Compressed Learning

University of Technology, Sydney

Tianyi Zhou

Faculty of Engineering and Information Technology

University of Technology, Sydney

A thesis submitted for the degree of

Doctor of Philosophy

31 August 2013

To my loving parents
Yongxi Zhou and *Jingping Yao*

CERTIFICATE OF ORIGINAL AUTHORSHIP

I certify that the work in this thesis has not previously been submitted for a degree nor has it been submitted as part of requirements for a degree except as fully acknowledged within the text.

I also certify that the thesis has been written by me. Any help that I have received in my research work and the preparation of the thesis itself has been acknowledged. In addition, I certify that all information sources and literature used are indicated in the thesis.

Student: Tianyi ZHOU
Date: 31/08/2013

Production Note:
Signature removed prior to publication.

Acknowledgements

I benefited and learned a lot from my advisor, several professors, my friends, my colleagues and my family during PhD study in University of Technology, Sydney and Nanyang Technological University. I would like to take this good opportunity to appreciate their significant helps to me.

The first person I would like to express my appreciation and gratitude to is Professor Dacheng Tao. I am so lucky to have Prof. Tao as my professional mentor and academic advisor. I benefited significantly from various detailed discussions with him. Discussing a problem with him has been always a humbling but eye-opening experience, and he always gives me sufficient freedom to think and explore. His careful criticisms bridging both theory and application have greatly broadened my thought in research. His vision, creativeness and enthusiasm in solving challenging problems has greatly encouraged me and inspired my works. Without his high scientific criterion, endless patience, generous support, and constant guidance, this thesis cannot be accomplished. I also want to thank Prof. Tao as a nice friend, for his invaluable suggestions and experienced instructions on my research career and life.

I also wish to express my appreciation to other Professors who have gave me helpful guidance and encouragement during my PhD study and on conferences: Prof. Chengqi Zhang, Prof. Xindong Wu, Prof. Xingquan Zhu, Prof. Jieping Ye, Prof. Jerome H. Friedman, Prof. Trevor Hastie, Prof. Dean P. Foster and Prof. Emmanuel Candès. I learned a lot from discussions with them and their attitude for doing high-quality research.

I have been fortunate to work in a group gathering the most brilliant researchers and best friends in the past 5 years: Dr. Wei Bian, Dr. Jun Li, Dr. Bo Geng, Dr. Xinmei Tian, Dr. Zhang Zhang, Dr. Chao Zhang, Dr. Naiyang Guan, Dr. Lusong Li, Dr. Xiaoguang Rui, Dr. Weifeng Liu, Dr, Shengzheng Wang, Yang Mu, Bo Xie, Dongjing Song, Si Si, Yuanyuan Fu, Yong Luo, Yangxi Li, Zhibin Hong, Nannan Wang, Mingming Gong, Lianyang Ma, Maoying Qiao, Xiaoyan Li, Tongliang Liu, Fei Gao, Changxin Ding, Jie Gui, Xinchao Wang, Tianhao Zhang. Especially, I am deeply indebted to Dr Wei Bian, who gave me strong motivation and critical guidance when I have meet difficulties in research. He spent much time to teach me valuable things that I cannot easily learn by myself, especially at the beginning of my PhD study. I learned lots from discussion and collaboration with all group members in statistics, machine learning and optimization. Moreover, I enjoyed the invaluable friendships with them, their kindly support and accompany are always my source of strength and courage in both research and daily life. I own my deepest thanks to all of them!

I am also grateful to all the other friends who made my 5 years at Singapore and Sydney unforgettable: Guodong Long, Jing Jiang, Chunyang Liu, Meng Fang, Shirui Pan, Mingsong Mao, Hongshu Chen, Can Wang, Junfu Yin, Guoxin Su, Dianshuang Wu, Yi Ji, Ming Xie, Xiang Li, Yifan Li, Qian Sun, and my dearest friends Taoyu Lin and Peng Su since my college. They are the ones who have given me support during both joyful and stressful times, to whom I will always be thankful.

Finally, it is my greatest honor to thank my family: my parents, my grandparents, my uncle and auntie. They are always believing in me, keeping encouraging me, giving me indispensable suggestions, and fully supporting all my final decisions. No words could possibly express my deepest gratitude for their endless love, self-sacrifice and unwavering help. To them I dedicate this dissertation.

Abstract

There has been an explosion of data derived from the internet and other digital sources. These data are usually multi-dimensional, massive in volume, frequently incomplete, noisy, and complicated in structure. These "big data" bring new challenges to machine learning (ML), which has historically been designed for small volumes of clearly defined and structured data. In this thesis we propose new methods of "compressed learning", which explore the components and procedures in ML methods that are compressible, in order to improve their robustness, scalability, adaptivity, and performance for big data analysis. We will study novel methodologies that compress different components throughout the learning process, propose more interpretable general compressible structures for big data, and develop effective strategies to leverage these compressible structures to produce highly scalable learning algorithms. We present several new insights into popular learning problems in the context of compressed learning. The theoretical analyses are tested on real data in order to demonstrate the efficacy and efficiency of the methodologies in real-world scenarios.

In particular, we propose "manifold elastic net (MEN)" and "double shrinking (DS)" as two fast frameworks extracting low-dimensional sparse features for dimension reduction and manifold learning. These methods compress the features on both their dimension and cardinality, and significantly improve their interpretation and performance in clustering and classification tasks.

We study how to derive fewer "anchor points" for representing large datasets in their entirety by proposing "divide-and-conquer anchoring", in which the global solution is rapidly found for near-separable

non-negative matrix factorization and completion in a distributed manner. This method represents a compression of the big data itself, rather than features, and the extracted anchors define the structure of the data.

Two fast low-rank approximation methods, "bilateral random projections (BRP)" of fast computer closed-form and "greedy bilateral sketch (GreBske)", are proposed based on random projection and greedy augmenting update rules. They can be broadly applied to learning procedures that requires updates of a low-rank matrix variable and result in significant acceleration in performance.

We study how to compress noisy data for learning by decomposing it into the sum mixture of low-rank part and sparse part. "GO decomposition (GoDec)" and the "greedy bilateral (GreB)" paradigm are proposed as two efficient approaches to this problem based on randomized and greedy strategies, respectively. Modifications of these two schemes result in novel models and extremely fast algorithms for matrix completion that aim to recover a low-rank matrix from a small number of its entries. In addition, we extend the GoDec problem in order to unmix more than two incoherent structures that are more complicated and expressive than low-rank or sparse matrices. The three proposed variants are not only novel and effective algorithms for motion segmentation in computer vision, multi-label learning, and scoring-function learning in recommendation systems, but also reveal new theoretical insights into these problems.

Finally, a compressed learning method termed "compressed labeling (CL) on distilled label sets (DL)" is proposed for solving the three core problems in multi-label learning, namely high-dimensional labels, label correlation modeling, and sample imbalance for each label. By compressing the labels and the number of classifiers in multi-label learning, CL can generate an effective and efficient training algorithm from any single-label classifier.

Contents

List of Figures

List of Tables

Chapter 1

Introduction

Over the past few decades there has been a dramatic growth in both the fundamental theories and practical algorithms used in statistical machine learning [109][21] (ML). The theoretical and practical use of these algorithms has attracted considerable attention from numerous fields. ML explores the inner structure of data by building statistical models for related variables, and then either learning or averaging the models by fitting data samples. The extracted structure provides essential information about the data and brings significant improvement on traditional techniques for solving both supervised and unsupervised tasks, because the statistical model is exquisitely fit for purpose and is derived from the data itself. ML enables the computer to learn directly from the data, free from human experience or bias, and is therefore a more efficient and adaptive realization of artificial intelligence. The practical use of ML can result in far-reaching impact on the world, from data networks, communication, information technology, graphics, and signal processing to biology, medical science [115], public health, meteorology, and economics. It influences not only the way in which people discover the physical world around them and their relationship with it, but also the speed and accuracy of how they do it. ML has the potential to improve lives and reduce unnecessary costs. However, applying ML to real-world data remains challenging and has significant limitations.

In most of todays practical applications, large-volume and high-dimensional big data with missing values and unexpected noise is far more common than small, clean data that strictly satisfies the assumptions in many ML methodolo-

gies. Many ML methods benefit from the complex and detailed data structure encoded in their models. Although improvements are possible when this kind of ML method is applied to small quantities of data whose structure can be kept consistent with the model assumptions, dramatic increases in data scale are usually accompanied by violation of these underlying (and often complicated) assumptions. The performance of ML becomes unpredictable and frequently unstable. In addition, the high volume and dimensionality leads to an increase in processing time and sample complexity, and this impacts on computational burden and results in a large variability in learning results. Conventional ML methods may therefore fail when applied to big data, and novel ML techniques that allow more flexibility in the size of data and their structure need to be developed. These requirements demand that the ML methods a) adopt simple structured assumptions that are powerful and flexible enough to fit data with different structures, noise, and missing values, b) can be used for both supervised or unsupervised learning tasks, and c) keep time and sample complexity sufficiently small for use on big data.

With these requirements in mind, here we tackle the problem of compressed learning of big data for different tasks. In particular, we take advantage of compressible components within big data to express complicated data structures and develop highly scalable algorithms for specific learning tasks. The compressible components in the model simultaneously preserve variable correlations, which is extremely helpful for improving performance. This scheme is called "compressed learning" because it aims to develop big data learning methods on compressed features, labels, or learning tasks by fully exploring their correlations and structures. Although the methodology can be traced back to classic learning methods, such as dimension reduction and feature selection, the techniques presented in this thesis broaden the concept beyond feature extraction and to data with more complicated structure. In addition, we provide several novel insights on how to develop scalable and robust algorithms specifically for big data.

We first introduce low-rank and sparse structures, which are the most familiar compressible structures already used in ML research. We then provide a brief overview of the compressed learning models and algorithms proposed in this thesis.

1.1 Low-rank and Sparse Structures in Learning Problems

There has been an explosion in the quantity, although not necessarily the quality, of information from the internet and other digital sources. These data are usually characterized by high-dimensionality, huge volume, incompleteness, frequently dominant noise, and complicated structure, all of which bring intriguing new challenges to compressive acquisition, signal processing, and ML. Traditional methods are limited by the computational time needed, sample complexity, storage, and de-noising capability. These limitations have driven recent attempts to exploit the intrinsic redundancy and structure of the data. For a single signal or individual instance, the redundancy is usually exhibited by its sparsity, i.e. it is nonzero only for a small set of entries. Compressed sensing [71][41] recovers a signal from its highly compressed measurements by an undetermined linear system, and therefore leveraging this sparsity. For multiple instances, or more specifically a matrix, the redundancy is often identified by its low-rank structure, i.e. all the instances lie in a subspace spanned by a small number of bases. Low-rank structure can be analogized to sparsity due to its sparse spectrum. Alternatively, the matrix X can be written as the sum of a few rank-1 matrices such that $X = \sum_{i=1}^{r} U_i V_i$, in which U_i is a column vector and V_i is a row vector.

1.1.1 Low-rank Structure and Dimension Reduction

A primary focus of ML is to find succinct and effective representations of originally high-dimensional samples [109][143][209][207]. Linear dimensionality deduction is such a tool that projects the original samples from a high-dimensional space to a low-dimensional subspace.

Real-world data matrices are unlikely to be exactly low rank. However, approximating them to low-rank matrices provides a good trade-off between accuracy and time/space costs, especially when its singular values decay fast. The low-rank approximation [238] can replace the original matrix in least squares regression and matrix product, and also provides a low-dimensional representation which can boost both classification and clustering performance. Although

the low-rank approximation is probably optimal when constructed from singular value decomposition (SVD), the expensive time cost makes SVD prohibitive for large matrices. Therefore, many faster Monte Carlo algorithms [75][53][183] have been proposed as an alternative to SVD, which trade sufficiently high probability at the cost of acceptably small error. Typically, Monte Carlo algorithms build the low-rank approximation from randomly selected columns or rows [27], or linear projections (which are called "sketch") of certain random matrices [108]. These can be regarded as faster and more structured alternatives to principal components analysis (PCA) [119].

Some other examples of conventional linear dimensionality reduction algorithms include Fisher's linear discriminant analysis (FLDA) [87], regularized FLDA, and geometric mean-based subspace selection [208]. All of these algorithms assume samples are drawn from different Gaussians. PCA maximizes the mutual information between original high-dimensional Gaussian-distributed samples and projected low-dimensional samples. PCA, which is unsupervised, does not utilize class label information. In contrast, LDA finds a projection matrix that simultaneously maximizes the trace of the between-class scatter matrix and minimizes the trace of the within-class scatter matrix in the projected subspace. Similar to PCA, FLDA and regularized FLDA assume samples are drawn from homoscedastic Gaussians. Therefore, FLDA and regularized FLDA do not work well when Gaussians are heteroscedastic. Additionally, they always merge classes that are close together in the high-dimensional space. Although the geometric mean-based subspace selection and its harmonic mean-based extension [18] assume samples are drawn from heteroscedastic Gaussians and do not tend to merge close classes, they basically work for Gaussian distributed samples.

However, linear dimensionality reduction performs poorly when expressing data with nonlinear structure or local similar geometry. Manifold learning-based dimensionality reduction extends the low-rank concept from linear subspace to smooth manifolds, e.g. locally linear embedding (LLE) [196], ISOMAP [210], Laplacian eigenmaps (LE) [15][157], Hessian eigenmaps (HLLE) [72], Generative Topographic Mapping (GTM) [22][92] and local tangent space alignment (LTSA) [254]. LLE uses linear coefficients that reconstruct a given measurement from its neighbors to represent the local geometry and then seeks low-dimensional em-

bedding in which the coefficients are still suitable for reconstruction. ISOMAP preserves the global geodesic distances of all pairs of measurements. LE preserves proximity relationships by manipulations on an undirected weighted graph, which indicate the neighbor relationships of pairwise measurements. LTSA exploits the local tangent information as a representation of the local geometry, and this local tangent information is then aligned to provide a global coordinate. HLLE obtains the final low-dimensional representations by applying eigenanalysis to a matrix built by estimating the Hessian over the neighborhood. All these algorithms possess the out of sample problem, and thus several linearizations have been proposed, e.g. locality preserving projections (LPP) [110], neighborhood preserving embedding (NPE) [112], and orthogonal neighborhood preserving projections (ONPP). Recently, we developed a systematic framework (patch alignment [249][250] for understanding the common properties of, and intrinsic differences between, the different algorithms, including their linearizations. In particular, this framework demonstrated that i) algorithms are intrinsically different at the patch optimization stage, and ii) all algorithms share an almost identical whole alignment stage. Another unified view of popular manifold learning algorithms is the graph embedding framework [235]. Based on both frameworks, different algorithms have been developed, e.g. discriminative locality alignment [164], manifold regularization [16], and marginal Fisher's analysis [227].

Specific dimension reduction models can be designed for different learning tasks. In these methods, the data samples are compressed to a low-dimensional feature space, in which the useful information is well preserved, the assumptions of the learning model are fulfilled, and the time cost of applying learning algorithms is reduced. However, the compression is only applied to the dimensions of data, is not robust to outliers, and cannot express more complicated structures. Moreover, the training stage of these methods always requires SVD, and the associated expensive time cost significantly limits these applications for big data.

1.1.2 Sparse Structure and Sparse Learning

Sparsity has been widely exploited to compress information, obtain efficient coding of massive data, and select important features. In signal processing, com-

pressed sensing [40][71][206][166][261] has proved that a sparse signal can be exactly recovered from a small number of its random projections significantly less than the Nyquist rate. In statistics, *lasso* [211] and other ℓ_1-regularized regression models [160][161][166] have been proposed to select important variables for building a parsimonious prediction model of a response. In sparse coding [1, 142, 150], an over-complete dictionary leads to sparse representations for dense signals of the same type. The sparse representation has been proven to be effective for many learning tasks such as classification and recognition [230][273][262].

Early statistics research on sparse learning mainly focused on solving the feature selection problem, e.g. lasso [211], least angle regression (LARS) [77], elastic net [275], the smoothly clipped absolute deviation penalty (SCAD) [128], sure independence screening [84], Dantzig selector [38] and Dantzig selector with sequential optimization (Dasso) [130]. A direct method to reduce the number of involved features in learning problems is to set very small coefficients as zero. However, this strategy is problematic because small coefficients might be very important. Since each new base is a linear combination of the original ones, it is reasonable to consider each as the response of several variables, i.e. the original features. In this way, the problem of sparse learning becomes similar to variable selection and coefficient shrinkage. In linear regression, the Lp norm penalty is always combined with the loss function to reduce over-fitting. In particular, ℓ_1-norm (or lasso) possesses the desirable property of driving an acceptable number of coefficients to zero, and this leads to a sparse model between responses and variables because of its singularity in the origin [185][123]. The number of lasso-selected variables is no larger than the number of samples. Moreover, lasso randomly selects one variable from the group of variables that are highly correlated. Therefore, elastic net has been proposed to address the above problems and achieve the grouping effect by adding the ℓ_2 penalty to lasso.

In recent years, group sparsity [243] and structured sparsity [131][124] have been proposed to not only capture the sparse structure in learning problems, but also consider the variable relations when selecting features. This is effective in problems such as multi-task learning [6][159][7].

Although sparse learning has recently become extremely popular, it also has several limitations. Firstly, the algorithms that pursue a sparse solution are

always slow in speed. Secondly, sparse learning merely compresses the features or weights the variables in learning problems. Thirdly, a sparse structure is not expressive on some big data with complicated structures.

1.2 Literature Survey of Compressed Learning

In history, several names have been proposed or used for different types of compressed learning. Various broadly discussed techniques, such as dimension reduction and sparse learning mentioned before, can be classified as two categories of compressed learning, because of the compressible structures and data redundancy they explored. Most methods in these categories focus on feature extraction or selection resulting in a decreasing number of features. However, there also exists other compressed learning methods beyond these two categories. We will give a brief introduction to them below, while their details can be found in the beginning of each Chapter.

Instead of merely assuming low-rank or sparse structures, a more expressive compression of the original data with structural information is to decompose it as a mixture of different compressible structures rather than a simple one. An representative of this kind is robust PCA [37], which decompose the data matrix into the sum of a low-rank part and a sparse part. A more general assumption $X = L + S$ is applied in this case [43], i.e., the data matrix X can be decomposed as the sum of a low-rank matrix L and a sparse matrix S. L explains the components that lie in a subspace and smoothly change across different instances, while S contains the spiky anomalies that are rarely shared by different instances. This model is called "robust PCA" due to its robustness to sparse noise S when recovering principle components of L from X, and can be applied to video surveillance, graphical mode selection, image alignment, multi-label learning [259], etc. PCP [37] recovers L and S from X by minimizing sum of the trace norm of L and the ℓ_1 norm of S. It can be proved that the solution to this convex relaxation is the exact recovery if $X = L + S$ indeed exists and L and S are sufficiently incoherent [43][37]. That is, L obeys the incoherence property in (8.1) and thus is not sparse, while S has nonzero entries uniformly selected at random and thus is not low-rank. Popular optimization algorithms such as augmented Lagrangian mul-

tiplier, accelerated proximal gradient method and accelerated projected gradient method [46] have been applied. But full SVD as a costly subroutine is required to be repeatedly invoked in any of them.

Another commonly observed phenomenon on big data is the large amount of missing values, which cannot be always replaced by arbitrary numbers (for example, zeros) and thus needs to be completed according to the low-rank structure among data entries. The goal of matrix completion [39][36] is to recover the whole data matrix X from partial noisy entries or undersampled linear measurements $\mathcal{A}(X)$ ($\mathcal{A}$ is the sampling operator) by leveraging the connections between different instances. Such connections can be well captured by low-rank structure. Matrix completion has broad applications in various real problems of considerable importance, such as collaborative filtering in recommendation system [272] link prediction for social network, quantum state tomography and traffic distance completion. The matrix completion problem was firstly written as a rank minimization that is NP-hard both to solve and approximate. Thus trace norm (ℓ_1 norm of singular values) as the convex surrogate of rank has been minimized in many popular approaches [35][132]. For a matrix $X \in \mathbb{R}^{m\times n}$ of rank-r with SVD decomposition $X = B\Sigma D$, the obtained global solution is provable to exactly recover X from $\mathcal{O}(nr\log^6 n)$ uniformly sampled noisy entries if it fulfills incoherence property. Another norm receiving much attention for encouraging low-rank solution is max-norm $\|X\|_{max} = \inf\{\|U\|_{2,\infty}\|V^T\|_{2,\infty} : X = UV\}$ [204][151] ($\|\cdot\|_{2,\infty}$ is the maximum ℓ_2 row norm of a matrix), whose theoretical recovery bound is established based on Rademacher complexity of its unit ball [89]. The max-norm can be defined as the global solution to an non-convex optimization on left and right factors U and V.

Both trace norm minimization and max-norm minimization can be formulated as semidefinite programming (SDP) which has standard solvers. Various accelerated optimization methodologies [168][151] also have been applied to them for practical purpose. However, most of them rely on costly computation of full SVD in each iterate, and thus do not scale well to large-scale problems. This fact induces a revisit of rank minimization/constraint based formulations. Forward greedy selection methods including ADMiRA [152] and GECO [200] are developed for rank minimization. GECO adopts an interesting greedy strategy:

it increments the rank by 1 and adds the optimal rank-1 direction into the optimization per iterate. Incremental OptSpace [137] also has a similar scheme. We inherit a similar spirit but solve different optimization models in GreB. Error minimization with rank constraint is also considered in SVP [129]. Some of them like OptSpace and GECO model the low-rank variable as factorization USV and optimize over (U, V) pair and S. Unfortunately, truncated SVD or large matrix multiplication is still required in these approaches. In addition, the update of S is to solve a large-scale overdetermined linear system and thus time consuming in practice. Furthermore, the rank is fixed within some algorithms, so in this case the recovery accuracy strongly relies on the quality of rank estimation. Online/stochastic gradient method [12] and aggregated (divide-and-conquer) method [170] have also been considered for pursuing approximated recovery. Since GreB can be straightforwardly transferred to or invoked as a subroutine by them to take their advantages, they will not be included in later comparison.

While most other compressed learning methods are seeking for low-dimensional latent features or sparse/compressible representations, the number of labels or responses could also be huge in real problems and lead to a large output space that is difficult to be handled by current supervised learning methods. One sub-category of compressed learning contains approaches simplifying the learning process by compressing the output space. These techniques are always designed for joint prediction, multi-label learning or multi-task learning. In this thesis, we are mainly interested in multi-label learning adopting this label compression strategy. The $\{0, 1\}$ label matrix Y is sparse and thus compressible. In [122], Y is compressed via random projections $Y' = YA$, and then a new regression model $Y' = XW'$ is obtained, the original Y can be recovered via compressed sensing algorithms. This is a successful application of compressed sensing (CS) [71] to multi-label learning and inherits the theoretical merit of CS, i.e., only $\mathcal{O}(\log(k))$ models needs to be trained for data with k labels. This method reduces the model complexity caused by the large number of labels. Some label transformation methods (please refer to 10.1.2) can also be regarded as label compression by exploring their correlations and structure.

1.3 Main Contributions and Road Map

In the remainder of this thesis, we expand the compressed structures from low-rank and sparse to more complicated structures that are more common and more adaptive for big data purposes. For different learning tasks, we will develop compressed learning models that leverage these structures to reduce the time cost and improve the learning performance, especially in noisy and large-scale cases that cannot be fully handled by previous methods. The concept of "compressed learning" plays an essential role in these models and algorithms, and provides new insights in to various significant learning problems. We show the relationships between the proposed approaches from different perspectives in Figure 1.1. A brief introduction to these techniques are given below in the sequence of Chapters.

In particular, Chapter 2 and Chapter 3 aim to develop fast algorithms that extract the low-dimensional sparse features of the data for a broad set of learning tasks. This problem not only compresses data on its dimensions, but also on its cardinality. Chapter 2 introduces the "manifold elastic net (MEN)" [269, 267], which learns low-dimensional feature space with a sparse projection matrix or sparse data representations by formulating the problem as a sparse regression with elastic net regularization. It is a unified framework that can be applied to most manifold learning and dimensionality reduction models in order to obtain sparse solutions. Chapter 3 introduces "double shrinking (DS)" [264], in which similar learning tasks are solved by establishing an ℓ_1-norm regularized eigenvalue maximization/minimization model and by developing a path-following optimization algorithm. The DS model is more challenging to optimize, but is more generalizable than the regression model adopted by MEN. Both methods improve the interpretation of extracted features and the learning performance in different learning tasks, such as clustering and recognition, and when applied to different real data including human faces, object images, and genomic data.

