (3.6) seems intractable at first sight in part since the atomic norm involves a set $S$ of $2^N$ atoms. In this section, we study a variant of approximate greedy coordinate descent method for tractably solving problem (3.6). We begin by rewriting the optimization problem (3.6) as an $\ell_1$-regularized problem
Algorithm 5 A Greedy Algorithm for Latent Feature Lasso

[0:] \( \mathcal{A} = \emptyset, c = 0 \).

for \( t = 1 \ldots T \) do

[1:] Find a greedy atom \( zz^T \) by solving (3.8).

[2:] Add \( zz^T \) to an active set \( \mathcal{A} \).

[3:] Minimize (3.7) w.r.t. coordinates in \( \mathcal{A} \) via updates (3.9).

[4:] Eliminate \( \{ z_j z_j^T | c_j = 0 \} \) from \( \mathcal{A} \).

end for.

with \( \bar{K} = 2^N - 1 \) coordinates, by expanding the matrix \( M \) in terms of the \( \bar{K} \) atoms underlying the atomic norm \( \| \cdot \|_S \):

\[
\min_{c \in \mathbb{R}^{\bar{K}}} \begin{cases} 
    g \left( \sum_{j=1}^{K} c_j z_j z_j^T \right) + \lambda \|c\|_1 \\
    F(c) \end{cases} \quad (3.7)
\]

where \( \{ z_j \}_{j=1}^{K} \) enumerates all possible \( \{0, 1\}^N \) patterns except the 0 vector. Our overall algorithm is depicted in Algorithm 5. In each iteration, it finds

\[
j^* := \arg \max_j -\nabla_j f(c) \quad (3.8)
\]

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 (3.8). Then we minimize (3.7) over coordinates in \( \mathcal{A} \) by a sequence of proximal updates:

\[
c_r + 1 \leftarrow \left[ c_r - \frac{\nabla f(c_r) + \lambda}{\gamma |\mathcal{A}|} \right]_+ \quad r = 1 \ldots T_2 \quad (3.9)
\]

where \( \gamma \) is the Lipschitz-continuous constant of the coordinate-wise gradient \( \nabla c_j f(c) \).

**Computing coordinate-wise gradients.** By Danskin’s Theorem, the gradient of function \( f(c) \) takes the form

\[
\nabla c_j f(c) = z_j A^* A^T z_j / (2N^2 \tau), \quad (3.10)
\]

which in turn requires finding the maximizer \( A^* \) of (3.4).

**Computing \( A^* \).** By taking advantage of the strong duality between (3.4) and (3.3), the maximizer \( A^* \) can be found by finding the minimizer \( W^* \) of

\[
\min_W \frac{1}{2N} \| X - Z_A W \|_F^2 + \sum_{k \in \mathcal{A}} \frac{\tau}{2e_k} \| W_{k,:} \|_2^2 \quad (3.11)
\]

and computing \( A^* = (X - Z_A W^*) \), where \( Z_A \) 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 (3.11) of the form

$$W^* = (Z_A^T Z_A + N \tau^{-1}(c_A))^{-1} Z_A^T X.$$  \hspace{1cm} (3.12)

An efficient way of computing (3.12) is to maintain $Z_A^T Z_A$ and $Z_A^T X$ whenever the active set of atoms $A$ changes. This has a cost of $O(NDK_A)$ for a bound $K_A$ on the active size, which however is almost neglectable compared to the other costs when amortized over iterations. Then the evaluation of (3.12) would cost only $O(K_A^3 + K_A^2D)$ for each evaluation of different $c$. Similarly the matrix computation of (3.10) can be made more efficient as $\nabla_c f(c) \propto

$$((Z_A^T X - Z_A^T Z_A W^*)(Z_A^T X - Z_A^T Z_A W^*)^T)$$

can be computed in $O(K^2D + K^3)$ via the maintenance of $Z_A^T Z_A$, $Z_A^T X$.

The output of Algorithm 5 is the coefficient vector $c$, and with the resulting latent feature matrix $W(c)$ given by (3.12). 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 $\{c_k ||W_k||^2\}_{k \in A}$ 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 5) is to find the direction (3.8) 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 (3.8) 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 [68, 69].