Chapter 4 studies how to compress big data into fewer "anchor points" that are able to represent the whole dataset in learning problems. This problem is different from dimensionality reduction that compresses features, since it aims to compress entire data samples. In particular, we propose a "divide-and-conquer anchoring (DCA)" [257] method to rapidly find anchor points as the global optimal solution

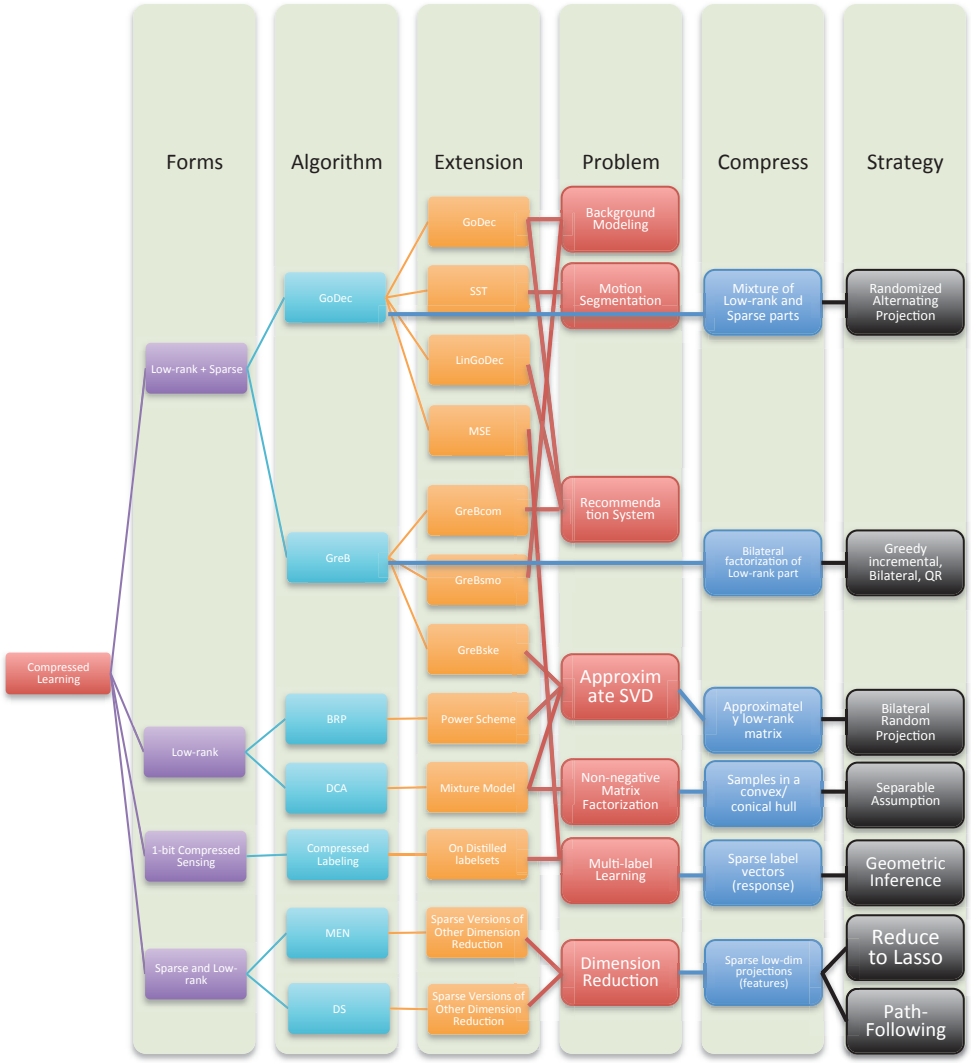

Figure 1.1: Relationships between the proposed approaches in this thesis.

for near-separable, non-negative matrix factorization (NMF). This method is able to process extremely large-scale data in a distributed manner and in cases that are corrupted with noise or contain missing values. Empirical studies highlight its robust properties across different types of data.

Chapter 5 provides two low-rank matrix approximation strategies to deal with large numbers of data samples that can be accurately approximated by a small number of basis vectors. The first strategy, "bilateral random projections (BRP)"

[263], is developed based on spectrum analysis of random matrix theory. BRP offers a closed form of low-rank approximation built from left and right random projections of a data matrix. The number of required random projections can be far less than the size of the data matrix, and as a result the time cost is very small. The other strategy, "greedy bilateral sketch" [258], formulates the problem as an optimization of the left and right factors of the low-rank approximation, and updates the two factors in a mutually adaptive and greedy augmenting manner, that is the size of factor matrices are dynamically increased by adding the best rank-one direction into them. These two strategies are able to handle low-rank structure learning problems on big data, and are supported by solid theoretical guarantees.

Chapter 6 and Chapter 7 study learning noisy data by compressing it to the linear mixture of the two compressible structures, i.e. represent the data matrix X as the sum of a sparse part S, a low-rank part L, and noise G. This mixture structure is more expressive and generalizable for big data, in which not all the samples can be limited to a single simple structure. Chapter 6 proposes "GO decomposition (GoDec)" [265], in which the two incoherence structures are separated from data by fast randomized update rules, while the "greedy bilateral (GreB)" scheme [258] in Chapter 7 provides a greedy and factor model-based method for solving the unmixing problem, resulting in even greater acceleration. The low-rank structure always represents the shared information between samples, while the sparse structure usually captures the individual or anormal effect in specific samples, or outliers. Hence, the extracted compressible structures can be applied to a vast range of classification or clustering-based learning tasks in computer vision and text mining, and is able to achieve robust performance. Convergence analysis of these two approaches is presented.

Chapter 8 focuses on matrix completion, which predicts the missing values of a data matrix from partial entries by exploring their relationships as defined by low-rank structure. This problem aims to learn unknown data from compressed information, and has tremendous utility in recommendation systems and signal processing. However, the slow speed of existing algorithms limits its potential. We extend GoDec and GreB (Chapter 6 and 7) [265, 258] to produce randomized and greedy algorithms for matrix completion of large-scale matrices. Experimen-

tal results on real recommendation data of movies and jokes shows significant improvements in both speed and prediction accuracy using these two algorithms.

In Chapter 9 we extend the low-rank and sparse structures of GoDec and GreB to more expressive and general structures, and propose fast unmixing algorithms for these more complicated structures. In particular, we propose three non-trivial variants: 1) in "shifted subspace tracking (SST)" [260] for motion segmentation, we further decompose the sparse S (moving objects) as the sum of multiple row-sparse matrices, each of which is a low-rank matrix after a specific geometric transformation sequence and defines a motion shared by multiple objects; 2) in "multi-label subspace ensemble (MSE)" [259, 266] for multi-label learning, we further decompose the low-rank L into subcomponents with separable subspaces, each corresponding to the mapping of a single label in feature space. The prediction can then be effectively conducted by group lasso on the subspace ensemble; 3) in "linear functional GoDec (LinGoDec)" for estimating the scoring functions of each user in a recommendation system, we further decompose the low-rank L as WZ^T, where the rows of W are the linear scoring functions and the rows of Z are the items represented by available features. These variants expand the concepts of incoherent structures defined by low-rank and sparse matrices to multiple (≥ 2) components and more complicated structures, and allow nonlinear correlations among samples. Therefore, compared to robust PCA and GoDec, which limit the clean data matrix to be the sum of a low-rank part and a sparse part, these variants are capable of modeling much broader classes and more general types of data in real applications, including motion in videos, multi-label data, and recommendation systems, where the mixing structures are more complicated and more difficult to separate.

Chapter 10 proposes a novel multi-label learning scheme, "compressed labeling (CL) on distilled label sets (DL)" [268], that can produce effective and efficient multi-label learning from any single-label classifier. This approach simultaneously solves three core problems in multi-label learning, namely high-dimensional labels, label correlation modeling, and sample imbalance for each label. Rather than compressing the features or samples in learning, CL compresses the responses and the number of classifiers in supervised learning. In particular, CL compresses the high-dimensional sparse label vectors to a compact 0-1 balanced label vector

of low dimensions, learns far fewer single-label classifiers independently, and uses a hypothesis-testing method to determine the appearance of frequent label sets (i.e. distilled label sets) from the compressed label vector in prediction. CL significantly improves the balance of the training samples and reduces the dependence between different labels. Moreover, it accelerates the learning process by training less binary classifiers for compressed labels, and makes use of label dependence via DL-based tests. Theoretically, we prove the recovery bound of CL, which verifies the effectiveness of CL for improving label compression and multi-label classification performance by the label correlations preserved in DLs. The compressed learning scheme in CL substantially reduces both the time and sample complexity of multi-label learning to less than those for independent single-label learning, and simultaneously fully explores the label correlations to improve the prediction accuracy.

In summary, this thesis studies compressed learning from several different perspectives. First, we study how to compress feature dimension, representation cardinality, representative samples, and responses of supervised learning for different learning tasks, and in doing so we broaden the methodology of compressed learning. Second, we study different forms of compressible structures, including sparse and low-dimensional features, the sum mixture of low-rank and sparse structures, low-rank structure under nonlinear transformation, low-rank structure defined by multiple functions, low-rank structure that can be decomposed into multiple distinguishable subspaces, and the sum mixture of these variants with sparse structure. These expressive and compressible structures provide powerful tools for building accurate and adaptive models for complicated data. Third, the proposed compressed learning models take both noise and missing values into account in their development, and are therefore robust on big data that are not as clean as many ML methodologies assume. In addition, the divide-and-conquer, randomized, and greedy strategies adopted in these methods lead to highly scalable algorithms that can easily handle large volumes of data that cannot be processed by conventional ML methods. Most of the proposed methods can be modified to fast online learning [255, 117] after minor changes. Finally, all of the proposed methods provide practical solutions to real-life problems, and we demonstrate their efficacy and efficiency on real data, underpinned by solid the-

ory from rigorous analysis. Some contents of this thesis come from our published papers [257]-[270].

Chapter 2

Manifold Elastic Net: A Unified Framework for Sparse Dimension Reduction

It is difficult to find the optimal sparse solution of a manifold learning based dimensionality reduction algorithm. The lasso or the elastic net penalized manifold learning based dimensionality reduction is not directly a lasso penalized least square problem and thus the least angle regression (LARS) [77], one of the most popular algorithms in sparse learning, cannot be applied. Therefore, most current approaches take indirect ways or have strict settings, which can be inconvenient for applications. In this chapter, we proposed the manifold elastic net or MEN for short. MEN incorporates the merits of both the manifold learning based dimensionality reduction and the sparse learning based dimensionality reduction. By using a series of equivalent transformations, we show MEN is equivalent to the lasso penalized least square problem and thus LARS is adopted to obtain the optimal sparse solution of MEN. In particular, MEN has the following advantages for subsequent classification: 1) the local geometry of samples is well preserved for low dimensional data representation, 2) both the margin maximization and the classification error minimization are considered for sparse projection calculation, 3) the projection matrix of MEN improves the parsimony in computation, 4) the elastic net penalty reduces the over-fitting problem, and 5) the projection

matrix of MEN can be interpreted psychologically and physiologically. Experimental evidence on face recognition [116] over various popular datasets suggests that MEN is superior to top level dimensionality reduction algorithms.

2.1 Introduction

One of the primary focuses in data mining and machine learning is finding a succinct and effective representation for original high dimensional samples [109][143][68][149][156][209][207]. Linear dimensionality deduction is such a tool that projects the original samples from a high dimensional space to a low dimensional subspace. Meanwhile some particular information, e.g., manifold structure and discriminative information, of the original high dimensional samples will be well preserved while noises will be removed in the selected subspace.

In recent years, sparse learning becomes popular, because:

1. sparsity can make the data more succinct and simpler, so the calculation of the low dimensional representation and the subsequent processing, e.g., classification and regression, becomes more efficient. Parsimony is especially an important factor when the dimension of the original samples is very high and the number of samples is very large;

2. sparsity can control the weights of original variables and decrease the variance brought by possible over-fitting with the least increment of the bias. Therefore, the learn model can generalize better; and

3. sparsity provides a good interpretation of a model, thus reveals an explicit relationship between the objective of the model and the given variables. This is important for understanding practical problems, especially when the number of variables is larger than that of the samples.

However, it is not easy to find the optimal solution of a sparse learning model. In the original lasso, the residue sum of squares is minimized subject to the sum of the absolute value of the coefficients being less than a constant. The quadratic programming is sequentially utilized to get the solution and thus the time cost

is not acceptable for practical applications. Recently, the least angle regression (LARS) [77] is proposed to seek a close form solution to the path of coefficients in each step without using the quadratic programming, so it is more efficient and less greedy than the original optimization algorithm used in lasso.

Hitherto, most of sparse dimensionality reduction algorithms are designed for linear regression and only a few can be applied for subsequent classification, e.g., sparse principal component analysis (SPCA) [273], Nonnegative sparse principal component analysis [245], sparse linear discriminant analysis (SLDA), sparse projections over graph (SPOG) [33][34] and SPCA using semi-definite programming [60]. Both SPCA and SPCA using semi-definite programming do not consider the sample label information and thus some discriminative information will be removed after dimensionality reduction. SLDA can work well for binary class classification but it cannot be applied for multi-class classification. SPOG utilizes a particular manifold learning based dimensionality reduction algorithm, e.g., locality preserving projections (LPP), to obtain the dense projection matrix and then applies lasso to regress the corresponding sparse projection matrix. Absolutely the problem is indirectly formulated to obtain the sparse projection matrix. A direct formulation should be imposing the lasso penalty over a loss function (i.e., a criterion) of a dimensionality reduction algorithm. However, it is difficult to use LARS to obtain its optimal solution because the objective function is not a direct regression problem. Therefore, researchers currently take indirect routs to obtain sparse projection matrices.

2.1.1 The proposed approach

In this chapter, we propose the manifold elastic net (MEN), which obtains a sparse projection matrix for subsequent classification. MEN directly imposes the elastic net penalty (i.e., the combination of the lasso penalty and the ℓ_2-norm penalty) over the loss (i.e., the criterion) of a discriminative manifold learning based dimensionality reduction algorithm. By using a series of complex linear algebra equivalent transformations, the objective function of MEN can be rewritten as a lasso penalized least square problem and thus LARS can be applied to obtain the optimal sparse solution of MEN.

In detail, we first apply the part optimization of the patch alignment framework to encode the local geometry of a set of training samples. In the second step, the whole alignment of the patch alignment framework is applied to calculate the unified coordinate system for local patches obtained in the first step. For low dimensional data representation, the linearization or the linear approximation is adopted in MEN. Although we can impose some discriminative information preservation criterion (e.g., margin maximization) over the part optimization stage, it is not directly relevant to the classification error minimization. Therefore, we put a new item that minimizes the classification error in the third step. To obtain a sparse projection matrix with the grouping effect, in the fourth step, the elastic net penalty is adopted in MEN. So far, the objective function of MEN is fully constructed.

With the well defined MEN, we then apply LARS to obtain the optimal solution of MEN. We transform MEN into a form in which the correlation of basis can be written as the correlation of coefficients. Active set is built according to LARS. In each step, no more than one element of the basis is added to the active set according to its correlation. All elements in the active set are changed in each step with special direction and distance in the space of coefficients. The direction and distance of a path in each step have closed form solution according to the extended simplex. The sparsity of the projection matrix is controlled by the cardinality of the active set. Because the LARS for MEN generates bases in an independent way, the same procedure is conducted multiple times to obtain a set of bases. Under this procedure, these bases are orthogonal. Thorough experiments on face recognition [199] task based on popular face datasets show the effectiveness of the proposed MEN by comparing against the top level dimensionality reduction algorithms.

The rest of the paper is organized as follows. Section 2 presents the proposed manifold elastic net (MEN) including the objective function of MEN and the LARS optimization for MEN. Section 3 shows the effectiveness of MEN for face recognition over different face datasets. Section 4 concludes.

2.2 Manifold Elastic Net

Consider in the discriminative dimensionality reduction problem with training samples and corresponding class labels. Let $X = [x_1, x_2, \cdots, x_n]^T \in \mathbb{R}^{n \times p}$ be a given training set in a high dimensional space $\mathbb{R}^{n \times p}$ and $C = [c_1, c_2, \cdots, c_n]^T \in \mathbb{R}^n$ be the corresponding class label vector. The objective here is to find a projection matrix $W = [w_1, w_2, \cdots, w_d]^T \in \mathbb{R}^{p \times d}$ that projects samples $x^T \in \mathbb{R}^p$ in the high dimensional space onto a low dimensional subspace, i.e., $z^T = x^T W$, such that samples from different classes can be well separate, i.e., the classification error can be extremely minimized.

Manifold learning based dimensionality reduction aims to find the corresponding low dimensional representation z in a low dimensional Euclidean space of x to preserve (actually approximate) the data intrinsic structure. Popular manifold learning based dimensionality reduction algorithms, however, have the following two problems: 1) the classification error is not directly and explicitly considered, although some algorithms compound discriminative information preservation criteria, e.g., margin maximization; and 2) the obtained low dimensional representation linear combines of all variables in the high dimensional space, so it is difficult to clear interpret and efficiently represent data.

Sparse learning provides sparse data representation via variable selection, and has the following advantages: 1) the sparsity improves the parsimony in computation, i.e., the computational cost can be significantly reduce; 2) the penalties and the constraints introduced in a learning model discourage the possible overfitting of the model; and 3) the learned model can be well interpreted. However, existing sparse learning algorithms are designed for linear regression problems and the data intrinsic structure is usually ignored.

To achieve the merits of manifold learning based dimensionality reduction and the advantages of sparse learning, in this chapter, we propose the manifold elastic net (MEN), which is a general framework to obtain the sparse solution of the manifold learning based discriminative dimensionality reduction. There are few research results on combining sparse learning and discriminative dimensionality reduction because the projection matrix of a lasso penalized model cannot be obtained directly by using the least angle regression (LARS).

MEN is not a direct combination of the manifold learning based dimensionality reduction and the sparse learning. It however finds the optimal sparse solution of every manifold learning based discriminative dimensionality reduction algorithm via the patch alignment framework and a new classification error minimization based criterion. In particular, MEN encodes the local geometry of a set of samples and finds an aligned coordinate system for data representation under the patch alignment framework; MEN utilizes the classification error minimization criterion to directly link the classification error with the selected subspace; and MEN incorporates the elastic net regularization to sparsify the projection matrix.

2.2.1 Part optimization

Different manifold learning algorithms encode different types of local geometry of samples, e.g., locally linear embedding (LLE) applies linear coefficients to reconstruct a sample by its neighbors. The patch alignment framework has well demonstrated that different algorithms have different optimization criteria to encode different local geometry over patches.

In MEN, the same as the part optimization in the patch alignment framework, each patch is constructed by a particular sample x_i and its k related ones $x_{i_1}, x_{i_1}, \cdots, x_{i_k}$. The patch is denoted by $X_i = \left[x_i^T, x_{i_1}^T, x_{i_2}^T, \cdots, x_{i_k}^T\right]^T \in \mathbb{R}^{(k+1)\times p}$. MEN finds a linear mapping f_i that projects the patch $X_i \in \mathbb{R}^p$ to a low dimensional subspace $\mathbb{R}^d$, i.e., $f_i : X_i \longmapsto Z_i$, where $Z_i = \left[z_i^T, z_{i_1}^T, z_{i_2}^T, \cdots, z_{i_k}^T\right]^T \in \mathbb{R}^{(k+1)\times d}$. The part optimization maximizes the similarity of the local geometry represented by X_i and that described by Z_i:

$$\arg\min_{Z_i} \operatorname{tr}\left(Z_i^T L_i Z_i\right), \tag{2.1}$$

where $L_i \in \mathbb{R}^{(k+1)\times(k+1)}$ encodes the local geometry of the patch X_i and it is different over different dimensionality reduction algorithms.

For a given sample x_i, its k related ones are divided into two groups: the k_1 ones in the same class with x_i and the k_2 ones from different classes with x_i. These two groups are selected independently and denoted by $\{x_{i^1}, x_{i^2}, \cdots, x_{i^{k_1}}\}$

and $\left\{x_{i_1}, x_{i_2}, \cdots, x_{i_{k_1}}\right\}$ respectively. Therefore, the patch for x_i is defined by

$$X_i = \left[x_i^T, x_{i^1}^T, x_{i^2}^T, \cdots, x_{i^{k_1}}^T, x_{i_1}^T, x_{i_2}^T, \cdots, x_{i_{k_1}}^T\right]^T \in \mathbb{R}^{(k_1+k_2+1)\times p}.$$

The corresponding the low dimensional representation is

$$Z_i = \left[z_i^T, z_{i^1}^T, z_{i^2}^T, \cdots, z_{i^{k_1}}^T, z_{i_1}^T, z_{i_2}^T, \cdots, z_{i_{k_1}}^T\right]^T \in \mathbb{R}^{(k_1+k_2+1)\times d}.$$

Let $F_i = \left\{i, i^1, i^2, \cdots, i^{k_1}, i_1, i_2, \cdots, i_{k_2}\right\}$ to be the index set. In the low dimensional subspace, we expect that the distances between the given sample and the group of related samples from different classes are as large as possible, while the distances between the sample and the group of related samples in the same class are as small as possible. Therefore the part optimization is:

$$\arg\min_{Z_i} \sum_{j=1}^{k_1} \|z_i - z_{i^j}\|_2^2 - \kappa \sum_{p=1}^{k_2} \|z_i - z_{i_p}\|_2^2, \tag{2.2}$$

where κ is a trade-off parameter to control the impacts of the two parts. Define the coefficient vector:

$$\omega_i = \left[\overbrace{1, 1, ..., 1}^{k1}, \overbrace{-\kappa, -\kappa, ..., -\kappa}^{k2}\right]^T, \tag{2.3}$$

then we can obtain the part optimization matrix,

$$L_i = \begin{bmatrix} \sum_{j=1}^{k_1+k_2} (\omega_i)_j & -\omega_i^T \\ -\omega_i & \operatorname{diag}(\omega_i) \end{bmatrix}. \tag{2.4}$$

2.2.2 Whole alignment

Each patch X_i for $1 \le i \le n$ has a corresponding low dimensional representation Z_i. To unify all low dimensional patches $Z_i = \left[z_i^T, z_{i^1}^T, z_{i^2}^T, \cdots, z_{i^{k_1}}^T, z_{i_1}^T, z_{i_2}^T, \cdots, z_{i_{k_1}}^T\right]^T$ for $1 \le i \le n$ together into a consistent coordinate system, according to the patch alignment framework, we assume that the coordinate of Z_i is selected from the global coordinate $Z = \left[z_1^T, z_2^T, \cdots, z_n^T\right]^T \in \mathbb{R}^{n\times d}$ by a using sample selection

matrix $S_i \in \mathbb{R}^{(k_1+k_2+1)\times n}$:

$$Z_i = ZS_i, \tag{2.5}$$

where the selection matrix S_i is defined by

$$(S_i)_{pq} \begin{cases} 1, & \text{if } q = F_i\{p\}; \\ 0, & \text{else.} \end{cases} . \tag{2.6}$$

According to (2.5), the part optimization defined in (2.1) can be rewritten as:

$$\arg\min_Z \operatorname{tr}\left(Z^T S_i^T L_i S_i Z\right). \tag{2.7}$$

After summing over all part optimizations together, the whole alignment is given by:

$$\begin{aligned} &\arg\min_Z \sum_{i=1}^{n} \operatorname{tr}\left(Z^T S_i^T L_i S_i Z\right) \\ =&\arg\min_Z \operatorname{tr}\left(Z^T \sum_{i=1}^{n}\left(S_i^T L_i S_i\right) Z\right) \\ =&\arg\min_Z \operatorname{tr}\left(Z^T L Z\right), \end{aligned} \tag{2.8}$$

where L is the alignment matrix. It is obtained by an iterative procedure:

$$L\left(F_i, F_i\right) \leftarrow L\left(F_i, F_i\right) + L_i. \tag{2.9}$$

It is worth emphasizing that the mapping $f : X \mapsto Z$ from the high dimensional space to the low dimensional subspace can be nonlinear and implicit. However, the linear approximation $Z = XW$ is adopted, i.e., we expect the difference between Z and XW is minimized. In particular, $W = [w_1, w_2, \cdots, w_d] \in \mathbb{R}^{p\times d}$. Therefore, the objective function is:

$$\arg\min_{Z,W} \operatorname{tr}\left(Z^T L Z\right) + \beta\|Z - XW\|_2^2. \tag{2.10}$$

2.2.3 Classification error minimization

In MEN, although the discriminative information for classification is considered duly in (2.10), the classification error is not directly modeled. To further enhance the performance of MEN for classification problems, it is necessary to provide an explicit way to represent the classification error minimization in the objective function. The least square error minimization is usually adopted in binary classification,

$$\arg\min_{W} \|Y - XW\|_2^2. \tag{2.11}$$

However, it is very challenging to apply (2.11) to multi-class classification. This is mainly because the class label vector C cannot be directly utilized as the output (response) Y.

Recently, the least squares linear discriminant analysis [239][205] or LS-LDA for short is proposed and presents the equivalence relationship between the least square formulation and the conventional linear discriminant analysis (LDA) for multi-class classification under a mild condition. However, the dimension of the indicator matrix is the number of classes c. Therefore, LS-LDA can only reduce the original data to a $c-1$ dimensional subspace. It is pretty fine when samples are drawn from homoscedastic Gaussians because the Bayes optimal is achieved iff the dimension of the subspace is $c-1$. However, for practical applications, samples are usually not sampled from homoscedastic Gaussians and a dozen of experimental evidences show that we usually achieve the best classification performance in a subspace lower than $c-1$ when c is large.

In this chapter, we propose a flexible method to design the indicator matrix Y and the dimension of the selected subspace is allowed to be any number between 1 and $c-1$. In comparing with LS-LDA, the proposed indicator design method is more flexible and powerful to gain a lower dimensional representation and higher recognition rate. Therefore, the new method meets most demands for practical applications, e.g., face recognition.

The nearest-neighbor (NN) rule is commonly applied in classification problmes. In NN, it would be perfect when samples in the same class are projected onto the same point after dimensionality reduction, and this point is the low

dimensional representation of the corresponding class center. Meanwhile the variance of these projected class centers is expected to be maximized. As a consequence, the low dimensional projection of class centers can be conveniently obtained by the weighted principal component analysis (PCA).

In detail, suppose the given n samples belong to c classes, and there are c_i samples in the i^{th} class. The i^{th} class center is $o_i = (1/c_i)\sum_{j=1}^{m_i} x_j$, wherein x_j is the j^{th} sample in the i^{th} class and is a row vector in $\mathbb{R}^p$. The proportion of the i^{th} class is $p_i = c_i/n$. Therefore, the weighted covariance matrix of class centers is given by:

$$V = \sum_{i=1}^{m} p_i o_i^T o_i. \tag{2.12}$$

Suppose we expect to find a d dimensional subspace. The d eigenvectors associated with the largest d eigenvalues $\eta = [\eta_1, \eta_2, \cdots, \eta_d]$ of V are selected to calculate the low dimensional representation of the class center o_i according to

$$\hat{o}_i = o_i \eta. \tag{2.13}$$

Therefore, the indicator matrix $Y = [y_1, y_2, \cdots, y_n]^T$ is given by $y_j = \hat{o}_i$. On combining (2.10) and (2.11), we have

$$\arg\min_{Z,W} \|Y - XW\|_2^2 + \alpha \mathrm{tr}\left(Z^T L Z\right) + \beta \|Z - XW\|_2^2, \tag{2.14}$$

where α and β are trade-off parameters to control the impacts of different parts.

2.2.4 Elastic net penalty

In MEN, we expect to obtain a sparse projection matrix for explicit data representation and effective interpretation, i.e., control the number of nonzero elements in each column of the projection matrix. This nonzero number of the entries of the projection matrix can be characterized by the ℓ_0-norm of the projection matrix. We can impose it over the objective function defined in (2.14) as a penalty. However, it turns to be an NP-hard problem and thus it is always impossible to be solved in a polynomial time, because the penalty is nonconvex [166]. Therefore,

the ℓ_1-norm of the projection matrix, i.e., lasso, is usually adopted as a relaxation of the ℓ_0 penalty. Although lasso is convex, it is difficult to find the solution of the lasso regularized model. This is because the lasso term is not differentiable. Least angle regression or LARS for short has been proposed to greedily search the optimal solution of the lasso penalized linear regression problem. LARS continuously shrinks the particular coefficients (entries of the projection matrix W) towards zeros, while simultaneously preserves high prediction accuracy.

However, the lasso penalty has the following two disadvantages: 1) the number of selected variables is limited by the number of observations and 2) the lasso penalized model can only select one variable from a group of correlated ones and does not care which one is selected. By imposing an ℓ_2-norm of the projection matrix on the lasso penalized problem, similar to the elastic net, we can overcome the aforementioned two disadvantages and retain the favorable properties of the lasso penalty. In detail, the ℓ_2-norm of the projection matrix is helpful to increase the dimension (and the rank) of the combination of the data matrix and the response. In addition, the combination of the ℓ_1 and ℓ_2 of the projection matrix is convex with respect to the projection matrix and thus the obtained projection matrix has the grouping effect property.

Therefore, to obtain a sparse projection matrix W with the grouping effect, both ℓ_1-norm and ℓ_2-norm of the projection matrix are added as penalties to the objective function defined in (2.14) and we obtain the full definition of MEN:

$$\arg\min_{Z,W} \|Y - XW\|_2^2 + \alpha \mathrm{tr}\left(Z^T L Z\right) + \beta\|Z - XW\|_2^2 + \lambda_1\|W\|_1 + \lambda_2\|W\|_2^2. \tag{2.15}$$

Although there are four parameters, i.e., $\alpha, \beta, \lambda_1, \lambda_2$, needed to be determined in the above model, the parameter tunning can be done in an efficient way by considering the roles of each term in the objective function (and its weight). The first three terms are the objective of discriminative manifold learning, while the last two terms are elastic net regularization encouraging group sparsity. Therefore, we can jointly tune α and β by temporarily ignoring the elastic net regularization, where large α stretches the low-dimensional points Z close to the nonlinear manifold defined by L but might lead to worse linear approximation, large β can

result in better linear approximation but worse reflection of the favored manifold structure. Large α and β put more weight onto manifold learning than the discriminative information. After determining α and β, we can tune λ_1 and λ_2 in the same way as elastic net, because current problem equals to linear square regression with elastic net regularization.

2.2.5 LARS for MEN

It has been demonstrated that LARS is effective and efficient to find the optimal solution of the lasso or the elastic net (the combination of ℓ_1 and ℓ_2) penalized multiple linear regression. Therefore, it can be directly applied to penalized least squares only. However, the proposed MEN defined in (2.15), at the first glance, is not a penalized least square.

In this Section, we detail utilizing LARS to obtain the optimal solution of MEN. Although LARS is designed to solve the penalized multiple linear regression where the coefficients are a vector rather than a matrix, the column vectors of the projection matrix W in MEN are independent bases. Therefore, we can calculate them one by one. In the following analysis, we consider a particular column of W, i.e., w_i, and the corresponding vector y_i in the indicator matrix Y. To simplify the notations below, we keep using W and Y instead of w_i and y_i.

Because the low dimensional representation Z and the projection matrix W are independent, we can eliminate Z in the objective function. In detail, Z is obtained by setting the differentiate of the objective function F with respect to Z as 0, i.e.,

$$\frac{\partial F}{\partial Z} = \alpha \left(L + L^T\right) Z + 2\beta \left(Z - XW\right) = 0. \tag{2.16}$$

Therefore, we have

$$Z = \beta \left(\alpha L + \beta I\right)^{-1} XW. \tag{2.17}$$

According to (2.17), we can eliminate Z in the objective function defined in (2.15),

and thus we have:

$$\arg\min_{Z,W} W^T X^T A X W - 2W^T X^T Y + \lambda_1 \|W\|_1 + \lambda_2 \|W\|_2^2. \tag{2.18}$$

where this A is an asymmetric matrix computed from L:

$$\begin{aligned} A =& \alpha \left(\beta \left(\alpha L + \beta I\right)^{-1}\right)^T L \left(\beta \left(\alpha L + \beta I\right)^{-1}\right) + \\ & \beta \left(\beta \left(\alpha L + \beta I\right)^{-1} - I\right)^T \left(\beta \left(\alpha L + \beta I\right)^{-1} - I\right) + I. \end{aligned} \tag{2.19}$$

To apply LARS to obtain the optimal solution of (2.18), we expect the first item in it to be a quadratic form. Because $2X^T A X = X^T \left(A + A^T\right) X$ and the eigenvalue decomposition of $\left(A + A^T\right)/2$ can be written as UDU^T, the objective function defined in (2.18) without the elastic net penalty can be rewritten as:

$$\begin{aligned} & W^T X^T A X W - 2W^T X^T Y \\ =& W^T X^T \left(D^{1/2} U^T\right)^T \left(D^{1/2} U^T\right) X W - 2W^T X^T \left(D^{1/2} U^T\right)^T \left(\left(D^{1/2} U^T\right)^T\right)^{-1} Y \\ =& \left\| \left(\left(D^{1/2} U^T\right)^T\right)^{-1} Y - \left(D^{1/2} U^T\right) X W \right\|_2^2. \end{aligned} \tag{2.20}$$

The constant item can be ignored in optimization without loss of generality. We further set

$$X^* = (1 + \lambda_2)^{-1/2} \begin{bmatrix} \left(D^{1/2} U^T\right) X \\ \sqrt{\lambda_2} I^{p \times p} \end{bmatrix} \in \mathbb{R}^{(n+p) \times p}, \tag{2.21}$$

$$Y^* = \begin{bmatrix} \left(\left(D^{1/2} U^T\right)^T\right)^{-1} Y \\ \mathbf{0}^{p \times 1} \end{bmatrix} \in \mathbb{R}^{(n+p) \times 1} \tag{2.22}$$

in (2.18), and then we get

$$\arg\min_{W^*} \|Y^* - X^* W^*\|_2^2 + \lambda \|W^*\|_1, \tag{2.23}$$

where $\lambda = \lambda_1 / (1 + \lambda_2)$ and $W^* = \sqrt{1 + \lambda_2} W$.

According to (2.23), the LARS algorithm can be applied to obtain the optimal solution of the proposed MEN. LARS provides an efficient algorithm to solve the lasso penalized multiple linear regression.

Below we sketch LARS for the transformed MEN defined in (2.23) and provide novel viewpoints to LARS, which are helpful to better understand the proposed MEN.

We begin with a coefficient vector W^* (a column in the projection matrix with i^{th} entry $(W^*)_i$ with all zero entries. A variable (a column vector in X, i.e., a particular feature) in $\mathbb{R}^n$ is most correlated with the objective function is added to the active set A. Then the corresponding coefficient in W^* increases as large as possible until a second variable (another column vector in X, i.e., another feature) in $\mathbb{R}^n$ has the same correlation as the first variable. Instead of continuously increasing the coefficient vector in the direction of the first variable, LARS proceeds on a direction equiangular over all variables in the active set A until a new variable earns its way into A. To make the coefficient vector W^* becomes K-sparse (at most K nonzero entries), we conduct the above procedure for K loops. The optimization path direction and the corresponding path length (step size) in LARS are determined by the correlations, which are the negative gradient of the objective function defined in (2.23) without the lasso penalty, i.e.,

$$C = -\frac{\partial F}{\partial W^*} = 2\left(X^*\right)^T\left(Y^* - X^*W^*\right) = \left[c_1, c_2, \cdots, c_p\right]^T. \tag{2.24}$$

The constant 2 can be simply ignored without loss of generality in the following analysis.

The larger the correlation c_i is, the more important the corresponding variable will be, and thus the larger the corresponding coefficient $(W^*)_i$ in W^* will be. In sparse learning, important variables are added to the active set A sequentially according to their corresponding correlations defined in (2.24), and then the direction and distance of coefficient vector of all the important variables are determined.

Let A be the active set of "most correlated" variables whose coefficients are nonzero, while the other variables form an inactive set I. Thus the sparsity is determined by the cardinality of A. The correlations of variables in A are always

identical to each other in A and larger than the correlations of variables in I. Those correlations of variables in I are usually different to each other. Initially, all the variables are in inactive set I and thus the corresponding coefficients are all zero.

To make W^* K-sparse, we need to conduct the following three steps for K loops. In the first step, the variable in the inactive set I with the largest correlation is added to the active set A, i.e.,

$$\hat{C} = \max_j \{|\hat{c}_j|\} \quad \text{and} \quad A = \left\{ j : |\hat{c}_j| = \hat{C} \right\}, \tag{2.25}$$

where $\hat{c}_j$ is the current correlation of the j^{th} variable.

In the second step, the direction of the coefficient vector W^* is calculated. To make the optimization more global and less greedy, the correlations of the active variables are required to decrease equally in preferred direction. In the k^{th} loop, if the direction vector is ω, then the current correlation is given by

$$\begin{aligned} C_k =& \left(X_A^*\right)^T \left(Y^* - X^* W_k^*\right) \\ =& \left(X_A^*\right)^T \left(Y^* - X^* \left(W_{k-1}^* + \rho\omega\right)\right) \\ =& C_{k-1} + \rho \left(X_A^*\right)^T X_A^* \omega_A, \end{aligned} \tag{2.26}$$

where X_A^* contains all variables in A and each its column is sampled from X^*, C_{k-1} is the correlation in the $(k-1)^{th}$ loop, ρ is a constant that is irrelevant to the direction computation, ω_A stores directions associated with variables in A, and the change of the correlation at this step is $\left(X_A^*\right)^T X_A^* \omega_A$. The sign of ω_A, i.e., s, is identical to that of C_{k-1}, so we can calculate the magnitude of ω_A directly and then assign its sign as s. This $X_A^* \omega_A$ is an extended simplex with vertices defined by active variables. We project the i^{th} column of X^*, i.e., $(X^*)_i$, onto $X_A^* \omega_A$ and thus we get $(X^*)_i^T X_A^* \omega_A$. Because the correlations of the active variables are required to decrease equally in preferred direction, i.e., $(X^*)_i^T X_A^* \omega_A$ equals to each other over the index i, the only possible solution of $X_A^* \omega_A$ is the normal vector through the origin in the simplex space. Therefore, we have

$$\omega_A = s \cdot \left(X_A^{*T} X_A^*\right)^{-1} \mathbf{1}_A = s \cdot G_A^{-1} \mathbf{1}_A, \tag{2.27}$$

where $G_A = X_A^{*T} X_A^*$ is the Gram matrix of X_A^*. In LARS, ω_A is obtained by minimizing the squared distance between the point $X_A^* \omega_A$ on the simplex and the origin, subject to $\|\omega_A\|_1 = 1$.

To normalize the change of the correlation $X_A^{*T} X_A^* \omega_A$ to a unit vector u_A, we need to update A_A and ω_A, and thus we obtain a normalized u_A, i.e.,

$$A_A =\leftarrow \left(\mathbf{1}_A^T G_A^{-1} \mathbf{1}_A\right)^{-1/2}, \tag{2.28}$$

$$\omega_A \leftarrow s \cdot A_A G_A^{-1} \mathbf{1}_A \quad \text{and} \tag{2.29}$$

$$u_A \leftarrow X_A^* \omega_A. \tag{2.30}$$

In the third step, we calculate the distance or magnitude of changes ρ_1. To have an efficient optimization procedure, this ρ_1 should be as large as possible. At the same time, we have to guarantee that correlations of variables in A are always identical to each other in A and larger than correlations of variables in I. Therefore ρ is increased until the correlation of a particular variable in I is equivalent to the correlations of active variables, i.e.,

$$\rho_1 = \min_{j \in A^C}^{+} \left\{ \frac{\hat{C} - \hat{c}_j}{A_A - a_j}, \frac{\hat{C} + \hat{c}_j}{A_A + a_j} \right\}, \tag{2.31}$$

where A_C is the complement of A, $a = X_A^{*T} u_A$, a_j is the j^{th} entry of a, $\hat{C}$ is the largest correlation defined in (2.25) and obtained in the first step, and ρ_1 is a possible candidate of ρ mentioned in (2.26).

According to LARS, to obtain an identical solution to MEN defined in (2.23), the lasso modification is considered, i.e., the argument of the distance ρ stops increasing when a coefficient of variables in A is zero, or mathematically,

$$W_{Ak}^* = W_{Ak-1}^* + \rho_2 s_A \omega_A = 0, \tag{2.32}$$

where ρ_2 is another possible candidate of ρ defined in (2.26). According to (2.32), we can obtain

$$\rho_2 = \min^{+} \left\{ -W_{Ak-1}^* / s_A \omega_A \right\}. \tag{2.33}$$

Therefore, the distance of W^*, i.e., ρ, is the minimum of ρ_1 and ρ_2, i.e.,

$$\rho = \min^+ \{\rho_1, \rho_2\}. \tag{2.34}$$

In each loop, one new variable is added to the active set A according to (2.25), the direction and distance of the coefficient vector W^* are calculated according to (2.30) and (2.34). After K loops, W^* is K-sparse. According to the elastic net, to eliminate the double shrinkage, the optimal W should be corrected:

$$W = \sqrt{1 + \lambda_2} W^*. \tag{2.35}$$

2.2.6 Fast LARS

LARS is inefficient when the size of the training set is large, because the time cost for calculating the inverse of the Gram matrix G_A defined in (2.27) is huge. Because the dimension of this G_A is increasing at each of the K loops, according to [99], the inverse of G_A can be obtained incrementally, i.e., the inverse of the Gram matrix $(G_{A_k})^{-1}$ in the k^{th} loop can be updated from $\left(G_{A_{k-1}}\right)^{-1}$ in the previous loop. Particularly, in the k^{th} loop, a new variable $(X)_i \in \mathbb{R}^n$ is added to the active set A, and thus we have

$$\begin{aligned}
G_{A_k} =& X_{A_k}^{*T} X_{A_k}^* = X_{A_k}^T X_{A_k} + 2\lambda_2 I \\
=& \begin{bmatrix} X_{A_{k-1}}^T \\ (X)_i^T \end{bmatrix} \begin{bmatrix} X_{A_{k-1}} & (X)_i \end{bmatrix} + 2\lambda_2 I \\
=& \begin{bmatrix} X_{A_{k-1}}^T X_{A_{k-1}} & X_{A_{k-1}}^T (X)_i \\ (X)_i^T X_{A_{k-1}} & (X)_i^T (X)_i \end{bmatrix} + 2\lambda_2 I \\
=& \begin{bmatrix} X_{A_{k-1}}^T X_{A_{k-1}} + 2\lambda_2 I & X_{A_{k-1}}^T (X)_i \\ (X)_i^T X_{A_{k-1}} & (X)_i^T (X)_i + 2\lambda_2 \end{bmatrix}.
\end{aligned} \tag{2.36}$$

Let A, B, C and D be the blocks of G_A, i.e., $A = X_{A_{k-1}}^T X_{A_{k-1}} + 2\lambda_2 I$, $B = X_{A_{k-1}}^T (X)_i$, $C = (X)_i^T X_{A_{k-1}}$, and $D = (X)_i^T (X)_i + 2\lambda_2$. Let S_A to be the Schur complement of A, i.e., $S_A = D - CA^{-1}B$. According to rules of the block

matrix calculation, $(G_{A_k})^{-1}$ is given by:

$$(G_{A_k})^{-1} = \begin{bmatrix} A^{-1} + A^{-1}BS_A^{-1}CA^{-1} & -A^{-1}BS_A^{-1} \\ -S_A^{-1}CA^{-1} & S_A^{-1} \end{bmatrix}, \tag{2.37}$$

where $A^{-1} = \left(G_{A_{k-1}}\right)^{-1}$ is the inverse of the Gram matrix obtained in the previous loop. The time cost for calculating the inverse of the Gram matrix in the k^{th} loop can be reduced from $\mathcal{O}(p^3)$ to $\mathcal{O}(p^2 + 5p)$ (p is the size of active set in the k^{th} loop) when the inverse of the Gram matrix in the previous loop is available.

We can further accelerate the computation of LARS for MEN by taking the advantage of the sparse structure of X^*. For example, when calculating the equiangular vector a and the inner product G_A, the block matrix calculation can reduce the time cost as well.

2.2.7 Algorithm

In this chapter, we propose an efficient framework MEN for discriminative dimensionality reduction with sparse projection. Based on the discussion in the above

six subsections, MEN is shown in Algorithm 1.

Algorithm 1: Manifold Elastic Net (MEN)

Input: Training data matrix $X = [x_1, x_2, \cdots, c_n] \in \mathbb{R}^{n \times p}$; Class label vector $C = [c_1, c_2, \cdots, c_n]^T$; $W = [w_1, w_2, \cdots, w_d] \in \mathbf{0}^{p \times d}$, where d is the dimensions of subspace

Output: Sparse projection matrix $W = [w_1, w_2, \cdots, w_d] \in \mathbb{R}^{p \times d}$

for $k \leftarrow 1$ **to** d **do**

Optional PCA reconstruction of original data X;

Part optimization: build n patches for the n given samples according to definition of manifold, calculate matrix L_i for each patch using (2.3) and (2.4);

Whole alignment: unify the patches in a global coordinate, compute big matrix L using (2.9);

Classification error minimization: Calculate the indicator matrix Y using scaled PCA for class centers using (2.13;

New data matrix and indicator matrix: Calculate X^* and Y^* from X and Y using (2.21) and (2.22);

for $m \leftarrow 1$ **to** K **do**

Update active set: add the variable with largest correlation to A using (2.24) and (2.25); Direction calculation using (2.29), (2.30) and fast LARS (2.37); Distance calculation using (2.31), (2.33) and (2.34); Update w_k using (2.35);

end

Update projection matrix W by adding w_k into W;

end

In MEN, after necessary initializations, we first build patches for all training samples by calculating L_i of each patch in the part optimization according to (2.4) in subsection 1. Then these L_i matrixes are unified in a global coordinate system into one matrix L according to (2.9) in whole alignment step explained in Section 2.2.2. Afterwards, the indicator matrix Y is computed according to the weighted PCA over class centers defined in (2.13) in Section 2.2.3. A matrix A defined in (2.15) in the objective function can be obtained from L and other parameters. The eigenvalue decomposition is conducted over $\left(A + A^T\right)/2$ to construct the

new data matrix X^* and the new indicator matrix Y^* according to (2.21) and (2.22), respectively.

Then the LARS algorithm is applied to calculate a sparse projection matrix. The direction and distance of each loop are computed according to (2.30) and (2.34). The incremental method to obtain the inverse of the Gram matrix defined in (2.37) is considered speeding up LARS. This process is conducted several times and the projection matrix is computed column by column. Finally a sparse projection matrix is obtained as the output of MEN. This matrix is ready to project a given sample in $\mathbb{R}^p$ to a low dimensional subspace $\mathbb{R}^d$ with K-sparse.

MEN is an efficient algorithm with high convergence velocity, because the computation in LARS explained in subsections 5 and 6 is equivalent to the cost of a least square fit. Given a training set $X \in \mathbb{R}^{n\times p}$, to obtain a sparse matrix $W \in \mathbb{R}^{p\times d}$ each column of which contains K nonzero elements, d times of LARS are required in MEN. Most steps in LARS are simple matrix computations. For $p \gg n$, MEN requires $\mathcal{O}(dK^3 + dpK^2)$ operations.

2.2.8 Discussions

MEN integrates the merits of both manifold learning and sparse learning via a unified framework. It is not a direct combination of these two popular learning schemes but a complementary embedding of both. Through the patch alignment framework, the local geometry of a given dataset is retained in MEN. The weighted lasso and ℓ_2 penalties are added to produce a sparse projection matrix with the grouping effect. The combined lasso and ℓ_2 is also termed as the elastic net. Therefore, we term the proposed framework as the manifold elastic net. As a consequence, MEN is superior to existing dimensionality reduction algorithms, because of its powerful variable selection function and consideration of the intrinsic structure of the dataset.

It has been well demonstrated that LARS is effective and efficient to solve a lasso regularized least square problem. Therefore, to apply LARS to find the optimal solution of MEN, it is essential to prove that MEN is equivalent to a lasso regularized least square problem and LARS converges for optimization. In particular, we prove that LARS can optimize a general form of the lasso

regularized problem, which contains both MEN and the lasso regularized least square problem as special cases.

Theorem 1. *LARS can solve a general form of the lasso regularized problem defined below:*

$$\arg\min_{\beta} \beta^T A\beta + \beta^T B + C + t\|\beta\|_1, \tag{2.38}$$

where $\beta \in \mathbb{R}^{p\times 1}$ *and* $A \in \mathbb{R}^{p\times p}$ *(could be an asymmetric square matrix),* $B \in \mathbb{R}^{p\times 1}$*, and* C *and* t *are constants.*

Proof. It is equivalent to prove that the problem defined in (2.38) is equivalent to a lasso regularized least square problem.

The objective function defined in (2.38) without the lass penalty can be written as:

$$\beta^T A\beta + \beta^T B + C = \beta^T \left(\frac{A + A^T}{2}\right)\beta + \beta^T B + C, \tag{2.39}$$

where $\left(A + A^T\right)/2 \in \mathbb{R}^{p\times p}$ is a symmetric matrix and its eigenvalue decomposition is $\left(A + A^T\right)/2 = UDU^T$.

Therefore, we have:

$$\begin{aligned}
&\beta^T \left(\frac{A + A^T}{2}\right)\beta + \beta^T B + C \\
=&\beta^T \left(D^{1/2}U^T\right)^T \left(D^{1/2}U^T\right)\beta - \\
&2\beta^T \left(D^{1/2}U^T\right)^T \left(-\frac{1}{2}\left(\left(D^{1/2}U^T\right)^T\right)^{-1} B\right) + C \\
=&\left\|\left(-\frac{1}{2}\left(\left(D^{1/2}U^T\right)^T\right)^{-1} B\right) - \left(D^{1/2}U^T\right)\beta\right\|_2^2 + \text{const.}
\end{aligned} \tag{2.40}$$

To simply represent the above objective function, without loss of generality, let

$$Y = -\frac{1}{2}\left(\left(D^{1/2}U^T\right)^T\right)^{-1} B, X = \left(D^{1/2}U^T\right), \tag{2.41}$$

and ignore the constant. Therefore, we can transform the problem defined in

(2.38) to

$$\arg\min_{\beta} \|Y - X\beta\|_2^2 + t\|\beta\|_1, \tag{2.42}$$

which is a lasso regularized least square problem. It is not difficult to prove that MEN is a special case of the problem defined in (refequ:theo1). Therefore, LARS can be applied to solve MEN and the problem defined in (2.38). □

Theorem 2. *LARS converges in optimizing the problem defined in (2.38) in Theorem 1.*

Proof. Let the objective function defined in (2.38) without the lasso penalty be F. After the k^{th} loop, assume the estimate of the objective function becomes F_k. If F is smooth in each loop, we have:

$$\begin{aligned}\frac{F_k - F_{k-1}}{\omega_i} \in \Bigg[&\min\left\{\frac{\partial F_k}{\partial \beta_i}\Big|_{\beta_i=\beta_i^k}, \frac{\partial F_k}{\partial \beta_i}\Big|_{\beta_i=\beta_i^{k-1}}\right\}, \\ &\max\left\{\frac{\partial F_k}{\partial \beta_i}\Big|_{\beta_i=\beta_i^k}, \frac{\partial F_k}{\partial \beta_i}\Big|_{\beta_i=\beta_i^{k-1}}\right\}\Bigg],\end{aligned} \tag{2.43}$$

where β_i is the i^{th} element in coefficient vector β, and ω is the change of β between two consecutive loops, i.e., $\omega = \beta^k - \beta^{k-1} = [\omega_1, \omega_2, \cdots, \omega_p]^T$.

In LARS for the problem defined in (2.38), the sign of ω is the negative gradient of objective function F on β^{k-1}, i.e.,

$$\text{sign}(\omega_i) = \text{sign}\left(-\frac{\partial F_k}{\partial \beta_i}\Big|_{\beta_i=\beta_i^{k-1}}\right). \tag{2.44}$$

In each loop of LARS, when correlation of one active variable becomes zeros, the length of the coefficient path will stop increasing. Therefore, the sign vector of correlations will not change in one loop, i.e.,

$$\text{sign}\left(-\frac{\partial F_k}{\partial \beta_i}\Big|_{\beta_i=\beta_i^k}\right) = \text{sign}\left(-\frac{\partial F_k}{\partial \beta_i}\Big|_{\beta_i=\beta_i^{k-1}}\right) = \text{sign}\left(\frac{F_k - F_{k-1}}{\omega_i}\right) = -\text{sign}(\omega_i)$$

According to the analyses, we can obtain the sign of $(F_k - F_{k-1})$:

$$\text{sign}(F_k - F_{k-1}) = -\text{sign}(\omega) \cdot \text{sign}(\omega) = -1. \tag{2.45}$$

According to the above equation, the objective function F is monotonic. In addition, F is bounded. Therefore, we can safely draw the conclusion that LARS converges in optimizing the problem defined in (2.38). □

2.3 Experiments

In this section, we evaluate the performance of MEN by comparing against six representative dimensionality reduction algorithms, i.e., principal component analysis (PCA), Fisher's linear discriminant analysis (FLDA), discriminative locality alignment (DLA) [249][250], supervised locality preserving projection (SLPP), neighborhood preserving embedding (NPE), and sparse principal somponent analysis (SPCA), on three standard face image databases, i.e., UMIST [11], FERET [188] and YALE [13].