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

$$\max_{z \in \{0,1\}^N} \langle C, zz^T \rangle$$  \hspace{1cm} (3.13)

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

$$\max_{y \in \{-1,1\}^N} \frac{1}{4} \left(\langle C, yy^T \rangle + 2\langle C, 1y^T \rangle + \langle C, 11^T \rangle \right).$$  \hspace{1cm} (3.14)

where 1 denotes $N$-dimensional vector of all 1s. By introducing a dummy variable $y_0$, (3.14) can be expressed as

$$\max_{(y_0;y) \in \{-1,1\}^{N+1}} \frac{1}{4} \left[ \begin{array}{c} y_0 \\ y \end{array} \right]^T \left[ \begin{array}{cc} 1^T C 1 & 1^T C \\ C 1 & C \end{array} \right] \left[ \begin{array}{c} y_0 \\ y \end{array} \right].$$  \hspace{1cm} (3.15)

Note that one can ensure finding a solution with $y_0 = 1$ by flipping signs of the solution vector to (3.15), since this does not change the quadratic form objective value. Denote the quadratic form matrix in (3.15) be $\hat{C}$. Problem of form (3.15) is a MAXCUT-like Boolean Quadratic problem, for which there is SDP relaxation of the form

$$\max_{Y \in \mathbb{S}^N} \langle \hat{C}, Y \rangle$$  \hspace{1cm} s.t. $Y \succeq 0$, $\text{diag}(Y) = 1$ \hspace{1cm} (3.16)

rounding from which guarantees a solution $\hat{y}$ to (3.15) satisfying

$$\bar{h} - h(\hat{y}) \leq \rho(\overline{h} - h)$$  \hspace{1cm} (3.17)

for $\rho = 2/5$ [70], where $h(y)$ denotes the objective function of (3.15) and $\overline{h}$, $\bar{h}$ 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 $\hat{C}$, $\bar{h} = 0$ and thus

$$-\nabla_j f(c) = h(\hat{y}) \geq \mu \overline{h} = \mu( -\nabla_j f(c))$$  \hspace{1cm} (3.18)
for $\mu = 1 - \rho = 3/5$, where $j$ is coordinate selected by rounding from a solution of (3.16) and $j^*$ is the exact maximizer of (3.8).

Finally, it is noteworthy that, although solving a general SDP is computationally expensive, SDP of the form (3.16) has been shown to allow much faster solver that has linear cost w.r.t. the matrix size $\text{nnz}(\tilde{C})$ [68, 69]. In our implementation we adopt the method of [69] due to its strong empirical performance.

**Convergence Analysis** The aim of this section is to show the convergence of Algorithm 5 under the approximation of greedy atom generation. In particular, we show the multiplicative approximation error incurred in the step (3.8) only contributes an additive approximation error proportional to $\lambda$, as stated in the following theorem.

**Theorem 4.** The greedy algorithm proposed (Algorithm 5) satisfies

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

where $c^*$ is any reference solution, $\mu = 3/5$ is the approximation ratio given by (3.18) and $\gamma$ 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 $\lambda$ of any reference solution $c^*$ scaled in main by its $\ell_1$ 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 5.** Let $\mathcal{A}^* \subseteq [K]$ be a support set and $c^* := \arg\min_{c:\text{supp}(c) = \mathcal{A}^*} F(c)$. Suppose $F(c)$ is strongly convex on $\mathcal{A}^*$ with parameter $\beta$. The solution given by Algorithm 5 satisfies

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

Let $K = 2^N$ be the size of the atomic set. Any target latent structure $Z^* W^*$ can be expressed as $Z D(c^*) \tilde{W}^*$ where $Z$ is an $N \times K$ dictionary matrix, $D(c^*)$ is a $K \times K$ diagonal matrix of diagonal elements $D_{kk} = c^*_k$ 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 5 shows that our algorithm has an iteration complexity of $O(K/\epsilon)$ to achieve $\epsilon$ error, with an additional error term proportional to $\lambda \sqrt{K}$ due to the approximation made in (3.18).

**Risk Analysis** In this section, we investigate the performance of the output from Algorithm 5 in terms of the population risk $r(\cdot)$ defined in (3.1). Given coefficients $c$ with support $\mathcal{A}$ obtained from algorithm (5) for $T$ iterations, we construct the weight matrix by $\tilde{W} = (\sqrt{c_A}) W^* + W^*(c_A) = \frac{1}{N} Z^T_A A^*$, where $A^*$ is the maximizer of (3.4) as a function of $c$. It can be seen that $\tilde{W}$ satisfies

$$F(c) = \frac{1}{2N} \|X - Z_A \tilde{W}\|_F^2 + \frac{\tau}{2} ||\tilde{W}||_F^2 + \lambda \|c_A\|_1.$$  

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

**Theorem 6.** Let $\tilde{W} = (\sqrt{c_A}) W^*(c_A)$ be the weight matrix obtained from $T$ iterations of Algorithm 5, and $\hat{W}$ be the minimizer of the population risk (3.1) with $K$ components and $||\hat{W}||_F \leq R$. We then have the following bound on population risk: $r(\tilde{W}) \leq r(\hat{W}) + \epsilon$ with probability $1 - \rho$ for

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

with $\lambda, \tau$ chosen appropriately as functions of $N$. 

26
Note the output of Algorithm 5 has number of components $\hat{K}$ bounded by number of iterations $T$. Therefore, Theorem (6) gives us a trade-off between risk and sparsity—one can guarantee to achieve $\epsilon$-suboptimal risk compared to the optimal solution of size $K$, via $O(K/\epsilon)$ components and $O(DK/\epsilon^3)$ samples. Notice the result (6) is obtained without any distributional assumption on $p(x)$ and $p(z)$ except that of boundedness. Comparatively, the theoretical result obtained from Spectral Method [29] requires the knowledge/assumption of the distribution $p(x|z), p(z)$, which is sensitive to model mis-specification in practice.

### 3.1.4 Experiments

![Figure 3.1: 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 $\hat{\Theta} = Z^*W^*$ and the estimated $\hat{\Theta} = \hat{Z}\hat{W}$.](image1)

![Figure 3.2: 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.](image2)

### Table 3.1: Data statistics.

<table>
<thead>
<tr>
<th>Dataset</th>
<th>$N$</th>
<th>$D$</th>
<th>$K$</th>
<th>$\sigma$</th>
<th>$\text{nnz}(W_{k,:})$</th>
</tr>
</thead>
<tbody>
<tr>
<td>Syn0</td>
<td>100</td>
<td>196</td>
<td>4</td>
<td>0</td>
<td>$\leq 8$</td>
</tr>
<tr>
<td>Syn1</td>
<td>1000</td>
<td>1000</td>
<td>35</td>
<td>0.01</td>
<td>1000</td>
</tr>
<tr>
<td>Syn2</td>
<td>1000</td>
<td>900</td>
<td>14</td>
<td>0.1</td>
<td>49</td>
</tr>
<tr>
<td>Syn3</td>
<td>1000</td>
<td>900</td>
<td>35</td>
<td>0.1</td>
<td>36</td>
</tr>
<tr>
<td>Tabletop</td>
<td>100</td>
<td>8560</td>
<td>4</td>
<td>n/a</td>
<td>n/a</td>
</tr>
<tr>
<td>Mnist1k</td>
<td>1000</td>
<td>777</td>
<td>n/a</td>
<td>n/a</td>
<td>n/a</td>
</tr>
<tr>
<td>YaleFace</td>
<td>165</td>
<td>2842</td>
<td>n/a</td>
<td>n/a</td>
<td>n/a</td>
</tr>
<tr>
<td>Yeast</td>
<td>1500</td>
<td>104</td>
<td>n/a</td>
<td>n/a</td>
<td>n/a</td>
</tr>
</tbody>
</table>