PCA is an unsupervised linear dimensionality reduction algorithm which projects the data along the direction of maximal variance. FLDA is a supervised linear dimensionality reduction method. SLPP is a supervised modification of the locality preserving projections, which is a linearization of the Laplacian Eigenmap. NPE is a linear approximation to the locally linear embedding (LLE). SPCA is a sparse dimensionality reduction algorithm which combines the lasso penalty with PCA to produce sparse loadings.

Three standard face image datasets, e.g., UMIST, FERET and YALE, are utilized in this chapter to evaluate the proposed MEN for discriminative dimensionality reduction. There are 565 face images from 20 individuals in the UMIST dataset. The samples demonstrate variations in race, gender, pose and appearance. The FERET dataset consists of $13,539$ face images from $1,565$ individuals. The images vary in size, gender, pose, illumination, facial expression and age. We randomly select 100 individuals, each of which has 7 images from FERET for performance evaluation. The YALE dataset contains 165 face images of 15 individuals. Lighting conditions, gender, facial expressions and configurations are different among these images. All images from these three databases are normalized to 40×40 pixel arrays with 256 gray levels per pixel. Figure 2.1 shows sample images from these three datasets. Each image is reshaped to a long vector by concatenating its pixel values in a particular order.

Figure 2.1: Sample face images from the three databases. The first row comes from UMIST; the second row comes from FERET; and the third row comes from YALE.

Different algorithms follow an equivalent procedure for all face recognition experiments on various datasets. Firstly, the database is randomly divided into two separate sets: training set and testing set. Then the training set is used to learn the low dimensional subspace and corresponding projection matrix through given algorithm. After this, samples in the testing set are projected to a low dimensional subspace via the projection matrix. Finally, the nearest neighbor classifier is used to recognize testing samples in the subspace.

We apply PCA to reduce dimensions of original high dimensional face images before FLDA, DLA, LPP (with supervised setting) and NPE (with supervised setting). For FLDA, we retain $n - c$ dimensions in the PCA projection, where n is the number of samples and c is the number of classes. We project samples to the PCA subspace with $n - 1$ dimensions for DLA, SLPP and NPE.

For UMIST and YALE, we randomly select $p = (5, 7)$ images per individual for training, while the remaining images are used as testing samples. For FERET, $p = (4, 5)$ images per individual are selected as training set, and the remaining for testing. All experiments are repeated five times, and the average recognition rates are calculated.

The results of these dimensionality reduction algorithms on two settings of

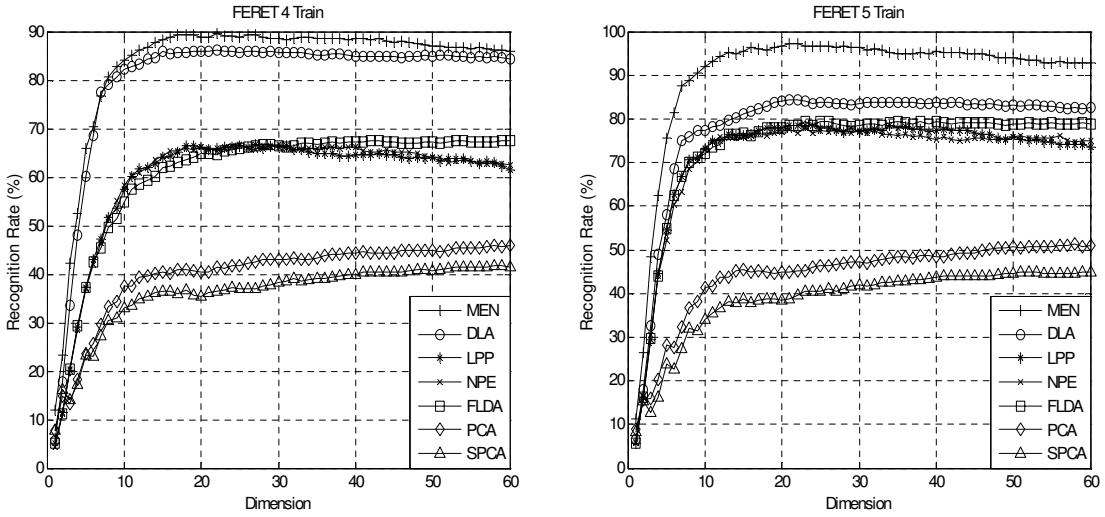

Figure 2.2: Recognition Rate vs. Dimension on FERET

FERET are shown in Figure 2.2. These seven algorithms can be divided into 3 groups according to their performance: PCA and SPCA are at the bottom level, because they are unsupervised and the label information is not considered. PCA is slightly better than SPCA, because SPCA is designed to approximate PCA but with less information retained to hold the sparse property. LPP, NPE and LDA are at the middle level. They are much better than PCA and SPCA because they consider the class label information. LPP and NPE preserve the local geometry based on the neighborhood information of samples, while LDA ignores the local geometry. LPP and NPE cannot perform as well as DLA and MEN because both of them ignore the margin maximization or the inter-class information. MEN and DLA are at the top level. MEN outperforms DLA because it reduces the noises by using the elastic net penalty.

Experimental results on UMIST are shown in Figure 2.3. MEN outperforms the other six algorithms consistently. Note the fact that MEN keeps having the highest recognition rate when the dimension of the selected subspace is low. This verifies the robustness of MEN in low dimension situation. In addition, the computational cost is proportional to the dimension of the selected subspace. Therefore MEN produces better results with less computational cost than other dimensionality reduction methods.

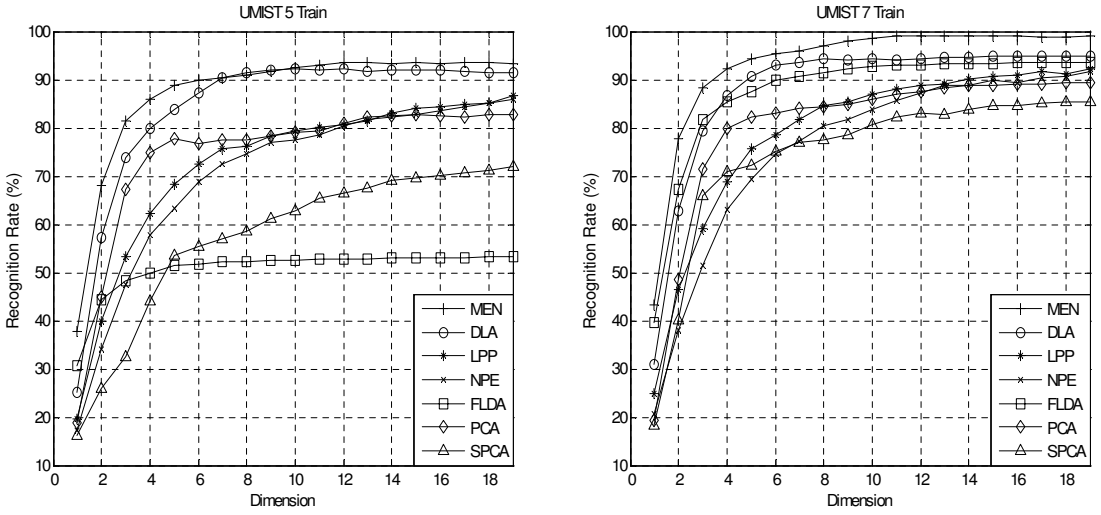

Figure 2.3: Recognition Rate vs. Dimension on UMIST

Figure 2.4 shows MEN outperforms the other six algorithms on the YALE dataset. The curves of MEN are smoother than those of the other algorithms. This implicates that MEN is more stable than the other algorithms. MEN has high recognition rate even when the training set is small and the dimensions of the selected subspace is low. The priority of MEN can be attributes to its supervised learning property, consideration of data manifold structure, feature selection ability brought by sparsity and the grouping effect. The sparsity of MEN filters out classification irrelevant features, which bring unnecessary noises for classification. This is effective especially when the number of classes is much smaller than the number of the original features. Furthermore, the sparse projection matrix brings better interpretation and lower computational cost for subsequent calculation than dense projection matrices.

Table 2.1 lists the best recognition rate and the corresponding subspace dimension for each algorithm in the experiments on the three face image datasets. Sparse dimensionality reduction algorithm including MEN and SPCA always arrive their best recognition rate in lower dimensional subspace than other five algorithms. This is because the sparsity brought by the lasso penalty is able to select the most significant features. However, because SPCA does not consider the class label information, it always performs more poorly than other supervised

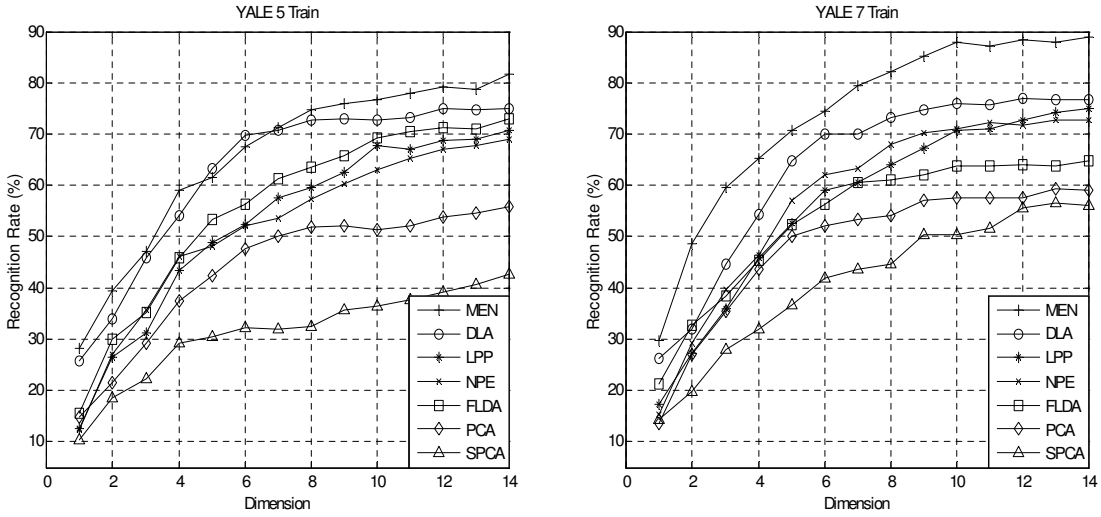

Figure 2.4: Recognition Rate vs. Dimension on YALE

algorithms. For each algorithm, the dimension of the best recognition rate is decreasing with the increasing of training samples. This is because more training samples make the low dimensional representation more stable and reliable.

Boxplots of the experimental results of these seven dimensionality reduction algorithms on the three face image datasets are shown in Figure 2.5, Figure 2.6 and Figure 2.7, respectively. Each boxplot produces a box and whisker plot for each method. The box has lines at the lower quartile, median, and upper quartile values. Whiskers extend from each end of the box to the adjacent values in the

		MEN	DLA	LPP	NPE	LDA	PCA	SPCA
FERET	4	90.67(17)	88.67(19)	74.00(17)	74.33(21)	76.33(25)	48.00(54)	45.67(41)
	5	96.50(30)	88.50(35)	83.50(36)	82.00(19)	84.00(49)	54.00(51)	48.50(58)
		MEN	DLA	LPP	NPE	LDA	PCA	SPCA
UMIST	5	95.89(17)	94.57(18)	90.11(19)	89.68(19)	88.21(18)	88.63(13)	80.63(19)
	7	99.21(16)	97.62(19)	95.40(19)	95.17(18)	97.24(14)	93.79(19)	90.57(18)
		MEN	DLA	LPP	NPE	LDA	PCA	SPCA
YALE	5	82.78(13)	79.11(12)	79.33(13)	77.11(14)	82.22(12)	61.11(12)	63.33(13)
	7	90.33(12)	87.00(12)	85.00(13)	84.33(11)	81.67(11)	66.67(13)	63.33(12)

Table 2.1: Best recognition rate (%) on three databases. For MEN, DLA, LPP (SLPP), NPE, LDA (FLDA), PCA, SPCA (Sparse PCA), the numbers in the parentheses behind the recognition rates are the subspace dimensions. Numbers in the second column denote the number of training samples per individual.

data-by default and the most extreme values within 1.5 times the interquartile range from the ends of the box.

MEN achieves the most robust recognition rate, because it considers the sparse property, the local geometry of intra-class samples, and the margin maximization and classification error minimization of inter-class samples. MEN selects features with the largest correlation and eliminates the most unstable ones. Manifold learning methods, such as LPP, DLA and NPE, as well as LDA are more stable than PCA and SPCA according to these boxplots because they consider the class label information.

Figure 2.8, Figure 2.9 and Figure 2.10 show the columns of the projection matrix of the seven algorithms on the three face image datasets. The low dimensional subspace is spanned by the column vectors, which is called bases. The bases of PCA are called Eigenfaces [222], while the bases of LDA are called Fisherfaces [111] in previous literatures. Similar methods can be applied to DLA, SLPP, NPE, SPCA and MEN. The bases of MEN are sparser and have less noise than PCA and DLA because of its sparsity, and more grouping than SPCA because of its grouping effect adopted from the ℓ_2 penalty. Sparse bases lead to computational efficiency and good interpretation. According to Figure 2.8, Figure 2.9 and Figure 2.10, "MEN faces" retain the most discriminative facial features, e.g., eyebrows, eyes, nose, mouth, ears and facial contours, while leave the other parts blank. "SPCA faces" are sparse but without the grouping effect, their facial contours and organs are represented by some isolate points. "LPP faces" and "NPE faces" are very similar in appearances and this fact well explains that they perform comparably in these datasets. "DLA faces" have better description of features and less noises than those obtained by LPP, NPE and FLDA.

In each LARS loop of the MEN algorithm, according to the algorithm listed in Algorithm 1, all entries of one column in the MEN projection matrix are zeros initially. They are sequentially added into the active set according to their importance. The values of active ones are increased with equal altering correlation. In this process, the ℓ_1-norm of the column vector is augmented. Figure 2.11 shows the altering tracks of some entries of the column vector in one LARS loop. We called these tracks "coefficient path" in LARS. In Figure 2.11, every coefficient path starts from zero when the corresponding variable becomes active, and

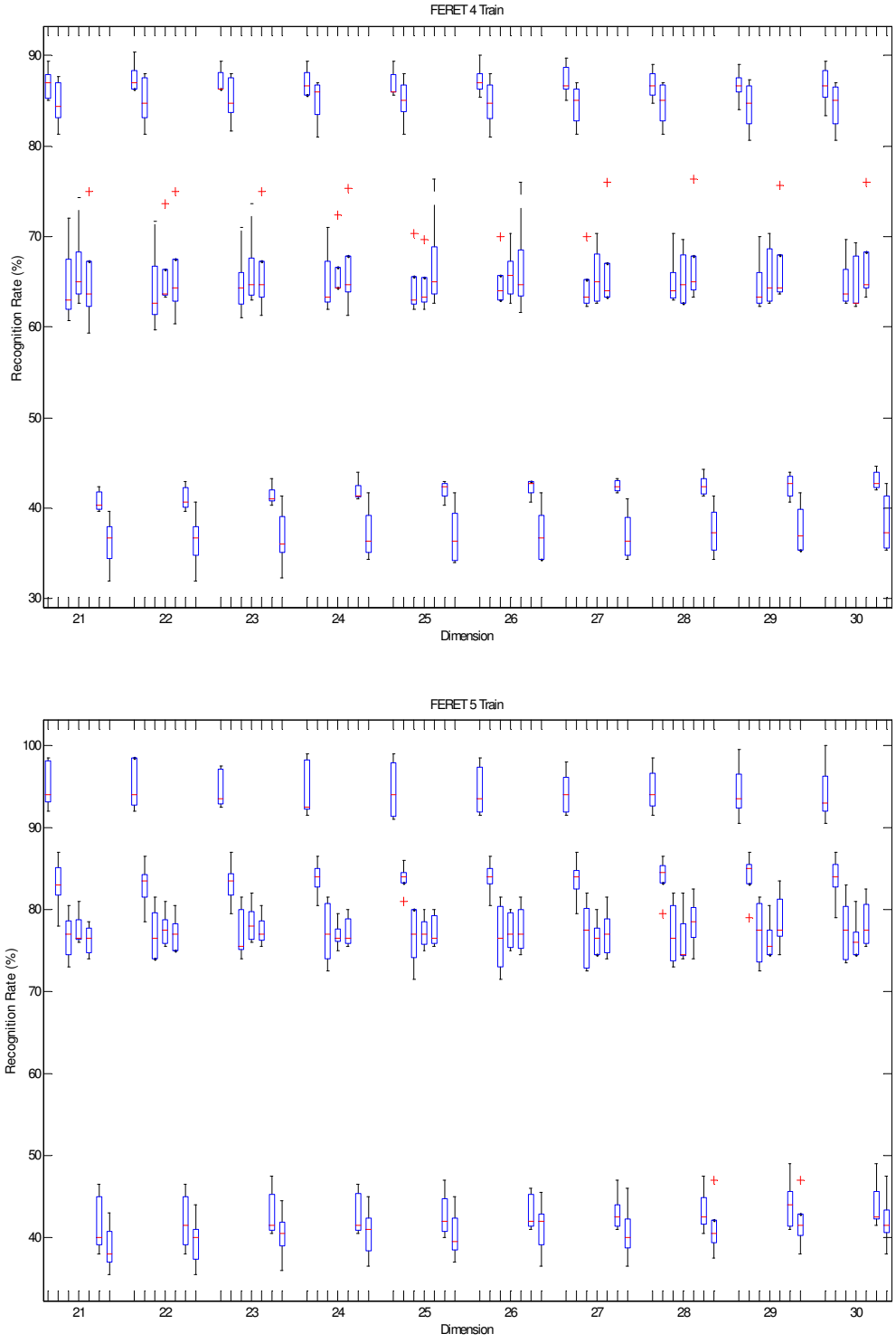

Figure 2.5: Boxplot of Recognition Rate vs. Dimension (from 21 to 30) on FERET with 4 (5) training samples per person. For every dimension, from left to right, the seven boxes refer to MEN, DLA, LPP, NPE, FLDA, PCA, and SPCA.

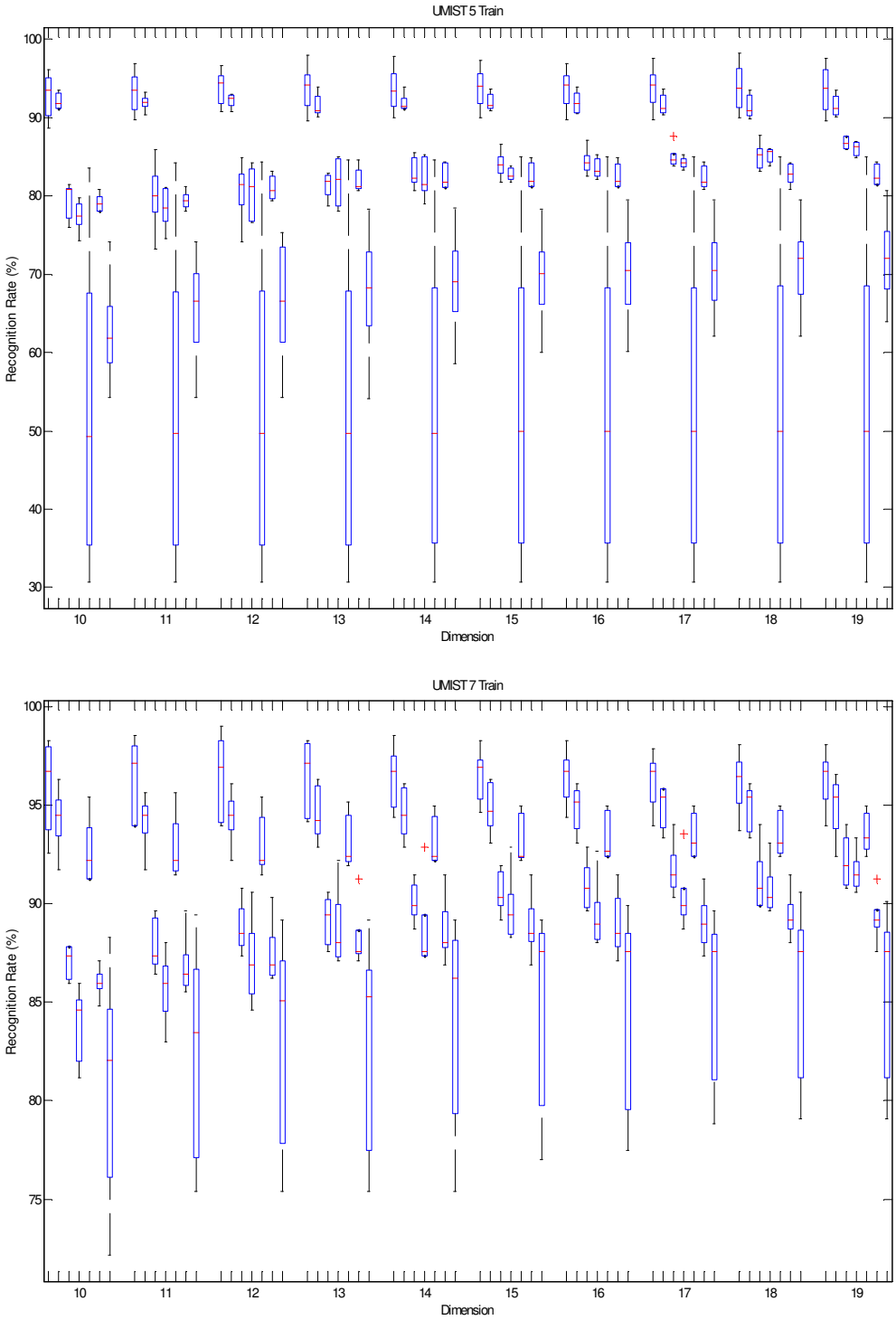

Figure 2.6: Boxplot of Recognition Rate vs. Dimension (from 10 to 19) on UMIST with 5 (7) training samples per person. For every dimension, from left to right, the seven boxes refer to MEN, DLA, LPP, NPE, FLDA, PCA, and SPCA.

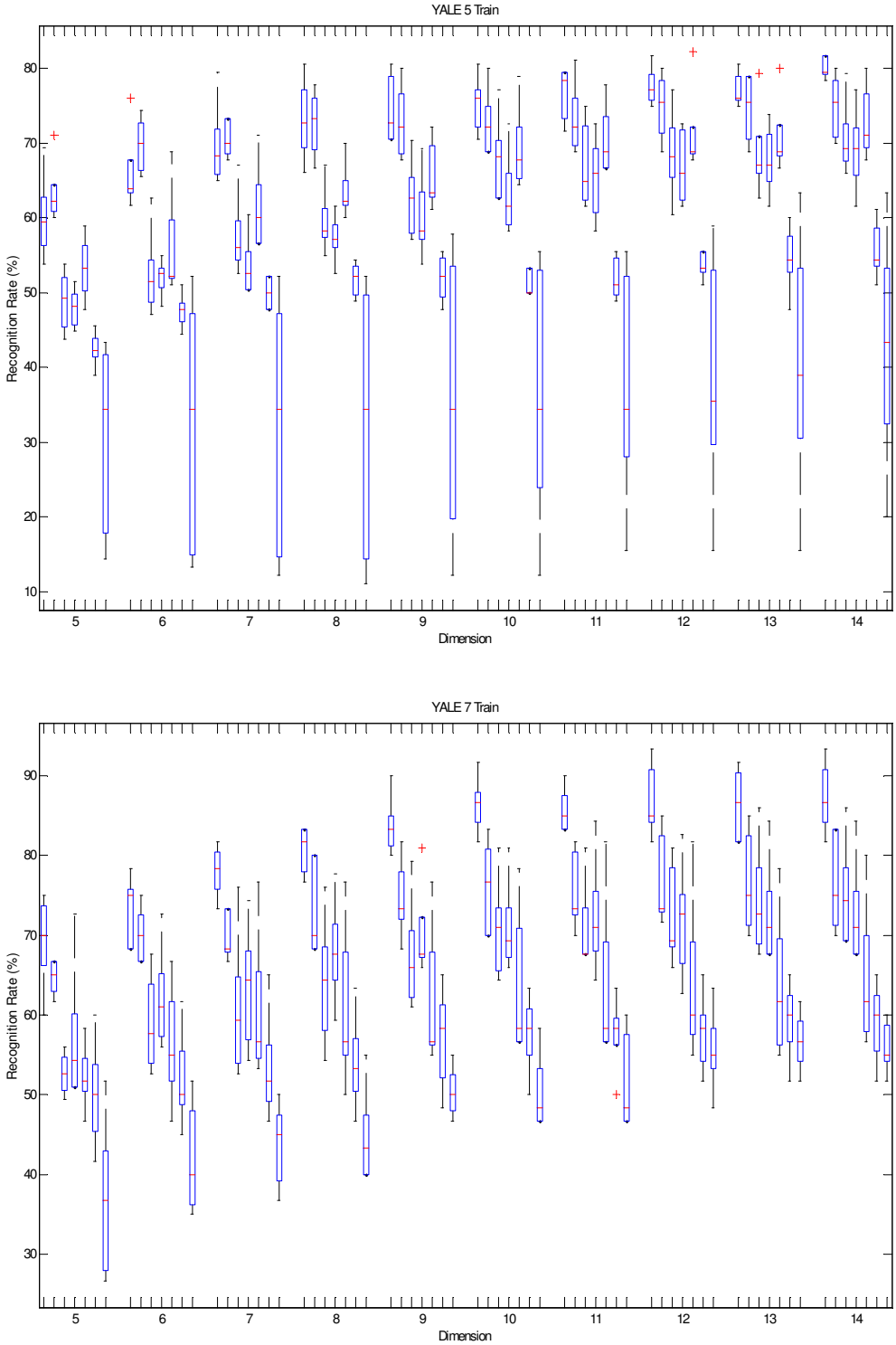

Figure 2.7: Boxplot of Recognition Rate vs. Dimension (from 5 to 14) on YALE with 5 (7) training samples per person. For every dimension, from left to right, the seven boxes refer to MEN, DLA, LPP, NPE, FLDA, PCA, and SPCA.