![Figure 3.3: Synthetic data (i.e. Syn1, Syn2, Syn3). The first row shows observations $X_{1,:}$, and the second row shows latent features $W_{k,:}$.](image3)
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 3.1. For the synthetic data experiments, we used a benchmark simulated dataset Syn0 that was also used in [30, 29]. 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 3.3 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 3.3, 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 $\sigma$ becomes larger. For real data, we use a benchmark Tabletop data set constructed by [27], 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**: $\min_{S: |S|=K} \left\| Z_{\cdot,j} - Z^*_{\cdot,j} \right\|^2_N / K$.
- **RMSE**: $\left\| Z^* W^* - Z W \right\|_F / \sqrt{NK}$.
- **RMSEnoisy**: $\left\| X - ZW \right\|_F / \sqrt{ND}$.

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 [31]. 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 [32]. We used implementation published by the author \(^4\). (c) **MF-Binary**: A Matrix Factorization with the Binary Components model [64], which has recovery guarantees in the noiseless case but has an $O(K2^K)$ 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 [30]. 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 [29]. 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 5).

The results are shown in Figure 3.1 and 3.2. 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 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

---
\(^1\)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://mlloss.org/software/view/185/
\(^5\)https://sites.google.com/site/slawskimartin/code
\(^6\)https://github.com/tbroderick/bp-means
(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.

3.2 Proposed Work

3.2.1 Matrix Factorization with Binary Components

We are interested in extending our methodology applied in *Latent-Feature Model* to any binary matrix factorization problems of the form:

$$
\min_{U \in \{0,1\}^{m \times k}, S \in \text{diag}(k), V \in \{0,1\}^{n \times k}} L(U S V^T, R).
$$

(3.20)

or the form with only one of the factor being binary:

$$
\min_{U \in \{0,1\}^{m \times k}, V \in \mathbb{R}^{n \times k}} L(U V^T, R).
$$

(3.21)

This has wide applications that we would like to experiment on. To name a few examples:

- **Recommendation**: here $R_{ij} \in \mathbb{R}$ denotes the ratings given by each user to each item and $L(\cdot, \cdot)$ denotes square loss penalized on non-zero positions of $R$.

- **Community Detection**: here $R_{ij} \in \{0,1\}$ denotes whether a link exists in a social network from the $i$th person to the $j$th person and $L(Z, R)$ denotes a cross entropy loss between the scores $Z$ and links $R$.

- **Word Embedding**: In this case, $R_{ij}$ denotes the log-frequency of word $j$ in a window centered at an occurrence of word $i$ and $L(\cdot, \cdot)$ denotes the square loss penalized on all entries, as proposed in the popular Glove word embedding model [71].

There are many reasons to learn a binary-valued factor instead of a real-valued factor. We list a couple of common ones as follows:

- **Interpretability**: binary-valued results can be often interpreted as binary relations such as ownership, belonging and appearance.

- **Prior knowledge**: In some applications such as community detection or blind-source identification, the factors interested are by itself binary-valued.

- **Semi-supervison**: Existing labels or tags in a semi-supervised learning task are usually binary-valued. If the factors in the factorization model are binary, one can easily enforce consistency between the learned factors and the ground-truth labels.

- **Computational reason**: Binary embedding is more efficient to store and compute in an inner-product operation, because (i) the number of non-zero entries is small, and (ii) each non-zero entry only requires 1-bit to store, giving a speedup (compression rate) of 64.

To our knowledge, the high-dimensional sparse estimator and greedy algorithm proposed in the previous section can be extended to the general cases (3.20), (3.21) in a straight-forward way, so it remains to be verified whether factors learned this way could perform significantly better than existing approaches.
Chapter 4

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 estimators of computational complexity sublinear to the domain size while guaranteeing optimality of the solution. We demonstrate the power of such scheme through the development of state-of-the-art algorithms in Extreme Structured and Non-structured Classification, and we propose to extend the algorithms to speed up not only the last but also the intermediate hidden layers in a Neural Network. We also design the first polynomial-time estimator for Latent-Feature Models, and expect to extend the methodology to estimate any Matrix Factorization problem with binary components.
Bibliography