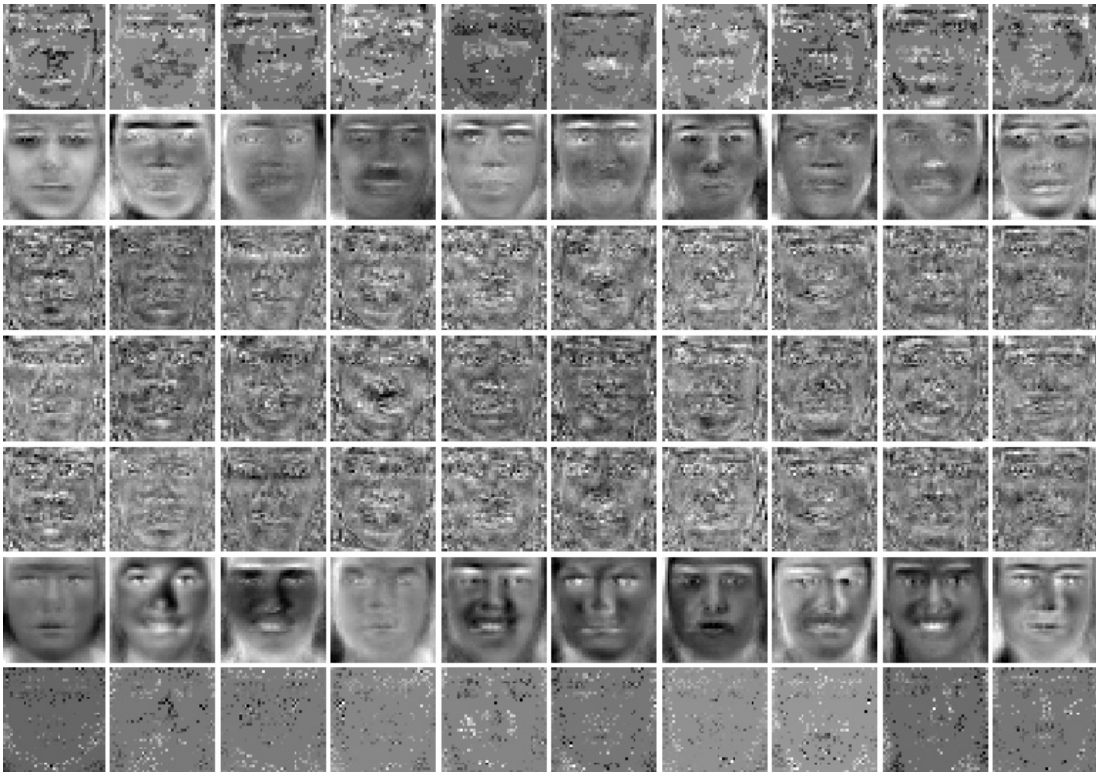

Figure 2.8: Plots of first 10 bases obtained from 7 dimensionality reduction algorithms on FERET For each column, from top to bottom: MEN, DLA, LPP, NPE, FLDA, PCA, and SPCA

changes its direction when another variable is added into the active set. All the paths keep in the directions which make the correlations of their corresponding variables equally altering. The ℓ_1-norm is increasing along the greedy augment of entries. The coefficient paths proceed along the gradient decent direction of objective function on the subspace, which is spanned by the active variables.

Figure 2.12 shows 10 of the 1600 coefficient paths from LAPS loop for the first base in experiment on FERET dataset. MEN selects ten important variables (facial features) sequentially here. Each feature, its corresponding coefficient path and the"MEN fac" when the feature is added into active set are assigned the same color which is different with the other 9 features. In each "MEN face", the new added active feature is marked by a small circle, and all the active features are marked by white crosses. The features selected by MEN can produce explicit

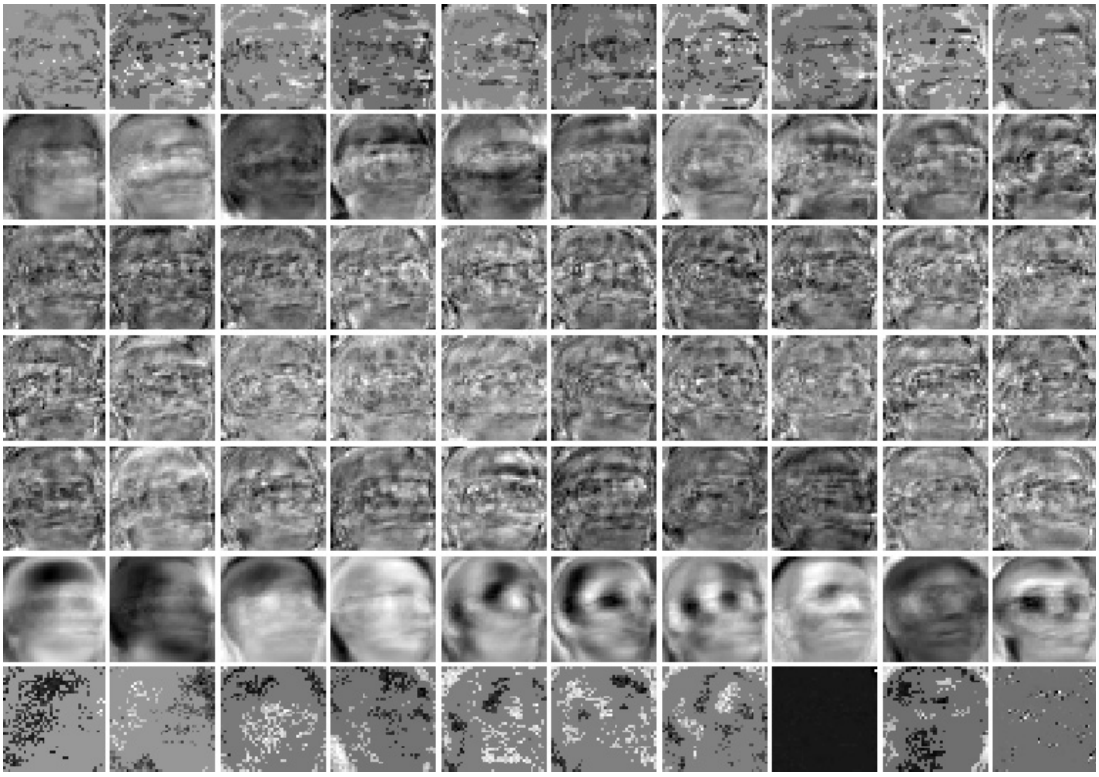

Figure 2.9: Plots of first 10 bases obtained from 7 dimensionality reduction algorithms on UMIST For each column, from top to bottom: MEN, DLA, LPP, NPE, FLDA, PCA, and SPCA

interpretation of the relationship between facial features and face recognition: feature 1 is the left ear, feature 2 is the top of nose, feature 3 is on the head contour, feature 4 is the mouth, feature 5 and feature 6 are on the left eye, feature 7 is the right ear, and feature 8 is the left corner of mouth. These features are already verified of great importance in face recognition by many other famous face recognition methods.

Moreover, Figure 2.12 also shows MEN can group correlated features, for example, feature 5 and feature 6 are selected sequentially because they are both on the left eye. In addition, features which are not very important, such as feature 9 and feature 10 in Figure 2.12, are selected after the selection of the other more significant features and assigned smaller value than those more important ones. Therefore, MEN is a powerful algorithm in variable (feature) selection.

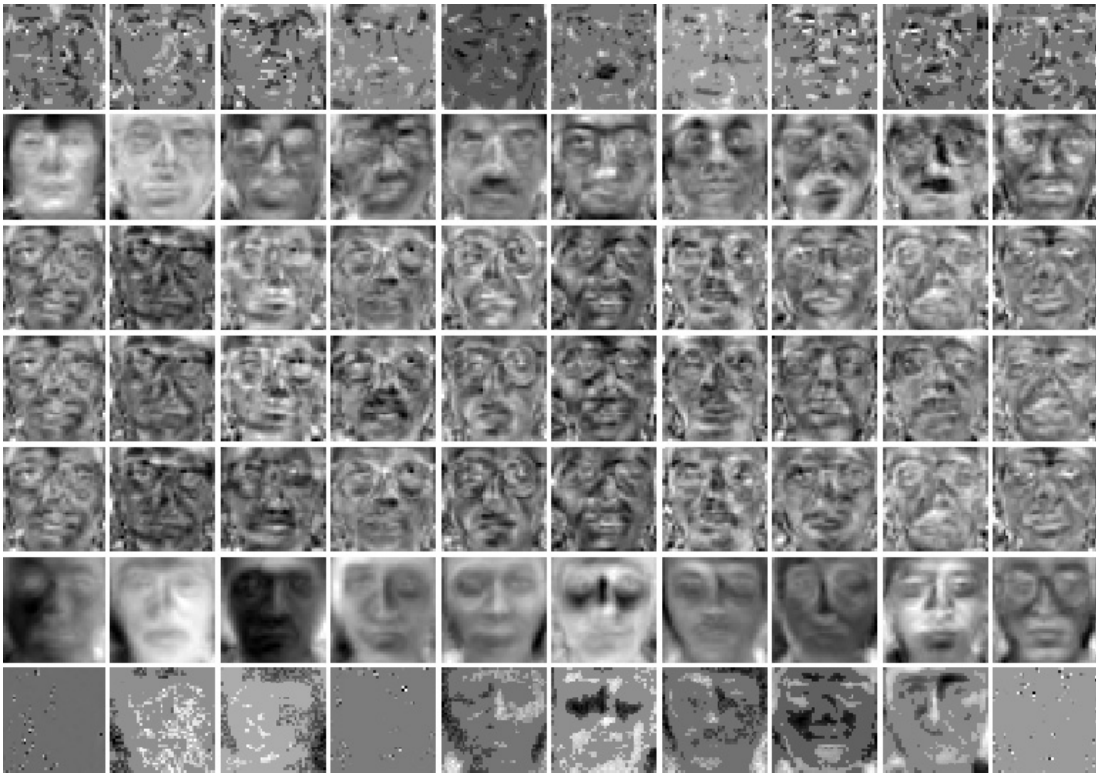

Figure 2.10: Plots of first 10 bases obtained from 7 dimensionality reduction algorithms on YALE For each column, from top to bottom: MEN, DLA, LPP, NPE, FLDA, PCA, and SPCA

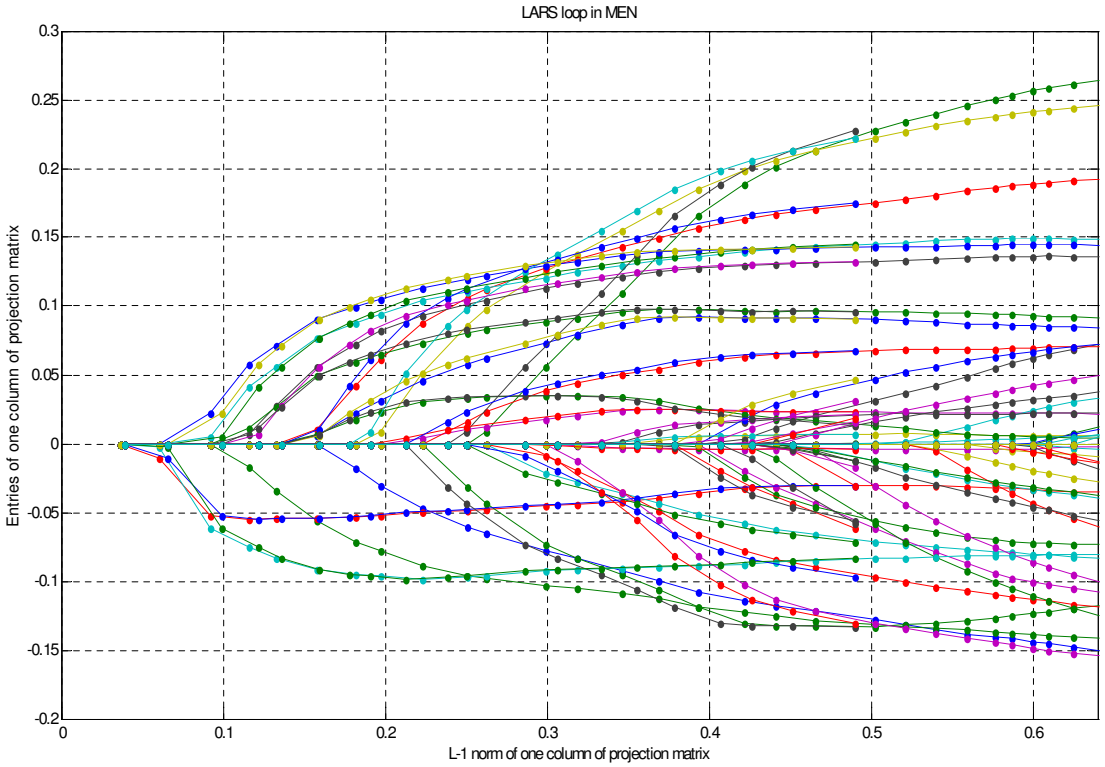

Figure 2.11: Entries of one column of projection matrix vs. its ℓ_1-norm in one LARS loop of MEN

2.4 Conclusion

In this chapter, we propose a unifying framework which obtains a sparse projection matrix for subsequent classification, termed manifold elastic net or MEN for short. MEN incorporates the advantages of both manifold learning based dimensionality reduction and sparse learning based dimensionality reduction, but it is not a direct combination of these two. To obtain a sparse projection matrix, MEN imposes the elastic net penalty over a loss function that is defined under the patch alignment framework. The objective function of MEN can be transformed into a lasso penalized least square problem by using a series of complex linear algebra equivalent transformations, and thus the least angle regression (LARS) can be applied to obtain the optimal sparse projection matrix.

In MEN, the patch alignment framework is first used to construct local patches

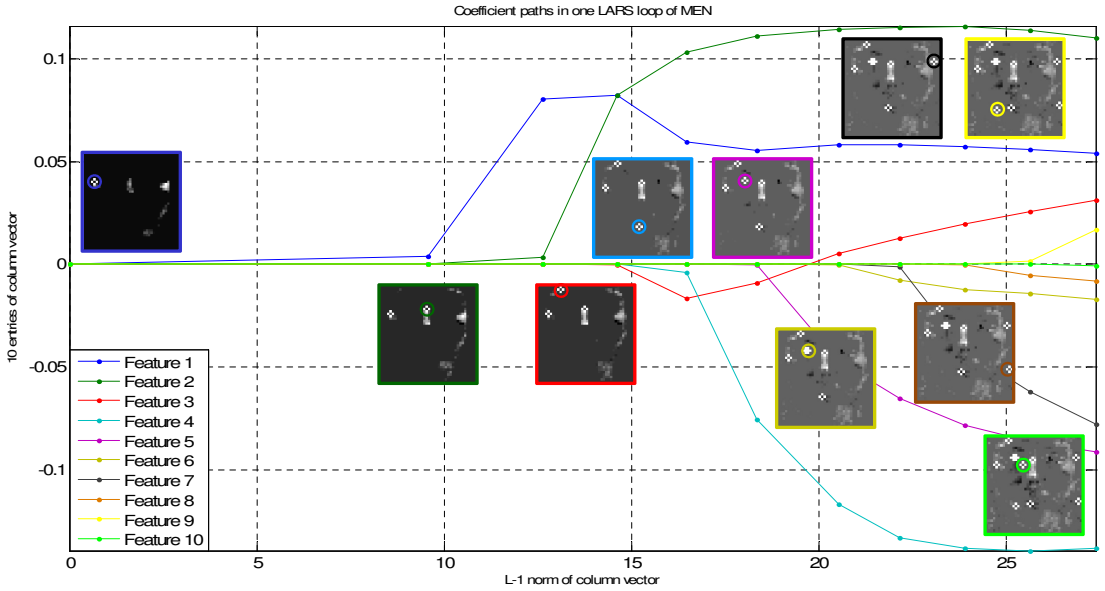

Figure 2.12: Coefficient paths of 10 entries (features) in one column vector

of data and unifies these patches into a global coordinate system. Secondly, the classification error is minimized directly via weighted principal component analysis (PCA) over class centers. Thirdly, to obtain a sparse projection matrix with the grouping effect, the elastic net penalty is added to the objective function. After a series of equivalent transformations, MEN can be rewritten as a lasso-type regression. Therefore, LARS can be applied to solve the problem efficiently. In each LARS loop for MEN optimization, important variables are added into the active set sequentially according to their correlation. All the elements in the active set are altered along a special direction with a special distance in each step. The special direction and distance keep the correlation of active elements identical and the largest in a LARS loop. The procedure is conducted several times to obtain a set of sparse bases because these bases are independent.

MEN enjoys advantages in several aspects: 1) the local geometry of intra-class samples is well preserved for low dimensional data representation, 2) both the margin maximization and the classification error minimization are considered for discriminative information preservation, 3) the sparsity of the projection matrix of MEN improves the parsimony in computation, 4) the elastic net penalty reduces

the over-fitting problem, and 5) the projection matrix of MEN can be interpreted psychologically and physiologically.

Experimental results of face recognition on UMIST, FERET and YALE show that MEN performs better and more stable than popular dimensionality reduction algorithms, such as the principal component analysis (PCA), Fisher's linear discriminant analysis (FLDA), the discriminative locality alignment (DLA), the locality preserving projections with supervised setting (LPP), the neighborhood preserving embedding with supervised setting (NPE), and the sparse principal component analysis (SPCA).

There are still many interesting properties of MEN which have not been targeted and formally proved in this chapter. In the future, we will analyze its error bounds under different situations. Another important problem in MEN is how to choose the optimal sparsity, so that we can remove most noise and retain most discriminative information for subsequent classification. The compressed sensing may be an effective tool to address the above concern. It is also valuable to replace the lasso penalty with the ℓ_0-norm penalty to further improve MEN with more "accurate sparsity". The lasso penalty is a relaxation of ℓ_0-norm penalty, and there are alternatives which could perform better than the lasso penalty, e.g., the smoothly clipped absolute deviation penalty (SCAD) [128], the reweighted ℓ_1 minimization [42], the adaptive lasso [274] and the adaptive elastic net. The advantages of these methods can be adopted in MEN to further enhance the variable selection ability of MEN, and there is still a long way to go.

Chapter 3

Double Shrinking for Sparse Dimension Reduction

Learning tasks such as classification and clustering usually perform better and cost less (time and space) on compressed representations rather than on the original data. Previous works mainly compress data via dimension reduction. In this chapter, we propose "double shrinking" to compress image data on both dimensionality and cardinality via building either sparse low dimensional representations or a sparse projection matrix for dimension reduction. We formulate double shrinking model (DSM) as an ℓ_1 regularized variance maximization with constraint $\|x\|_2 = 1$, and develop double shrinking algorithm (DSA) to optimize DSM. DSA is a path-following algorithm that can build the whole solution path of locally optimal solutions of different sparse levels. Each solution on the path is the "warm start" for searching the next sparser one. In each iterate of DSA, the direction, the step size and the Lagrangian multiplier are deduced from the Karush-Kuhn-Tucker (KKT) conditions. The magnitudes of trivial variables are shrunk and the importances of critical variables are simultaneously augmented along the selected direction with the determined step length. Double shrinking can be applied to manifold learning and feature selection for better interpretation of features, and can be combined with classification and clustering to boost their performance. The experimental results suggest that double shrinking produces efficient and effective data compression.

3.1 Introduction

Sparsity has been widely exploited to compress information, obtain efficient coding of massive data and select important features. In signal processing, compressed sensing [40][71][206][166][261] has proved that a sparse signal can be exactly recovered from a small number of its random projections. In statistics, *lasso* [211] and other ℓ_1 regularized regression models [160][161][166] are proposed to select important variables for building a parsimonious prediction model of a response. The success of sparsity and the ℓ_1 regularization for various applications is supported by various facts. For example, images often have sparse representations [172] after specific transformations such as cosine transform (DCT) and multi-scale geometric analysis (MGA) [8]; and many types of data, such as face images and non-coding RNAs, are usually obtained in the form of redundant features yet insufficient samples.

Dimension reduction has been broadly applied to machine learning for compressing image data and preserving important information in the low-dimensional subspace. For example, principal components analysis (PCA) [119][263] maximizes the mutual information between the original high-dimensional Gaussian distributed samples and the projected low-dimensional samples. Fisher's linear discriminant analysis (FLDA) [87] retains the discriminative information by maximizing the between-class scatter and minimizing the within-class scatter. Manifold learning algorithms [196][210] preserve the important geometric structure of images. In order to reduce the computational complexity and improve the performance, learning tasks such as classification [270] and clustering are usually conducted on the dimensionality reduced subspace instead of the original high dimensional space. Dimensionality reduction using feature selection has been formally shown to be important and effective in high dimensional classification in [82], who characterized the impact of dimensionality on classification and proposed the FAIR method for high-dimensional classification.

Compressed sensing compresses images by exploiting their sparsity. Dimension reduction compresses images by preserving the important informations. In this chapter, we seamlessly integrate them together in a sparse learning framework "double shrinking" that compresses image data on both dimensionality and

cardinality. To retain preferred information underlain in the high dimensional and dense features, it either directly compresses the data to low dimensional and sparse representations or finds a low dimensional and sparse projection matrix to obtain low dimensional approximations. Our experimental results suggest that each case has its own advantages on different applications. The sparse representation reduces the space cost and offers explicit interpretations to new coordinates. Moreover, sparse representations of samples in the same class or cluster tend to share the same support set, and thus the subsequent classification and clustering can be improved. The sparse projection matrix for linear dimension reduction saves the time cost of projection and provides explicit interpretations to selected features. Our experimental results suggest that double shrinking can produce competitive performance in many learning tasks comparing with dimension reduction algorithms that we are aware of.

3.1.1 Double shrinking model

In this chapter, we formulate double shrinking model (DSM) by introducing the ℓ_1 regularization to the conventional dimension reduction problem. Thus DSM can be written as

$$\min_x \ x^T P x + \mu \|x\|_1 \quad s.t.\ x^T x = 1, \tag{3.1}$$

where $x^T P x$ refers to the conventional manifold embedding. This P is borrowed from [17] to retain important information for different applications and includes popular dimension reduction algorithms as special cases, such as locally linear embedding (LLE) [196], ISOMAP [210], Hessian eigenmaps [72], Laplacian eigenmaps [15], and their respective linear approximations [112]. For example, if P is the normalized graph Laplacian $L = I - D^{-1/2} W D^{-1/2}$ [182], wherein W is the adjacency matrix and D is the degree matrix, the solution x is the low dimensional and sparse representation. If P is $X^T L X$, wherein X is the data matrix and L is the normalized graph Laplacian, the solution x is the sparse projection matrix and the corresponding low dimensional representation is Xx. The sparsity of the representation or the projection matrix is due to the ℓ_1 regularization in (3.1). Sparse eigenvalue maximization problem can be equivalently transformed to the sparse eigenvalue minimization problem (3.1) by changing the object from

maximizing $x^T Px - \mu\|x\|_1$ to minimizing $x^T(-P)x + \mu\|x\|_1$.

DSM provides the first explanation of "double shrinking", i.e., compressing data or finding a projection matrix by simultaneously shrinking dimensionality and cardinality. Most optimization problems solved by existing sparse PCA methods are relaxations of DSM or have similar forms with DSM. However, DSM is characteristic in its equality constraint $x^T x = 1$.

3.1.2 Previous works

Compared with existing works, the main challenge for optimizing DSM comes from the simultaneous appearance of the ℓ_1 regularization and the equality constraint $x^T x = 1$. Although either of them has been independently tackled in special optimization algorithms, such as *lasso* [211] and PCA [119], the direct optimization without relaxations for a problem simultaneously containing both is rarely found. We prefer the constraint $x^T x = 1$ because it is a critical constraint existing in most dimension reduction methods to avoid trivial solution such as all-zeros. In DSM, this constraint keeps the solution on the unit ball so the unimportant elements can drop to zeros quickly rather than to a very small nonzero values, and the magnitudes of important elements can be bounded.

Since the ℓ_1 norm is not differentiable, most of the frequently used optimization methods are not applicable to the ℓ_1 regularized problems. In compressed sensing and statistics, various algorithms have been developed to address the ℓ_1 regularized least square regression or the ℓ_1 norm minimization with a measurement constraint. Popular algorithms can be classified into the following four groups.

1. Greedy algorithms: Orthogonal matching pursuit (OMP) [214] and compressive sampling matching pursuit (CoSaMP) [178] sequentially select important variables by using the greedy search. The sparse solution for the compressed sensing problem is obtained by optimizing the selected variables.

2. Convex optimization based algorithms: Basis pursuit [47] doubles variables in the ℓ_1 norm minimization and then the ℓ_1 norm is replaced by the sum of

all the variables. Thus the objective becomes differentiable and the problem can be solved by linear programming. NESTA [24] adopts Nesterov's method [181] to minimize the smoothed approximation of the ℓ_1 norm and converges at rate $\mathcal{O}(1/k^2)$. An interior point algorithm based on the preconditioned conjugate gradient method is applied in [139] for solving the large scale compressed sensing problem. Coordinate gradient descent [215] and gradient projection [86] also have been introduced to the compressed sensing problem.

3. Iterative thresholding algorithms, e.g., message passing [73], iterative splitting and thresholding (IST) [61] and iterated hard shrinking [29], conduct soft or hard thresholding on the solution at each iteration round and finally obtain the sparse solution;

4. Fixed point method based algorithms: Bregman iterative algorithm [240], fixed point continuation [107] and iteratively re-weighted least squares (IRLS) [191] derive a fixed-point equation from the optimality condition of the compressed sensing problem. They can yield accurate sparse solution within a small number of iteration rounds.

However, pure greedy algorithms hardly ensure the optimality of DSM. The additional equality constraint $x^T x = 1$ in DSM makes the existing convex relaxation methods of the ℓ_1 regularized optimization invalid. Furthermore, the iterative thresholding algorithms and the fixed point method based algorithms have to change the ℓ_2 norm of the solution in each iteration round and thus violates the equality constraint $x^T x = 1$ in DSM. In sum, most of the optimization methods for optimizing the ℓ_1 regularized problem cannot be directly applied to DSM.

The problems solved by existing sparse PCA algorithms are similar to DSM. Most sparse PCA methods find sparse principal components (PCs) with large explained variance by solving two kinds of optimization: 1) sparsity constrained/regularized regression-type problem with an inequality constraint $x^T x \leq 1$ or normalization of the obtained x, e.g., Sparse PCA (SPCA) [273], sPCA-rSVD [201]; 2) sparsity constrained/regularized SDP that maximizes the explained variance of sparse

PC with an inequality constraint $x^T x \leq 1$, e.g., DSPCA [60], Path SPCA [59] and SPC in PMD [229]. Greedy method is also applied to sparse PCA in [176]. However, they cannot directly solve DSM.

In summary, it is essential to develop an effective and efficient algorithm to directly optimize DSM.

3.1.3 Main contribution

The main challenge for solving DSM comes from the ℓ_1 regularization and the equality constraint $x^T x = 1$. In this chapter, we propose DSA that builds a solution path for DSM from the dense solution to sparse ones. Each solution on the path is local optimal with respect to its associated regularization parameter. DSA starts from a point on the path and two initial sets of critical variables and trivial ones, respectively. In each iteration round, it proceeds on a direction along which the importances of the critical variables are augmented and the magnitudes of the trivial variables are shrunk until one of the following three events happens: 1) the magnitude of a trivial variable is shrunk to zero; 2) a critical variable is transferred to the trivial variable set once its magnitude is shrunk to zero; and 3) a trivial variable is transferred to the critical variable set once its importance reaches the minimum importance of the critical ones. The first two events provide the second explanation of the name "double shrinking". The direction, step size and Lagrangian multiplier in each iteration round are determined by the Karush-Kuhn-Tucker (KKT) conditions. Such continuation technique utilizes the current solution as the "warm start" of the sparser one in the next iteration round and thus accelerates the optimization. The time complexity of each iteration round is less than $\mathcal{O}(s_A^3 + s_B^2)$, wherein s_A is the number of the critical variables and s_B is the number of the trivial ones. DSA has only one free parameter.

Double shrinking can be applied to manifold learning and feature selection [167][83], and can be combined with classification [268] and clustering algorithms to boost performance. It also provides an effective scheme for other ℓ_1 regularized optimization with equality constraints. We apply double shrinking to different machine learning tasks on image datasets of face recognition, hand-written character classification, object categorization, UCI and gene expression, and ob-

tain promising performance. Some critical properties of double shrinking, i.e., variance-cardinality trade-off and speed, are evaluated and compared with existing sparse PCA methods and sparse coding methods, on both real datasets and artificial ones. The experimental results suggest that double shrinking provides an efficient and effective method for data compression.

The rest of the chapter is organized as follows. Section 3.2 defines concepts used in DSM and DSA. Section 3.3 presents DSA and related proofs. Section 3.4 shows the experimental results of double shrinking for classification, clustering and feature selection. Section 3.5 concludes the chapter.

3.2 Definitions

In this chapter, lower-case letter denotes a vector or a constant and capital letter denotes a matrix or a set. Let x_i be the i^{th} entry of a vector x and X_{ij} be the entry that lies in the i^{th} row and the j^{th} column of the matrix X. Given an index set S for a vector x, we define x_S that satisfies $(x_S)_i = x_{S_i}$. Given a row index set A and a column index set B for a matrix X, we define X_{AB} that satisfies $(X_{AB})_{ij} = X_{A_iB_j}$. We use superscripts $\cdot^k$ and $\cdot^*$ to signify a variable in the k^{th} iteration round and the final solution of DSA, respectively.

DSM (3.1) is an ℓ_1 regularized eigenvalue maximization with an equality constraint $x^Tx = 1$, and thus it has a locally optimal solution that satisfies two KKT conditions. In this section, we first show the KKT conditions of DSM by defining the subgradient of the ℓ_1 norm. Based on the KKT conditions, we define the importance and magnitude of a variable in x. Since a sparse solution x is composed of zero and nonzero variables, we then correspondingly define critical and trivial variables that will be sequentially determined and updated through DSA.

3.2.1 Karush-Kuhn-Tucker conditions

We start from the KKT conditions [28] of DSM defined in (3.1). The lagrangian L associated with (3.1) is

$$L(x, \eta) = x^TPx + \eta\left(x^Tx - 1\right) + \mu\|x\|_1. \tag{3.2}$$

Thus the KKT conditions of (3.1) are

$$\begin{cases} (P+\eta I)\, x = -\frac{\mu}{2}\partial\|x\|_1, \\ x^T x = 1, \end{cases} \tag{3.3}$$

where $\partial\|x\|_1$ signifies the subgradient of $\|x\|_1$ and has the form

$$\partial\|x_i\|_1 = \begin{cases} \operatorname{sign}(x_i), & x_i \neq 0; \\ \delta \in [-1,1], & x_i = 0. \end{cases} \tag{3.4}$$

According to the definition of subgradient, the ℓ_1 norm is not differentiable at 0, and thus δ in (3.4) could be any real number between -1 and 1. If there exist a Lagrangian multiplier η and an x that satisfy (3.3), x is at least a local optimum of DSM (3.1). However, it is difficult to obtain the solution x and the corresponding η from (3.3), because $\partial\|x_i\|_1$ is unknown. The proposed DSA can sequentially determine zero variables in x and update the Lagrangian multiplier η in its path-following scheme.

3.2.2 Definitions

DSA finds sparse local solutions of DSM via dynamically selecting and updating critical and trivial variables in x. The final critical and trivial variables determine the nonzero and zero variables in the solution x^*, respectively. We define the importance and magnitude of a variable in x. They and KKT conditions together decide the updating rules for critical and trivial variables in DSA.

Definition 1. (Importance) *The importance of a variable x_i is defined as the absolute value of the partial derivative of $x^T Px + \eta\left(x^T x - 1\right)$ w.r.t. x_i. Thus the importance vector c of x is*

$$c = \left|(P+\eta I)\, x\right|. \tag{3.5}$$

According to the first equation in the KKT conditions (3.3), on a locally optimal solution, c can also be represented as

$$c = \left|\frac{\mu}{2}\partial\|x\|_1\right|. \tag{3.6}$$

The Lagrangian of DSM (3.2) can be decomposed as the sum of a loss function $x^T Px + \eta\left(x^T x - 1\right)$ and its ℓ_1 regularization $\mu\|x\|_1$. Hence the importance of a variable x_i measures the contribution of the variable x_i to the reduction of the loss function. The gradient g of the loss function, is also used in the subsequent derivations,

$$g = (P + \eta I)\, x = -\frac{\mu}{2}\partial\|x\|_1, \tag{3.7}$$

$$c = \operatorname{sign}(g) \cdot g, g = \operatorname{sign}(g) \cdot . \tag{3.8}$$

Definition 2. (Magnitude) *The magnitude of a variable x_i is its absolute value. Thus the magnitude vector m of x is*

$$m = |x| . \tag{3.9}$$

We use A and $B = A^C$ (the complement of A) to denote the index sets of critical and trivial variables in x, respectively.

Definition 3. (Critical variable) *A critical variable is a variable with larger importance than trivial ones and nonzero magnitude in the current iteration round:*

$$A = \{i : c_i \geq c_{j:j\in B}, m_i \neq 0\} . \tag{3.10}$$

Set A is dynamically updated throughout DSA. The final A^* in DSA is the nonzero set in the final solution x^*.

In DSA, A is initialized at the beginning of the algorithm. The importances of critical variables c_A are augmented and their corresponding magnitudes m_A are kept nonzero throughout DSA. A Critical variable will be transferred to B once its magnitude shrinks to zero.

Definition 4. (Trivial variable) *A trivial variable is a variable with smaller importance than critical ones in the current iteration round:*

$$B = \{i : c_i \leq c_{j:j\in A}\} . \tag{3.11}$$

Set B is dynamically updated throughout DSA. The final B^* in DSA is the zero set in the final solution x^*.

In DSA, B is initialized at the beginning of the algorithm. The magnitudes of trivial variables m_B are shrunk throughout DSA until reaching zeros. A trivial variable will be transferred to A once its importance reaches the minimum importance in c_A.

3.3 Double shrinking Algorithm

In this section, we develop DSA to optimize DSM. DSA is a path-following algorithm that finds the locally optimal solutions to a sequence of DSM (3.1) with the tradeoff parameter μ increasing from small to large. This μ controls the sparsity of the learned model. Each point on the solution path is a "warm start" for searching the subsequent sparser solution.

DSA starts from the initial solution x, the critical variable set A and the trivial variable set B. In each iteration round of DSA, the KKT conditions of the current and subsequent locally optimal solutions determine the directions of x_A and x_B. Afterward, the corresponding step size a is determined by three events. Occurrence of either event will violate the definition of A or B. Thus the optimization should be paused immediately and then both A and B will be modified accordingly. Finally, the locally optimal solution x, the ℓ_1 regularization weight μ and the Lagrangian multiplier η are updated. DSA stops when x reaches preferred sparsity.

3.3.1 Initialization

The solution x, the critical variable set A and the trivial variable set B are initialized at the beginning of DSA. The initial x must satisfy the KKT conditions (3.3) with a certain weight μ. The initial sets A^0 and B^0 are preferred to be close to the final nonzero and zero variable sets respectively, and thus many iteration rounds can be saved.

In this chapter, we select the initial solution x^0 as the dense solution of (3.1) by setting $\mu = 0$, i.e., the eigenvector of P associated with the smallest eigenvalue. Let the target sparse solution x^* has $s \leq p$ nonzero variables. The initial critical variable set A^0 is set as the variables with the first s largest importances in c^0,

and $B^0 = (A^0)^C$. There will be no difference to set A^0 as the variables with the first s largest magnitudes in m^0 and $B^0 = (A^0)^C$. That is because

$$c^0 = \left|Px^0\right|, m^0 = \left|x^0\right|, Px^0 = \lambda x^0, \tag{3.12}$$

where λ is the corresponding eigenvalue. Thus, we have

$$c^0 = \lambda m^0. \tag{3.13}$$

Thus the order of variable importances in c^0 is the same as the order of variable magnitudes in m^0. According to (3.10) and (3.3), the initial Lagrangian multiplier $\eta^0 = -\lambda$ and the initial tradeoff $\mu^0 = 0$.

3.3.2 Direction

In the k^{th} iteration round, DSA starts from the current solution x^k, proceeds along a direction ∇x with a particular step size a, and fetches a sparser locally optimal solution x^{k+1} satisfying the KKT conditions of DSM (3.3) with $\mu = \mu^{k+1}$. In this subsection, we apply the KKT conditions and the definitions of critical/trivial variables to obtain a state transformation equation that reveals how importances and magnitudes of variables in x change between x^k and x^{k+1}. Thus the equation leads to the computation of the direction.

According to (3.3), x^k satisfies the KKT conditions

$$\mathfrak{R}^k : \begin{cases} \left(P + \eta^k I\right) x^k = -\frac{\mu^k}{2} \partial \|x^k\|_1 \\ {x^k}^T x^k = 1. \end{cases} \tag{3.14}$$

Consider the subgradient of ℓ_1 norm given in (3.4), $\partial\|x^k\|_1$ will keep the same until arbitrary variables in x^k change their signs. However, the updating rule in DSA find the next solution on the path once a variable becoming zero and thus avoid the change of solution signs before the update of variable set. In particular, since events 1) and 2) in the step size computation (presented in Section 3.3) will pause the current iteration round once the magnitude of a variable is shrunk to zero, the signs of positive/negative variables in x^k will not be inverted in x^{k+1}

(note positive/negative variables are permitted to turn to zeros). According to (4), we have

$$\partial\|x^k\|_1 = \partial\|x^{k+1}\|_1. \tag{3.15}$$

However, it is worthy noting that after the update of variable sets, the zero variables are allowed to alter to positive/negative values in the next iterate, hence the sign vectors of the solutions on the path can be changed in DSA. In summary, the inverting of variable sign are not direct in DSA and needs a pause on a intermediate state, i.e., zero. And a sparse solution is obtained when reaching such intermediate state. Since x^k and x^{k+1} are sequel sparse solutions on their associated intermediate states, we can derive (3.15). Therefore,

$$\mathfrak{R}^{k+1} - \mathfrak{R}^k : \begin{cases} \left(P + \eta^k I\right) \Delta x \doteq -\Delta\eta x^k - \frac{\Delta\mu}{2}\partial\|x^k\|_1, \\ {x^k}^T \Delta x = 0, \end{cases} \tag{3.16}$$

where $\Delta x = x^{k+1} - x^k$, $\Delta\eta = \eta^{k+1} - \eta^k$ and $\Delta\mu = \mu^{k+1} - \mu^k$.

We ignore two small quantities of the second order $\Delta\eta\Delta x$ and $\Delta x^T\Delta x$ in calculating $\mathfrak{R}^{k+1} - \mathfrak{R}^k$. The first equation of (3.16) can be decomposed by A and B into

$$\begin{aligned} \begin{pmatrix} Q^k_{AA} & Q^k_{AB} \\ Q^k_{BA} & Q^k_{BB} \end{pmatrix} \begin{pmatrix} \Delta x_A \\ \Delta x_B \end{pmatrix} &= - \begin{pmatrix} \Delta\mu\partial\|x^k_A\|_1/2 \\ \Delta\mu\partial\|x^k_B\|_1/2 \end{pmatrix} - \Delta\eta \begin{pmatrix} x^k_A \\ x^k_B \end{pmatrix} \\ &= - \begin{pmatrix} \Delta g_A \\ \Delta g_B \end{pmatrix} - \Delta\eta \begin{pmatrix} x^k_A \\ x^k_B \end{pmatrix}, \end{aligned} \tag{3.17}$$

where $Q^k = P + \eta^k I$ and $\Delta g = g^{k+1} - g^k$. The last equivalence is due to (3.7). We compute the direction ∇x by solving Δx from (3.17).

The following theorem determines Δx_B and Δg_A appeared in (3.17).

Theorem 3. *If x^k and x^{k+1} are two consecutive solutions in DSA that satisfy the KKT conditions of DSM (3.3) with $\mu = \mu^k$ and $\mu = \mu^{k+1}$, respectively, the corresponding Δg_A and Δx_B are*

$$\Delta g_A = \Delta\mu/2 \cdot \operatorname{sign}\left(g^k_A\right) = \Delta c_A \cdot \operatorname{sign}\left(g^k_A\right), \tag{3.18}$$

$$\Delta x_B = -t \cdot \operatorname{sign}\left(x^k_B\right) = \Delta m_B \cdot \operatorname{sign}\left(x^k_B\right), \tag{3.19}$$

where $\Delta c = c^{k+1} - c^k$, $\Delta m = m^{k+1} - m^k$, *and* t *is a constant.*

Proof. When the event 2) in the step size criteria (see Section 3.3.3) happens, i.e., the magnitude of a critical variable is shrunk to zero, DSA will stop at the current iteration round and this variable with zero magnitude will be transited from A to B. This ensures that all the critical variables are nonzero in DSA. According to definitions of g in (3.7) and the subgradient of $\|x\|_1$ in (3.4), we have

$$g_A^k = \mu^k \partial \|x_A^k\|_1/2 = \mu^k/2 \cdot \operatorname{sign}\left(x_A^k\right), \tag{3.20}$$

$$\Delta g_A = \Delta\mu \partial \|x_A^k\|_1/2 = \Delta\mu/2 \cdot \operatorname{sign}\left(x_A^k\right). \tag{3.21}$$

The ℓ_1 regularization weight μ is increasing throughout DSA to pursue solutions from dense to sparse, so $\Delta\mu > 0$. By combining (3.8), (3.20) and (3.21), we obtain

$$\Delta c_A = \left|g_A^k + \Delta g_A\right| - \left|g_A^k\right| = \Delta\mu/2. \tag{3.22}$$

Thus the importances of critical variables in A are equally augmented in DSA. This completes the proof of (3.18).

In DSA, the ℓ_1 norm $\|x\|_1$ is decreased along the gradient decent direction of the trivial variables in B. According to (3.4), the negative partial derivative of $\|x\|_1$ w.r.t. a nonzero variable x_i is $-\operatorname{sign}(x_i)$. Thus Δx_B in DSA is

$$\Delta x_B = -t \cdot \operatorname{sign}\left(x_B^k\right). \tag{3.23}$$

Due to the definition of magnitude in (3.9), we have

$$\Delta m_B = -t. \tag{3.24}$$

This completes the proof of (3.19). □

Theorem 3 indicates that in each iteration round, the importances of critical variables in A are equally augmented, while magnitudes of trivial variables in B are equally shrunk. This is consistent with the definition of critical and trivial variables in Section 3.2.2.

In order to simplify the future derivation, we use a, a_1 and a_2 to represent μ, η and t, and thus we have:

$$\begin{cases} a = \Delta\mu/2, \\ a_1 a = t, \\ a_2 a = \Delta\eta. \end{cases} \tag{3.25}$$

By submitting (3.18), (3.19) and (3.25) into (3.17), we can obtain the state transformation equation

$$\begin{pmatrix} Q_{AA}^k & Q_{AB}^k \\ Q_{BA}^k & Q_{BB}^k \end{pmatrix} \begin{pmatrix} \Delta x_A \\ -a_1 a \cdot \text{sign}\left(x_B^k\right) \end{pmatrix} = - \begin{pmatrix} a \cdot \text{sign}\left(x_A^k\right) \\ \Delta g_B \end{pmatrix} - a_2 a \begin{pmatrix} x_A^k \\ x_B^k \end{pmatrix}. \tag{3.26}$$

Both Δx_A and Δg_B in (3.17) are determined by solving the state transformation equation (3.26).

We decompose the state transformation equation (3.26) into the following two equations:

$$Q_{AA}^k \Delta x_A - a_1 a Q_{AB}^k \text{sign}\left(x_B^k\right) = -a\text{sign}\left(x_A^k\right) - a_2 a x_A^k, \tag{3.27}$$

$$Q_{BA}^k \Delta x_A - a_1 a Q_{BB}^k \text{sign}\left(x_B^k\right) = -\Delta g_B - a_2 a x_B^k. \tag{3.28}$$

Both Δx_A and Δg_B can then be obtained by solving the above two equations.

We summarize the obtained Δx_A, Δg_A, Δx_B and Δg_B:

$$\Delta x_A = -a\left(Q_{AA}^k\right)^{-1}\left(\text{sign}\left(x_A^k\right) + a_2 x_A^k - a_1 Q_{AB}^k \text{sign}\left(x_B^k\right)\right), \tag{3.29}$$

$$\Delta g_A = a \cdot \text{sign}\left(x_A^k\right), \tag{3.30}$$

$$\Delta x_B = -a \cdot a_1 \text{sign}\left(x_B^k\right), \tag{3.31}$$

$$\Delta g_B = -a\left(Q_{BA}^k \Delta x_A + a_2 x_B^k - a_1 Q_{BB}^k \text{sign}\left(x_B^k\right)\right). \tag{3.32}$$

The above results reveal the changes of variable importances and magnitudes between two consecutively obtained solutions in DSA. However, there still exist three unknown constants a, a_1 and a_2 in (3.29)-(3.32).

This a is a multiplier shared by Δx_A, Δg_A, Δx_B and Δg_B, and thus we define

a as the step size of DSA and the corresponding direction vectors are

$$\Delta x_A = a\nabla x_A, \Delta g_A = a\nabla g_A \tag{3.33}$$

$$\Delta x_B = a\nabla x_B, \Delta g_B = a\nabla g_B, \tag{3.34}$$

where the direction vectors ∇x_A, ∇g_A, ∇x_B and ∇g_B are

$$\nabla x_A = -\left(Q_{AA}^k\right)^{-1}\left(\text{sign}\left(x_A^k\right) + a_2 x_A^k - a_1 Q_{AB}^k \text{sign}\left(x_B^k\right)\right), \tag{3.35}$$

$$\nabla g_A = \text{sign}\left(x_A^k\right), \tag{3.36}$$

$$\nabla x_B = -a_1 \text{sign}\left(x_B^k\right), \tag{3.37}$$

$$\nabla g_B = -\left(Q_{BA}^k \nabla x_A + a_2 x_B^k - a_1 Q_{BB}^k \text{sign}\left(x_B^k\right)\right). \tag{3.38}$$

The step size a is determined by the step size criteria in Section 3.3.3.

In order to calculate the directions in (3.35)-(3.38), it is necessary to further determine a_1 and a_2. This a_1 is the only predefined algorithm parameter. We derive a_2 to ensure the second equation in KKT conditions (3.3), i.e., the equality constraint $x^T x = 1$. Since the initial x^0 satisfies $\|x^0\|_2 = 1$, a_2 can be derived from the second equation ${x^k}^T \Delta x = 0$ in (3.16):

$${x^k}^T \Delta x = {x_A^k}^T \Delta x_A - a_1 a {x_B^k}^T \text{sign}\left(x_B^k\right) = {x_A^k}^T \Delta x_A - a_1 a \left\|x_B^k\right\|_1 = 0. \tag{3.39}$$

By substituting (3.29) into (3.39), we have

$${x_A^k}^T \left(Q_{AA}^k\right)^{-1}\left(\text{sign}\left(x_A^k\right) + a_2 x_A^k - a_1 Q_{AB}^k \text{sign}\left(x_B^k\right)\right) + a_1 \left\|x_B^k\right\|_1 = 0. \tag{3.40}$$

Thus a_2 is obtained by solving (3.40):

$$a_2 = \frac{a_1\left(\left\|x_B^k\right\|_1 + {x_A^k}^T \left(Q_{AA}^k\right)^{-1} Q_{AB}^k \text{sign}\left(x_B^k\right)\right)}{{x_A^k}^T \left(Q_{AA}^k\right)^{-1} x_A^k} - \frac{{x_A^k}^T \left(Q_{AA}^k\right)^{-1} \text{sign}\left(x_A^k\right)}{{x_A^k}^T \left(Q_{AA}^k\right)^{-1} x_A^k}. \tag{3.41}$$

Since a_2 defined in (3.41) is a necessary and sufficient condition of (3.39), the second equation $x^T x = 1$ in KKT conditions (3.3) and the second equation ${x^k}^T \Delta x = 0$ in (3,14) are both satisfied throughout DSA.

The direction ∇x is then obtained by substituting a_2 (3.41) into (3.35) and (3.37).

3.3.3 Step size and update of A, B

One appealing property of DSA is that the step size a of each iteration round can be adaptively determined by the following step size criteria without any expensive computations such as the line search method. The variable sets A and B are automatically updated in the computation of a.

According to the derivation of direction in Section 3.3.2 and the definition of critical/trivial variables in Section 3.2.2, in each iteration round, DSA proceeds along the directions (3.35)-(3.38) until one of the following two criteria are violated,

- the sign of each variable in x does not change;
- the importance of each trivial variable in B is less than the importance of any critical variable in A.

The violations of the above two criteria result in the following three events. Thus the iteration round will pause immediately when one of the following three events happens.

1. The magnitude of a trivial variable is shrunk to zero, i.e.,

$$x_i^{k+1} = x_i^k + a\nabla x_i = 0, i \in B. \tag{3.42}$$

   Thus the minimum step size that makes event 1) happen is

$$a(1) = \min_{i\in B}^{+} -\frac{x_i^k}{\nabla x_i}, \tag{3.43}$$

   where $a(1)$ is positive, and ∇x_i is defined in (3.37).

2. The magnitude of a critical variable is shrunk to zero, i.e.,

$$x_i^{k+1} = x_i^k + a\nabla x_i = 0, i \in A. \tag{3.44}$$

Thus the minimum step size that makes event 2) happen is

$$a(2) = \min_{i \in A}^{+} -\frac{x_i^k}{\nabla x_i}, \tag{3.45}$$

where $a(2)$ is positive, and ∇x_i is defined in (3.35). Since the magnitude of each critical variable in A is nonzero throughout DSA, the critical variable x_i will be transferred to the trivial variable set B once the event 2) happens and it is directly shrunk to zero.

3. The importance of a nonzero trivial variable reaches the minimum importance of the critical ones, i.e.,

$$c_j^{k+1} = \min_{i \in A} c_i^{k+1} = \min_{i \in A} c_i^k + a, j \in B. \tag{3.46}$$

Since the trivial variable x_j^{k+1} is nonzero, according to the definitions of the gradient g in (3.7) and the subgradient of $\|x\|_1$ in (3.4), we have

$$g_j^{k+1} = \text{sign}\left(x_j^{k+1}\right) \cdot c_j^{k+1}. \tag{3.47}$$

By substituting (3.46) into (3.47), since $g_j^{k+1} = g_j^k + a\nabla g_j$, we have

$$\text{sign}\left(x_j^{k+1}\right) \cdot \left(\min_{i \in A} c_i^k + a\right) = g_j^k + a\nabla g_j. \tag{3.48}$$

Thus the minimum step size that makes event 3) happen is

$$a(3) = \min_{j \in B}^{+} \frac{\text{sign}\left(x_j^{k+1}\right) \cdot \min_{i \in A} c_i^k - g_j^k}{\nabla g_j - \text{sign}\left(x_j^{k+1}\right)}, \tag{3.49}$$

where $a(3)$ is positive, and ∇g_j is defined in (3.38). Since the importance of each trivial variable in B is less than the minimum importance of the critical variables in A, the trivial variable x_j will be transferred to the critical variable set A once the event 3) happens.

Therefore the step size a is determined by

$$a = \min\left\{a(1), a(2), a(3)\right\}. \tag{3.50}$$

3.3.4 Update of x, μ and η

Given the directions ∇x_A, ∇x_B and the step size a, the solution x^{k+1} is

$$\begin{cases} x_A^{k+1} = x_A^k + a\nabla x_A \\ x_B^{k+1} = x_B^k + a\nabla x_B, \end{cases} \tag{3.51}$$

where the step size a is obtained from (3.43)(3.45)(3.49)(3.50), directions ∇x_A and ∇x_B are calculated according to (3.35) and (3.37), respectively.

At the end of each iteration round, η is updated for the computation of $Q = (P + \eta I)$ of the next iteration round. According to the KKT conditions (3.3), we have

$$(P + \eta I)\, x = -\frac{\mu}{2}\partial\|x\|_1. \tag{3.52}$$

By multiplying x^T to both sides of (3.52) simultaneously, we have

$$x^T (P + \eta I)\, x = x^T P x + \eta = -\frac{\mu}{2}\|x\|_1. \tag{3.53}$$

Thus η is updated according to

$$\eta^{k+1} = -x^{k+1^T} P x^{k+1} - \frac{\mu^{k+1}}{2}\left\|x^{k+1}\right\|_1, \tag{3.54}$$

where $\mu^{k+1}/2$ is updated according to (3.25):

$$\frac{\mu^{k+1}}{2} = \frac{\mu^k}{2} + \frac{\Delta\mu}{2} = \frac{\mu^k}{2} + a. \tag{3.55}$$

3.3.5 Algorithm

DSA builds a solution path for DSM from the dense solution to sparse ones. The stopping criterion of DSA is defined as:

$$\text{cardinality}(x) \leq s, \tag{3.56}$$

where s is the target cardinality of the final sparse solution x^*.

Algorithm 2: Double shrinking Algorithm (DSA)

Input: P, s, a_1, x^0, A^0, B^0

Output: The solution x^*

Initialize $x := x^0$, $A := A^0$, $B := B^0$, $\eta := -\lambda$, $\mu := 0$, $k := 1$;

while cardinality$(x^k) > s$ **do**

Calculate the direction ∇x via calculating ∇x_B, ∇g_A, a_2, ∇x_A and ∇g_B by (3.37), (3.36), (3.41), (3.35) and (3.38), respectively;

Calculate the step size a by (3.43), (3.45), (3.49) and (3.50). Update A and B when event 2) or 3) happens;

Update $x^{k+1} = x^k + a \cdot \nabla x$, η^{k+1} by (3.50) and $Q^{k+1} = \left(P + \eta^{k+1} I\right)$;

k:=k+1;

end

return $x^* = x^k$;

Algorithm 2 shows how to obtain a sparse solution x of DSM by DSA. However, in practice, a sparse projection matrix or a sparse representation of more than one dimensionality is preferred. A sparse matrix $X = [X_1; X_2; \cdots; X_d]$ for DSM can be obtained by utilizing a sequence of DSAs. Before the $(i+1)^{th}$ DSA, we update P as the residual $P := P - (X_i^T P X_i) \cdot X_i X_i^T$. The initial solution x^0 for the $(i+1)^{th}$ DSA algorithm can be computed as the first eigenvector of the updated P. Algorithm 3 shows how to use DSA to obtain d sparse vectors. The update of P in sparse PCA has been broadly studied in previous literatures [169] as "deflation methods". We adopts the most commonly used deflation method in Algorithm 3. Please refer to [169] for the detailed introduction of other deflation methods.

Algorithm 3: DSA for d sparse vectors

Input: P, s, a_1

Output: The sparse matrix X

for $i \leftarrow 1$ **to** d **do**

$x := x^0$, the first eigenvector of P;

Conduct DSA, and obtain the solution x^*;

Update $P := P - (x^{*T} P x^*) \cdot x^* x^{*T}$;

end

3.3.6 Analyses and Proofs

We theoretically analyze some crucial properties of DSA. The corresponding proofs suggest that DSA converges to at least a local minimum of DSM (3.3) with preferred sparsity in a small number of iteration rounds[1]. The time complexity of each iteration round in DSA is not more than $\mathcal{O}({s_A}^3 + {s_B}^2)$ and is possible to be further reduced. We also analyze the influence of the unique parameter a_1 to the convergence of DSA.

DSA is derived from the KKT conditions of DSM, and thus the solution is at least a local optimum of the optimization (3.1). Theorem 4 proves that DSA converges to an approximate local optimum.

Theorem 4. (**Convergence**) *In DSA, the solution of its k^{th} iteration round x^k satisfies the KKT conditions (3.3) of the DSM (3.1) with an ℓ_1 penalty weight μ^k.*

Proof. The solution of the first iteration round is the dense solution x^0, which is an eigenvalue of P, and thus we have

$$\begin{cases} Px^0 = \lambda x^0, \\ {x^0}^T x^0 = 1. \end{cases} \tag{3.57}$$

Equation (3.58) can be written as

$$\begin{cases} (P - \lambda)\, x^0 = -\frac{0}{2}\partial \|x\|_1, \\ {x^0}^T x^0 = 1. \end{cases} \tag{3.58}$$

Refer to (3.3), the initial x^0 satisfies the KKT conditions of DSM with an ℓ_1 penalty weight $\mu^k = 0$. Without loss of generality, the solution of the first iteration round can be set as any x that satisfies the KKT conditions of DSM (3.1) with an arbitrary penalty weight μ^k.

Assume the solution of the k^{th} iteration round x^k satisfies the KKT conditions

[1]The number of iteration rounds is the number of occurrences of the three events in Section 3.3.3. The first two events happens $p - s$ times, and the occurrences of event 3) are much less than that of the first two events.

of DSM (3.1) with an ℓ_1 regularization weight μ^k, i.e.,

$$\begin{cases} \left(P+\eta^k I\right) x^k = -\frac{\mu^k}{2}\partial\|x^k\|_1, \\ {x^k}^T x^k = 1, \end{cases} \tag{3.59}$$

We initially consider the first equation in the KKT conditions (3.3). The state transform equation (3.26) is satisfied when Δx_A and Δg_B are obtained according to (3.29) and (3.32), respectively. Equation (3.26) derives (3.17). Since (3.17) is an equal decomposition of (3.16), (3.16) is automatically satisfied. Since the step size a ensures that the sign vector of x keeps unchanged inside each iteration round (may change between two consecutive iteration rounds), we have $\partial\|x^k\|_1 = \partial\|x^{k+1}\|_1$. Thus the first equation in (3.16) can be rewritten as

$$\begin{aligned} &\left(P+\eta^k I\right)\left(x^{k+1}-x^k\right) \\ =& -\Delta\eta x^k - \frac{\mu^{k+1}}{2}\partial\|x^{k+1}\|_1 + \frac{\mu^k}{2}\partial\|x^k\|_1 \\ \doteq& -\Delta\eta\left(x^k+\Delta x\right) - \frac{\mu^{k+1}}{2}\partial\|x^{k+1}\|_1 + \frac{\mu^k}{2}\partial\|x^k\|_1 \\ =& -\Delta\eta x^{k+1} - \frac{\mu^{k+1}}{2}\partial\|x^{k+1}\|_1 + \frac{\mu^k}{2}\partial\|x^k\|_1. \end{aligned} \tag{3.60}$$

By combining the first equation of (3.59) and (3.60), we arrive at

$$\left(P+\eta^{k+1} I\right) x^{k+1} = -\frac{\mu^{k+1}}{2}\partial\|x^{k+1}\|_1. \tag{3.61}$$

Thus the first equation in the KKT conditions (3.3) is satisfied.

We then consider the second equation in the KKT conditions (3.3). By substituting a_2 defined in (3.41) into Δx_A in (3.29), we have

$${x^k}^T \Delta x = {x_A^k}^T \Delta x_A + {x_B^k}^T \Delta x_B = 0. \tag{3.62}$$

By combining the first equation of (3.59) and (3.62), we arrive at

$${x^{k+1}}^T x^{k+1} = 1. \tag{3.63}$$

Thus (3.61) and (3.63) compose the KKT conditions of DSM (3.1) with an ℓ_1

regularization weight μ^{k+1}, and they are satisfied when $x = x^{k+1}$. According to the above analyses, we conclude that the solution of the k^{th} iteration x^k approximately satisfies the KKT conditions (3.3) of DSM with an ℓ_1 penalty weight μ^k in DSA. This completes the proof. □

In DSA, the only predefined algorithm parameter is a_1. Inappropriate selection of a_1 increases the number of iteration rounds for the rectification of the critical and trivial variables. However, different choice of a_1 will not influence the convergence to local optimums because Theorem 4 is independent of a_1.

The time complexity of each iteration round in DSA is determined by the cardinality of the critical variable set A in this iteration round. Theorem 5 shows the time complexity of DSA.

Theorem 5. (**Complexity**) *The time complexity of each iteration round of DSA is not more than* $\mathcal{O}\left(s_A^3 + s_B^2\right)$*, wherein* s_A *and* s_B *are the cardinalities of* A *and* B *respectively in this iteration round.*

Proof. In each iteration round of DSA, the directions ∇x_B, ∇g_A, a_2, ∇x_A and ∇g_B are calculated according to (3.37), (3.36), (3.41), (3.35) and (3.38), respectively. The step size a is calculated by using (3.43), (3.45), (3.49) and (3.50). Both x^{k+1} and η^{k+1} are updated by using (3.51) and (3.54), respectively. The main computational costs of these operations are the matrix inverse calculation $\left(Q_{AA}^k\right)^{-1}$ in (3.35) with complexity $\mathcal{O}\left(s_A^3\right)$ and the matrix multiplication $Q_{BB}^k \text{sign}\left(x_B^k\right)$ in (3.38) with complexity $\mathcal{O}\left(s_B^2\right)$. Thus, the time complexity of each iteration round is $\mathcal{O}\left(s_A^3 + s_B^2\right)$, wherein s_A and s_B are the cardinalities of A and B respectively in this iteration round. This completes the proof. □

It is possible to further reduce the time complexity of each iteration round by accelerating the matrix inverse computation. In particular, $\left(Q_{AA}^{k+1}\right)^{-1}$ can be updated from $\left(Q_{AA}^k\right)^{-1}$ approximately. If $\Delta\eta$ is small compared with Q_{AA}^k and I, according to [186], we have

$$\begin{aligned} \left(Q_{AA}^{k+1}\right)^{-1} &= \left(Q_{AA}^k + \Delta\eta I\right)^{-1} \\ &\cong \left(Q_{AA}^k\right)^{-1} - \Delta\eta \left(Q_{AA}^k\right)^{-1} \left(Q_{AA}^k\right)^{-1}. \end{aligned} \tag{3.64}$$

If A is not updated at the end of the k^{th} iteration round, then $\left(Q_{AA}^{k+1}\right)^{-1}$ can be updated from $\left(Q_{AA}^{k}\right)^{-1}$ according to (3.64). If one variable x_i is removed from A at the end of the k^{th} iteration, assume the updated A is A^*, it is still possible to update $\left(Q_{A^*A^*}^{k}\right)^{-1}$ from $\left(Q_{AA}^{k}\right)^{-1}$ by using the block matrix inverse [99]. The above analyses show preliminary results on the acceleration.

3.4 Extensions of double shrinkage

It has proved that the ℓ_p norms with $0 \leq p < 2$ can be employed as penalties to obtain sparse solutions. However, the sparse solution will own different properties when different p value is selected. In compressed sensing [41], the ℓ_1 norm is minimized as a convex relaxation of cardinality or the ℓ_0 norm minimization for exact recovery of sparse signal. In elastic net [275], the weighted sum of the ℓ_1 norm and the ℓ_2 norm is minimized, this elastic net penalty has a grouping effect to the strongly correlated variables. In adaptive *lasso* [274] and reweighted ℓ_1 minimization [42], the $\ell_p(0 \leq p < 1)$ norm or its approximation is minimized to obtain more sparse solutions, which enjoy consistency and better performance than the ℓ_1 norm when signal is extremely sparse or substantially fewer measurements are available. In group *lasso* [243] and structured sparsity [131][124], a mixed $\ell_1 - \ell_2$ norm is minimized, because it can encode the structural information of variables and this information can improve the accuracy and decrease the time cost. Double shrinkage can be extended to these sparse penalties. At the end of this section, we will discuss the possibilities to extend double shrinkage to sparse learning problems with equality constraints.

3.4.1 Elastic net double shrinkage

Elastic Net penalty [275] is a weighted sum of the ℓ_1 norm and the ℓ_2 norm, i.e.,

$$\mu_1 \|x\|_1 + \mu_2 \|x\|_2^2. \tag{3.65}$$

Compared with the ℓ_1 norm, elastic net penalty has special priorities in the following three scenarios: 1) when the data dimensionality p is much higher

than the sample number n, the ℓ_1 norm minimization can only select n nonzero variables at most. But elastic net penalty can select more variables and produce more stable solution in this case; 2) elastic net penalty can group the strongly correlated variables together in the final solution, while the ℓ_1 norm can only select one variable in each group; and 3) if $n > p$ and there exist highly correlated variables, elastic net penalty will produce better prediction than the ℓ_1 norm.

In DSM, if the elastic net penalty is adopted, the problem (3.1) can be written as

$$\min_x \ x^T P x + \mu_1 \|x\|_1 + \mu_2 \|x\|_2^2 \ \ s.t.\ x^T x = 1. \tag{3.66}$$

This problem can be rewritten as the ℓ_1 norm penalized optimization:

$$\min_x \ x^T \left(P + \mu_2 I\right) x + \mu_1 \|x\|_1 \ \ s.t.\ x^T x = 1. \tag{3.67}$$

Thus the elastic net penalized DSM (3.66) can be optimized via DSA algorithm by solving (3.67). According to [275], since x shrinks twice in (3.66), i.e., the ridge regularization and the ℓ_1 regularization, the final solution should be rescaled by using

$$x^* \leftarrow \sqrt{1 + \mu_2} x^*. \tag{3.68}$$

3.4.2 Reweighted ℓ_1 double shrinkage

In double shrinkage, when the solution x is required to be very sparse or the number of given samples are substantially insufficient, the $\ell_p (0 \leq p < 1)$ norm is a better choice than the ℓ_1 norm because it can produce more sparse or more robust solutions. However, the $\ell_p (0 \leq p < 1)$ norm is not convex and thus lots of optimization algorithms are invalid. In [42] and [274], the reweighted ℓ_1 minimization is used to achieve the similar effects of the $\ell_p (0 \leq p < 1)$ penalties. Reweighted ℓ_1 minimization iteratively solves a sequence of the weighted ℓ_1 minimization problems, wherein the ℓ_1 norm applied to the next iteration round is weighted by the inverse of the magnitude of the solution obtained in the current iteration round. We can apply this reweighted ℓ_1 minimization to DSA. In its

$(k+1)^{th}$ iteration round, the following ℓ_1 minimization problem is solved:

$$\min_x \ x^T P x + \mu \sum_{i=1}^{q} \frac{|x_i|}{|x_i^k|} \quad s.t.\ x^T x = 1, \tag{3.69}$$

where x^k is the solution obtained in the k^{th} iteration and has dimension of q. The problem (3.69) can be written as

$$\begin{aligned} \min_x \ & x^T Q x + \mu \|x\|_1 \quad s.t.\ x^T x = 1, \\ & Q = \text{diag}\left(\left|x^k\right|\right) P \text{diag}\left(\left|x^k\right|\right). \end{aligned}$$

This problem can be handled by DSA. Let the solution to be $\hat{x}^{k+1}$, and the solution of the subsequent $(k+1)^{th}$ iteration round is

$$x^{k+1} = \left|x^k\right| \cdot \hat{x}^{k+1}. \tag{3.70}$$

In summary, DSA can solve the $\ell_p (0 \leq p < 1)$ norm penalized double shrinkage.

3.4.3 Structured double shrinkage

Recent studies in structured sparsity [243][131][124][256] have proved that when the structural information of the variables in x is known, the time cost of the sparse solution pursuit can be decreased and the corresponding accuracy can be improved. In this case, variables are separated into different groups, and variables in the same group tend to be zero or nonzero simultaneously. In structured sparsity, a mixed $\ell_1 - \ell_2$ norm is chosen as the penalty to encode the structural information into the optimization problem. Assume that variables in x are separated into k groups, i.e., $x = \left[x^1, x^2, \cdots, x^k\right]^T$, wherein $x^i = \left[x_1^i, x_1^i, \cdots, x_{n_i}^i\right]$ is the i^{th} group. The structured sparsity penalty is

$$\|x\|_s = \sum_{i=1}^{k} \sqrt{\|x^i\|_2^2}. \tag{3.71}$$

In double shrinkage, structured sparsity can encode the discriminative information and the structural information of variables into the optimization to improve

the classification performance and accelerate the speed of variable selection, respectively. When x is the projection vector, the group can be defined as the strongly correlated features in the original data, and thus variables in the same group will be simultaneously selected if their importances are sufficiently large. When x is the low dimensional representation, the group can be defined as the samples in the same class, and thus the sparse low dimensional representations of the samples in the same group will approach to zero or nonzero simultaneously.

3.4.4 Sparse learning with multiple equality constraints

Because of the advantages of sparse learning in improving interpretability and computational efficiency, the ℓ_1 regularization has been widely introduced to many classical machine learning models. However, most of the current ℓ_1 optimization algorithms cannot deal with sparse learning problems with equality constraints. Double shrinkage provides a novel scheme to tackle this kind of problems. Assume the optimization problem has the following form:

$$\begin{aligned} \min_{x} \quad & F(x) + \mu\|x\|_1 \\ s.t. \quad & H_i(x) = 0, i = 1, \cdots, n. \end{aligned} \tag{3.72}$$

The corresponding KKT conditions are:

$$\begin{cases} \partial F(x) + \sum_{i=1}^{n} \eta_i \partial H_i(x) = -\mu \partial \|x\|_1, \\ H_i(x) = 0, i = 1, \cdots, n. \end{cases} \tag{3.73}$$

wherein η_i is the Lagrangian multiplier corresponding to the constraint $H_i(x) = 0$. Let the solution of the k^{th} iteration round be x^k, we have:

$$\begin{cases} \partial F(x^{k+1}) - \partial F(x^k) + \sum_{i=1}^{n} \eta_i^{k+1} \partial H_i(x^{k+1}) - \sum_{i=1}^{n} \eta_i^k \partial H_i(x^k) = -\Delta\mu \partial \|x^k\|_1, \\ H_i(x^{k+1}) - H_i(x^k) = 0, i = 1, \cdots, n. \end{cases} \tag{3.74}$$

By replacing F and H_i with their second order Taylor expansions, respectively, the first equation in (3.81) can be written as

$$\begin{aligned} &\begin{pmatrix} Q^k_{AA}(\eta^{k+1}) & Q^k_{AB}(\eta^{k+1}) \\ Q^k_{BA}(\eta^{k+1}) & Q^k_{BB}(\eta^{k+1}) \end{pmatrix} \begin{pmatrix} \Delta x_A \\ -a_1 a \cdot \text{sign}\left(x^k_B\right) \end{pmatrix} \\ &= -\begin{pmatrix} a \cdot \text{sign}\left(x^k_A\right) \\ \Delta g_B \end{pmatrix} - a \begin{pmatrix} h_A(\eta^{k+1}) \\ h_B(\eta^{k+1}) \end{pmatrix}, \end{aligned} \tag{3.75}$$

where Q^k is a $p \times p$ matrix and $h(\eta^{k+1})$ is a $p \times 1$ vector computed from η^{k+1}. The η^{k+1} in (3.82) includes n variables and is calculated by solving n equations $H_i(x^{k+1}) - H_i(x^k) = 0, i = 1, \cdots, n$.

In each iteration round, the direction Δx is computed by building the state transformation equation from (3.82) and solve the equation by using the similar method proposed in Section 3.3.2. Afterward, the step size a is obtained according to the three events defined in Section 3.3.3. Thus a solution path of the sparse learning problem (3.79) can be achieved via an iterative algorithm similar to DSA.

Sparse learning problem with inequality constraints can be easily solved by using the gradient projection method. In each iteration round, the solution is optimized on the gradient decent direction and then projected onto the feasible set determined by the inequality constraints. They are not the main focus of double shrinkage and we do not put details on due to the limited page length.

3.5 Relationships to existing techniques

Although double shrinkage is intrinsically different from sparse PCA, sparse coding, and LARS in terms of problem formulation and concrete algorithm, it is helpful to discuss their relationships for better understanding the proposed algorithm. In this section, we first analyze the links and differences between double shrinkage and sparse PCA, sparse coding, and LARS, respectively. We then show their connections in motivations and similarities in objective functions given additional constraints.

3.5.1 Relationship to sparse PCA

Sparse PCA aims at finding sparse PCs which simultaneously have simpler interpretation and less but comparable explained variance comparing with PCA. Several formulations and algorithms have been proposed for sparse PCA. Zou et al. [273] formulated sparse PCA as a regression-type problem and solved it by alternating optimization, where least angle regression (LARS) [77] for elastic net is utilized in each iteration round. Direct sparse PCA (DSPCA) [60] relaxes the original sparse PCA to a semidefinite programming (SDP) problem and solves it via standard SDP solver. Path sparse PCA [59] is a fast greedy search algorithm solving sparse PCA by relaxing the equality constraint $x^T x = 1$ with $x^T x \leq 1$. The sPCA-rSVD [201] formulates sparse PCA as regularized singular value decomposition (SVD) and solves it via combining linear regression and thresholding. Penalized matrix decomposition (PMD) [229] formulates sparse PCA as a matrix decomposition problem restricting upper bounds of the ℓ_1 and ℓ_2 norms of the two decomposition components. PMD is solved by sparse principal components (SPC) proposed in [229], which alternatively soft-thresholding based power iterations of the two components. The online sparse coding method proposed in [173] can also be applied to sparse PCA after modification. It adds the "elastic net" constraints to the dictionary D in the dictionary learning problem and performs alternative online updates of dictionary and representation. The rows of the obtained sparse D can be used as sparse PCs. Greedy SPCA [176] uses a combinatorial greedy method to determine the nonzero entries in sparse PC.

Double shrinkage shares the same motivation as sparse PCA, i.e., finding sparse alternatives of standard eigenvectors that remains the largest explained variance. In addition, DSM shares the same objective function with many existing sparse PCA methods. In applications, the results of both sparse PCA and double shrinkage can obtain sparse dimension reduction of the given data.

However, double shrinkage is intrinsically different from existing sparse PCA methods in the following two aspects:

- Double shrinkage and existing sparse PCA methods are different in optimization problem formulations. DSM only adds an ℓ_1 regularization on the original eigenvalue maximization for standard PCA. Existing sparse

PCA methods can be mainly interpreted as optimizations of two relaxations to DSM: 1) sparsity constrained/regularized regression-type problem with constraint $x^T x \leq 1$ or normalization of the obtained x, e.g., Sparse PCA, sPCA-rSVD and online sparse coding; 2) sparsity constrained/regularized SDP that maximizes the explained variance of sparse PC with constraint $x^T x \leq 1$, e.g., DSPCA, Path SPCA and SPC. One significant contribution of double shrinkage is that the equality constraint $x^T x = 1$ is not relaxed in the ℓ_1 regularized optimization.

- Double shrinkage and existing sparse PCA methods are different in optimization algorithms. DSA is a path-following algorithm for DSM and approximately ensures the local optimality of the solutions on the path. Most existing sparse PCA methods adopt special formulations of sparse PCA and develop algorithms based on existing ones in compressed sensing and sparse coding. Path SPCA [59] is a greedy algorithm solving one kind of relaxation of the cardinality penalized PCA with the ℓ_2 ball constraint. It builds the solution path by greedily selecting the nonzero entry that maximizes the increasing of variance. Such greedy path building method is different from the homotopy-type path-following scheme adopted in DSA, because DSA finds the sequential sparse solution by updating the KKT conditions required on the solution. This difference is analogous to the difference between forward selection and LARS in solving *lasso*. Another significant contribution of the proposed double shrinkage is that it optimizes a new sort of sparsity regularized optimization model with an equality constraint.

We provide a theoretical comparison of the time complexity of DSA with the existing sparse PCA algorithms in Table 3.5.1. The numerical comparisons are available in Section 3.6.5.

3.5.2 Relationship to sparse coding

Sparse coding (or equivalently dictionary learning) [142][1][150][173] builds a dictionary $D \in \mathbb{R}^{d \times p}$ and its corresponding sparse representation $A \in \mathbb{R}^{n \times d}$ for obtaining a sparse approximation AD of the given data $X \in \mathbb{R}^{n \times p}$. As a trade off

Table 3.1: Time complexity per iteration round of Sparse PCA, DSPCA, rSVD, SPC, Greedy SPCA and Double Shrinkage for calculating one sparse vector solution with s nonzero entries. We have $s_A, s_B \leq s \leq \min\{n, p\}$.

Method	Time Complexity
Sparse PCA	$\mathcal{O}\left(s^4\right)$
DSPCA	$\mathcal{O}\left(p^3\right)$
rSVD	$\mathcal{O}\left(np\right)$
SPC	$\mathcal{O}\left(np\right)$
Greedy SPCA	$\mathcal{O}\left(p^3\right)$
Double Shrinkage	$\mathcal{O}\left(s_A^3 + s_B^2\right)$

between dimensionality and sparsity with guaranteed approximation precision, D is usually an overdetermined dictionary with $d > p$. The problem of sparse coding has been formulated as an alternating unconstrained optimization:

$$\min_{A,D} \|X - AD\|_F^2 + \mu \|A\|_1 . \tag{3.76}$$

The objective function is composed of the least square error of the approximation and the ℓ_1 regularization of the sparse representation A. In some variants, one of these two parts is transformed into a constraint, and thus the above optimization turns to an ℓ_1 constrained least square regression or an ℓ_1 minimization with the constraint $X = AD$. Most of the existing sparse coding methods iteratively optimize A with the fixed D (sparse coding stage) and then optimize D with the fixed A (dictionary update stage). They differ in the concrete algorithms of the two stages. The first stage solves an ℓ_1 regularized least square regression and the second stage is a least square regression.

The objective function of sparse coding (3.76) is equivalent to that of DSM (3.1) if an extra constraint $A^T A = I$ is added to (3.76). To see this, we unfold the least square error in (3.76):

$$\begin{aligned} \|X - AD\|_F^2 &= \operatorname{tr}\left(X^T X - 2X^T AD + D^T A^T AD\right) \\ &= \operatorname{tr}\left(D^T D - 2X^T AD\right) + \operatorname{tr}\left(\mathrm{X^T X}\right). \end{aligned} \tag{3.77}$$

The second equivalence is due to the additional orthogonal constraint of A. Since X is a constant matrix, the last term $\mathrm{tr}\left(\mathrm{X^TX}\right)$ can be dropped in the optimization. We set the gradient of the objective function (3.77) w.r.t. D to zero and obtain $D = A^T X$. A new optimization problem is obtained by substituting $D = A^T X$ into (3.77):

$$\max_{A^T A = I} \mathrm{tr}\left(A^T X X^T A\right) + \mu \left\|A\right\|_1 . \tag{3.78}$$

When A includes only one column, the above optimization is equal to DSM (3.1) with $P = XX^T$. According to this connection, the sparse representations in dictionary learning can be deemed as sparse dimension reduction of the given data.

However, sparse coding is different from double shrinkage in both problem formulation and algorithm scheme. In particular,

- Sparse coding is an unconstrained optimization of two variables, while double shrinkage is a constrained optimization of a single variable. The objective function of sparse coding can be equally transformed to that of double shrinkage if its solution A fulfills $A^T A = I$. However, neither original sparse coding nor double shrinkage (in d sparse vectors situation) ensures the orthogonality of the solution matrix[1]. Thus they address different optimization problems and have different solutions.

- Sparse coding approximates the given data matrix X with its sparse representations on a dictionary, i.e., AD. To ensure an accurate approximation, an overdetermined dictionary and high dimensional ($d > p$) representations are required. In contrast, double shrinkage reduces the dimensionality of the given data X with maximum explained variance and sparse representations/projection matrix. Moreover, the matrix P in double shrinkage can be build by different dimension reduction methods besides $P = XX^T$ in (3.78) for different utilizations.

- Sparse coding algorithms adopt alternating optimization that sequentially

[1] Although the solution matrix is possible to be orthogonal in double shrinkage, the probability of this situation is extremely small in practice and thus can be ignored.

solves two unconstrained optimizations, while DSA is a path-following algorithm for an ℓ_1 regularized optimization with an equality constraint.

3.5.3 Relationship to LARS

LARS is a path-following algorithm [185] for the ℓ_1 regularized least square regression, which has been broadly used in compressed sensing and model selection:

$$\min_x \|y - Ax\|_2^2 + \mu\|x\|_1. \tag{3.79}$$

LARS starts from the origin $x = \mathbf{0}$ and sequentially adds the zero variable x_i with the largest correlation (the absolute value of gradient) into the active set (nonzero set) in each iteration round. According to the KKT conditions of (3.79), i.e., the subgradient of objective function is equal to $\mathbf{0}$, LARS optimizes the variables within the active set in each iteration round via selecting a direction that equally decreases their correlations. The active variables proceed along the direction until a zero variable outside the active set reaches their correlation. This is due to the subgradient of the ℓ_1 norm in (4). Then the iteration round terminates and the next one begins by adding a new zero variable into the active set. LARS builds the solution path of (3.79) from $\mathbf{0}$ to dense. Each solution on the path satisfies the KKT conditions with the corresponding μ.

DSA is a path-following algorithm and similar to LARS. In particular, DSA measures absolute value of gradient to discriminate whether a variable is critical or not, DSA determines the direction of x in each iteration round by ensuring the KKT conditions of DSM, and the corresponding step size is determined by the subgradient of the ℓ_1 norm. DSA inherits the advantages of LARS, i.e., the local optimality of solutions on the path and the simultaneous update of critical/trivial variable set (active set) and the coefficients of active variables.

However, DSA is intrinsically different from LARS. In particular,

- DSA optimizes an equality constrained optimization model rather than an unconstrained one in LARS, and thus it has two KKT conditions to guarantee rather than one in LARS on the solution path. The Lagrangian multiplier η is also updated in DSA by updating of a_2 according to the equality constraint $x^T x = 1$.

- DSA simultaneously shrinks the magnitudes of trivial variables and increases the importances of critical variables along a direction of both critical and trivial variables, while LARS only shrinks the correlation (similar to importance in DSA) of critical variables and keeps magnitudes of the trivial variables zero in the optimization.

- DSA has three stopping criteria for determining of the step size in each iteration round, while LARS has two.

- DSA can start from the dense solution or arbitrary point on the solution path, while LARS can only start from the origin $\mathbf{0}$ (or backward LARS can only start from the optimal dense solution of the least squares).

3.6 Experiments

In this section, we show double shrinkage can benefit several machine learning tasks including classification, nonlinear manifold learning, clustering and feature selection. Experiments are conducted on different kinds of datasets, such as face datasets [188][14][11][197][210], COIL-20 object dataset [180], UCI machine learning repository [10], Pitprops data [212] and gene expression data [5][4]. Double shrinkage is applied to classification and feature selection experiments for producing sparse projection matrix, while it is employed to derive sparse low dimensional representations in nonlinear manifold learning and clustering experiments. We show that the obtained sparse solutions perform competitively compared against the corresponding dense solutions. We also compare double shrinkage against existing sparse PCA solvers [273][176][59][201][229] on Pitprops data, gene expression data and artificial data. The experimental results demonstrate that double shrinkage is able to produce better solution with less time cost. We implement all the algorithms in MatLab and run all the experiments on a 3.0GHz Intel Xeon processor with 32GB main memory under Windows XP. In DSA, the only free parameter $a_1 = 0.4$ is fixed in all experiments.

3.6.1 Classification

Double shrinkage can obtain proper representations of data samples and thus benefit the subsequent classification. We evaluate double shrinkage by testing the classification performance of the sparse projection matrix obtained by DSA on 4 human face datasets, including FERET [188], UMIST [11], YALE [14], and ORL [197], and 2 handwritten digit datasets, including MNIST [147] and USPS [109].

The FERET dataset consists of $13,539$ face images from $1,565$ individuals. The images vary in size, gender, pose, illumination, facial expression and age. In our experiment, we randomly select 100 individuals, each of which has 7 images, 5 of which are randomly chosen for training and the rest for test. There are 565 face images from 20 individuals in the UMIST dataset. The samples change in race, gender, pose and appearance. We randomly choose 7 images of each individual for training and the rest for test. The YALE dataset contains 165 face images of 15 individuals. Lighting conditions, gender, facial expressions and configurations are different among these images. The ORL dataset includes 400 face images from 10 individuals. The images were taken at different times, varying the lighting, facial expressions and facial details. The MNIST dataset collects $70,000$ handwritten digit images from 0 to 9, we randomly choose 130 images for each digit in our experiment. The USPS dataset is composed of $9,298$ handwritten digit images from 0 to 9, we randomly select 80 images for each digit in our experiment.

Three linear dimension reduction methods, i.e., principal component analysis (PCA) [119], linear discriminant analysis (LDA) [87], neighborhood preserving embedding (NPE) [112] and their double shrinkage versions, i.e., DS-PCA, DS-LDA, are compared on both datasets. For DS-PCA, P is the negative covariance matrix $-X^TX$. For DS-LDA, P is $-[S_W^\dagger S_B + (S_W^\dagger S_B)^T]/2$, wherein S_W is within-class scatter matrix, S_B is between-class scatter matrix, and $\cdot^\dagger$ denotes the Moore-Penrose matrix inverse.

In each experiment, we initially obtain the dense projection matrix from a linear dimension reduction method and a corresponding 66% sparse projection matrix (2/3 entries are zeros) from its double shrinkage version. Then the test data is projected onto these two low dimensional subspaces defined by the two

projection matrices, respectively. Finally, the nearest neighbor classifier is used for classification.

Figure 3.1-Figure 3.6 show the classification performance of these 3 linear dimension reduction methods and their corresponding double shrinkage versions on the 6 datasets, respectively. All the figures show the recognition rates of double shrinkage versions are comparable to the corresponding linear dimension reduction methods on each dataset. This observation indicates that a very sparse projection matrix obtained by using DSA includes sufficient discriminative information with much less storage requirements (1/3 of the dense one's in the experiments).

For human face classification experiments, In each plot below the performance evaluation curves, we show the first 10 projection vectors of a linear dimension reduction method and the first 10 projection vectors of its double shrinkage version. Comparing against the dense ones, although the sparse projection vectors are blank at most areas, the important biological features, e.g., eyebrows, eyes, nose, mouth, mustache and profile, are usually selected for subsequent classification. Since these selected features have clear physical meanings, the sparse projection vectors can provide an explicit interpretation to the new coordinate. Therefore, by building a sparse projection matrix, double shrinkage provides a more efficient strategy to compress the useful information for subsequent classification and a more explicit interpretation to the obtained subspace than the conventional linear dimension reduction algorithms.

3.6.2 Nonlinear manifold learning

Nonlinear manifold learning remains the manifold structure of the high dimensional data in its low dimensional representations. In order to explore the advantages of double shrinkage for nonlinear manifold learning, we design two experiments based on the double shrinkage versions of ISOMAP [210] and LLE [196]. In each experiment, the matrix P is calculated according to [17]. Then DSA is applied to the given data for producing a sparse low dimensional representation.

We run the double shrinking version of ISOMAP (DS-ISOMAP) on the face dataset used in [210]. ISOMAP preserves global geodesic distances of all sample

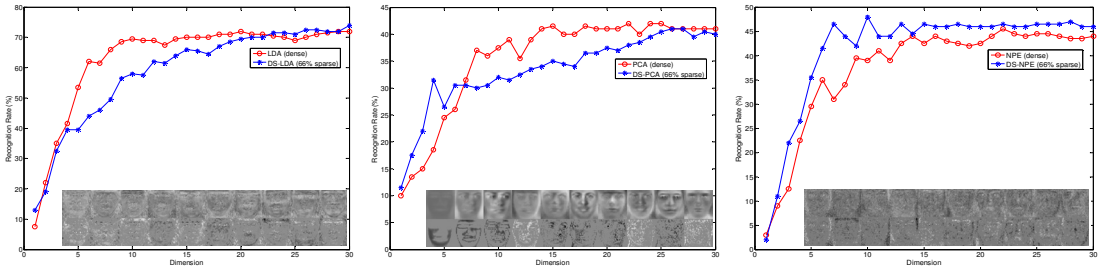

Figure 3.1: (FERET) Recognition rate vs. Subspace dimensions curves of LDA, PCA, NPE and their double shrinkage versions on FERET face dataset. The first 10 dense eigenfaces obtained via eigenvalue decomposition (the top row) and the corresponding 10 66% sparse eigenfaces obtained via double shrinkage (the bottom row) are shown on the bottom of each plot.

pairs. The face dataset is composed of 698 images of size 64×64 with different poses and light directions. In Fig 3.7, the two dimensional embedding of the face data obtained via DS-ISOMAP is presented with the neighborhood graph of the original data. The short distance between connected samples indicates that the neighborhood structure of the original data is well preserved in the two dimensional embedding. Because of the sparsity obtained by using double shrinking, 50% of the samples are projected onto the two axes in their low dimensional representations. The sample images on the two axes exhibited in Fig 3.7 imply that DS-ISOMAP entirely recovers the intrinsic geometric structure of the face data. For instance, the samples on X axis encode the poses from top-left to bottom-right with smoothing changing illumination. Thus the global geometric structure of the original data is well preserved. Note that the original ISOMAP results assign left-right pose change to the X axis, which is an obvious difference produced by dense representation. Therefore, double shrinking can find the intrinsic dimension of the data with favor of sparsity and inherit advantages from the used nonlinear manifold learning method.

We run the double shrinkage version of LLE (DS-LLE) on a subset of COIL-20 [180], including two objects duck and cat. Therefore, the intrinsic dimension of the subset is two and we can embed the samples in a two dimensional subspace for visualization. LLE [196] preserves the local geometry by retaining the neighbor reconstruction coefficients. The images of each object are taken 5 degrees apart

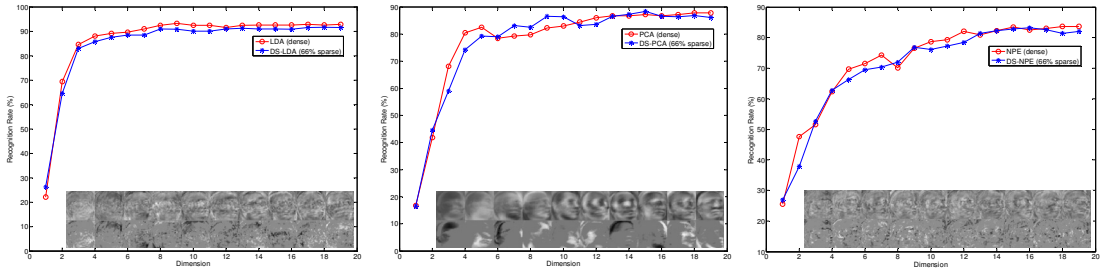

Figure 3.2: (UMIST) Recognition rate vs. Subspace dimensions curves of LDA, PCA, NPE and their double shrinkage versions on UMIST face dataset. The first 10 dense eigenfaces obtained via eigenvalue decomposition (the top row) and the corresponding 10 66% sparse eigenfaces obtained via double shrinkage (the bottom row) are shown on the bottom of each plot.

as the object is rotated on a turning table and each object has 72 images with size 32×32.

Figure 3.8 presents the two dimensional DS-LLE embedding by using the neighborhood graph of the original data. Because of the sparsity obtained by double shrinkage, most samples (90%) are projected onto the two axes. The sample images on the two axes are shown at the bottom of Figure 3.8. They imply that most duck images are distributed along the X axis, while most cat images are distributed along the Y axis. This observation indicates that double shrinkage is able to enhance the separability of low dimensional representations. Moreover, images on each axis preserve local similarity and smoothness over neighbor samples. This property is inherited from LLE. Therefore, compared against nonlinear manifold learning method, double shrinkage is able to provide sematic and more compact representations.

3.6.3 Data clustering

We apply DSA to PCA and obtain sparse low dimensional representations of given data for k-means [109] based clustering. The matrix P is $-XX^T$. The clustering results of PCA and DS-PCA are compared with each other in subspaces of different dimensions by using three different evaluation metrics, including sum of squares, accuracy and the normalized mutual information. Sum-of-squares adds the square deviation of each sample from its cluster center. Smaller sum-of-

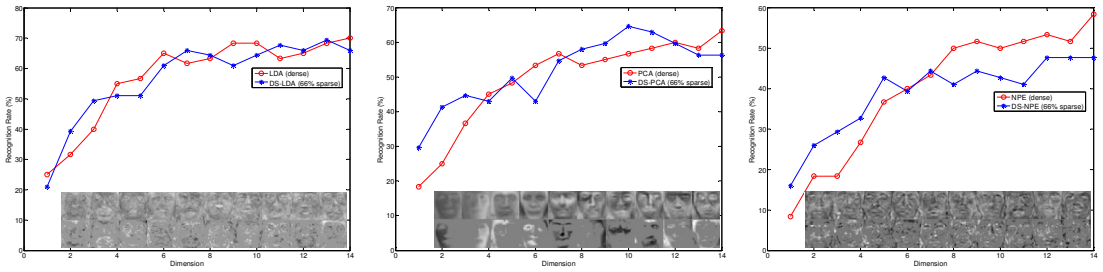

Figure 3.3: (YALE) Recognition rate vs. Subspace dimensions curves of LDA, PCA, NPE and their double shrinkage versions on YALE face dataset. The first 10 dense eigenfaces obtained via eigenvalue decomposition (the top row) and the corresponding 10 66% sparse eigenfaces obtained via double shrinkage (the bottom row) are shown on the bottom of each plot.

squares implies better clustering performance. Accuracy and normalized mutual information are defined in [234] and [32], respectively. Higher accuracy and larger normalized mutual information correspond to better clustering result. We test the clustering performance of double shrinkage on three datasets in the UCI machine learning repository [10], which are breast cancer, wine and Semeion handwritten digit dataset. The Breast cancer dataset includes 569 30-dimensional samples in 2 classes, the wine dataset includes 178 13-dimensional samples in 3 classes, and the Semeion dataset includes 1593 256-dimensional samples in 10 classes. The sparsity (the proposition of zero entries) in DSA is set as 60%.

Figure 3.9, Figure 3.10 and Figure 3.11 show the clustering results of the three evaluation metrics obtained by using PCA and DS-PCA on the three datasets. The sparse low dimensional representations obtained by DS-PCA outperform the dense representations obtained by PCA in most situations. This observation indicates that doubles shrinkage can effectively compress the original data and simultaneously preserve the important information for subsequent clustering. This advantage should be attributed to the sparsity of the low dimensional representations, which enhance the separability of the data.

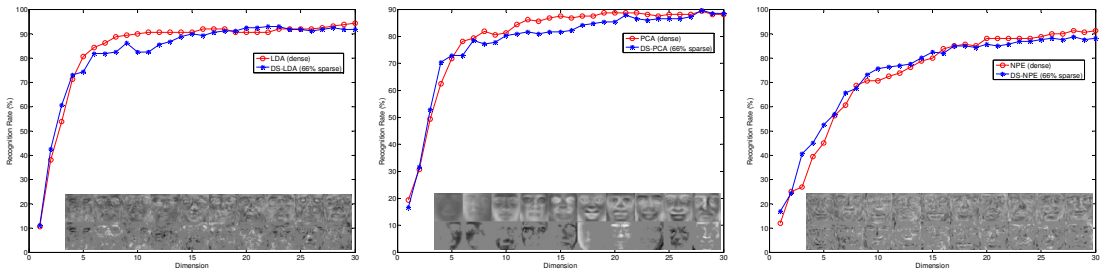

Figure 3.4: (ORL) Recognition rate vs. Subspace dimensions curves of LDA, PCA, NPE and their double shrinkage versions on ORL face dataset. The first 10 dense eigenfaces obtained via eigenvalue decomposition (the top row) and the corresponding 10 66% sparse eigenfaces obtained via double shrinkage (the bottom row) are shown on the bottom of each plot.

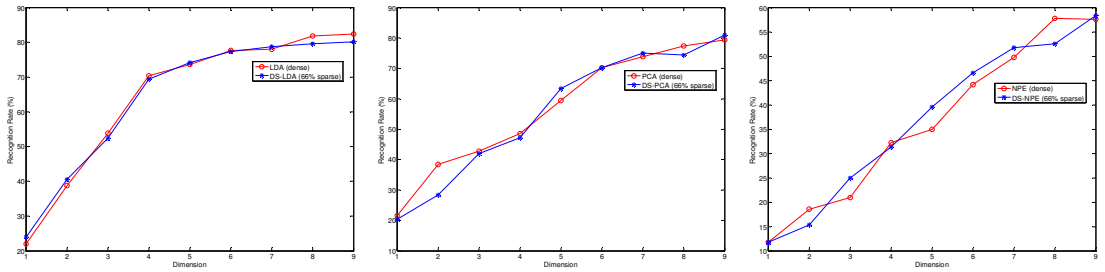

Figure 3.5: (MNIST) Recognition rate vs. Subspace dimensions curves of LDA, PCA, NPE and their double shrinkage versions on MNIST handwritten digit dataset. The corresponding projection matrices are 66% sparse.

3.6.4 Feature selection

We test the performance of DSA for solving sparse PCA and selecting critical features from a collection of high dimensional data. In this experiment, the matrix P is $-X^TX$. We compare the explained variance of DSA and popular existing sparse PCA algorithms, i.e., Sparse PCA [273], Greedy SPCA [176], Path SPCA [59], sPCA-rSVD [201] and SPC [229] on different cardinalities. We do not include DSPCA [60] in our experiments, because it relaxes the sparse PCA problem to an SDP problem and thus has an expensive computational complexity of $\mathcal{O}(n^3)$ per iteration round. We choose to use Path SPCA [59], because it is a faster alternative of DSPCA.

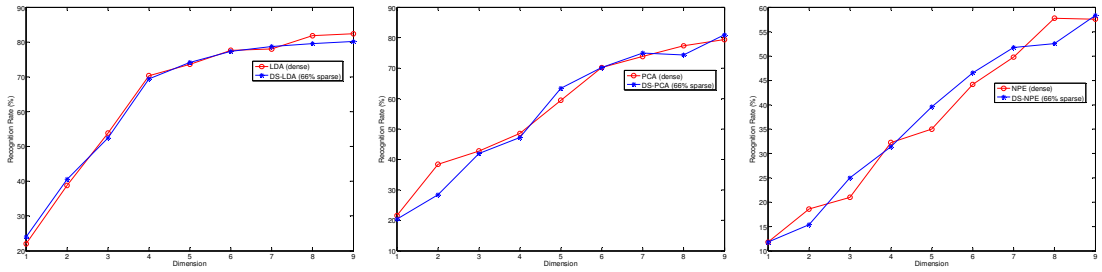

Figure 3.6: (USPS) Recognition rate vs. Subspace dimensions curves of LDA, PCA, NPE and their double shrinkage versions on USPS handwritten digit dataset. The corresponding projection matrices are 66% sparse.

We use the explained variance as the evaluation criterion of the obtained sparse loading vectors. When only one sparse loading vector v is considered, the variance explained by the corresponding component Xv is

$$Var(v) = v^T X^T X v. \tag{3.80}$$

When the obtained sparse loading vectors are more than one, for example, V including k sparse loading vectors, the corresponding components are possibly to be correlated. Thus, summing up the variance explained individually by each of the components overestimates the variance explained by all the components. In this case, we use the QR decomposition of the first k sparse PCs $XV = QR$, and define the variance explained by the k corresponding components XV as

$$Var(V) = \sum_{i=1}^{k} R_{i,i}^2. \tag{3.81}$$

The proportion of the explained variance is defined as $Var(V)/Var(V')$, wherein V' is the first k loading vectors obtained by the classical PCA.

The "pitprops" dataset, composed of 180 observations of 13 variables, is a standard benchmark to evaluate algorithms for solving sparse PCA in [212], [176] ,[201] and [273]. Following the setting used in previous studies, we compute the first 6 sparse loading vectors from the 13×13 pitprops covariance matrix by

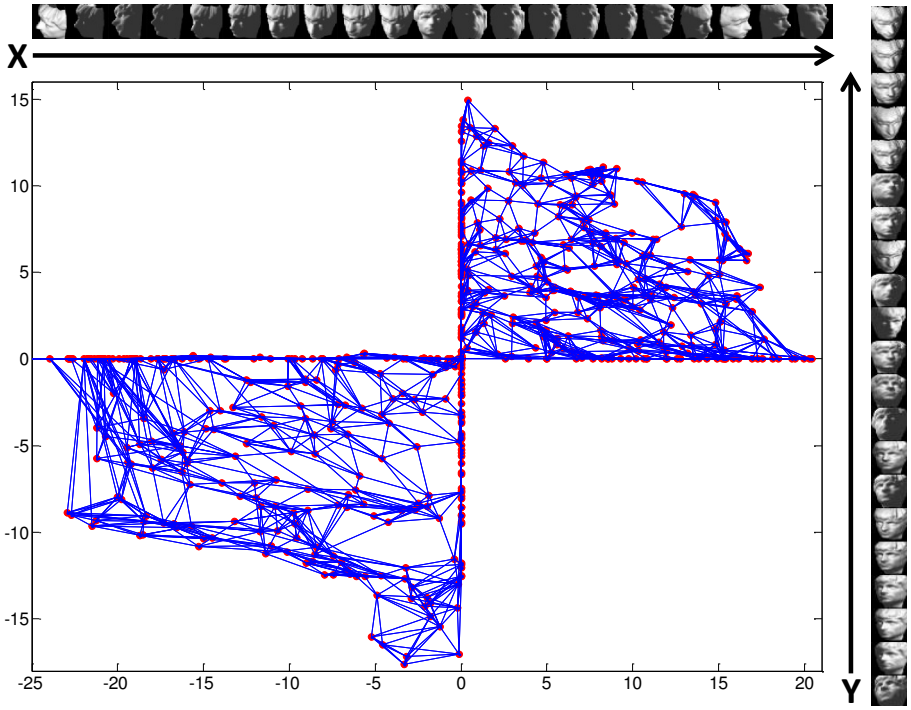

Figure 3.7: (3D face) Two-dimensional embedding (with neighborhood graph of the original data) of 698 64×64 face images via double shrinking-ISOMAP. The images were sampled from a face rendered with different poses. Illumination differences were artificially eliminated. 50% of the face images have sparse representations in the two-dimensional subspace and thus are projected on the two coordinate axes X and Y. We sample 21 images from each axis and show them on the top and right of this figure, respectively.

using DSA. In Table 3.6.4, we present the total cardinality and the proportion of the explained variance of the first 6 sparse loading vectors obtained by Sparse PCA, rSVD, Greedy SPCA and DSA. The experimental results illustrate that the sparse loading vectors obtained by DSA can retain more variance with the same or even smaller cardinality than existing sparse PCA algorithms.

Since DSA is able to build the whole solution path of sparse PCA, we take the construction of the solutions with the total cardinality of 24 in Table 3.6.4

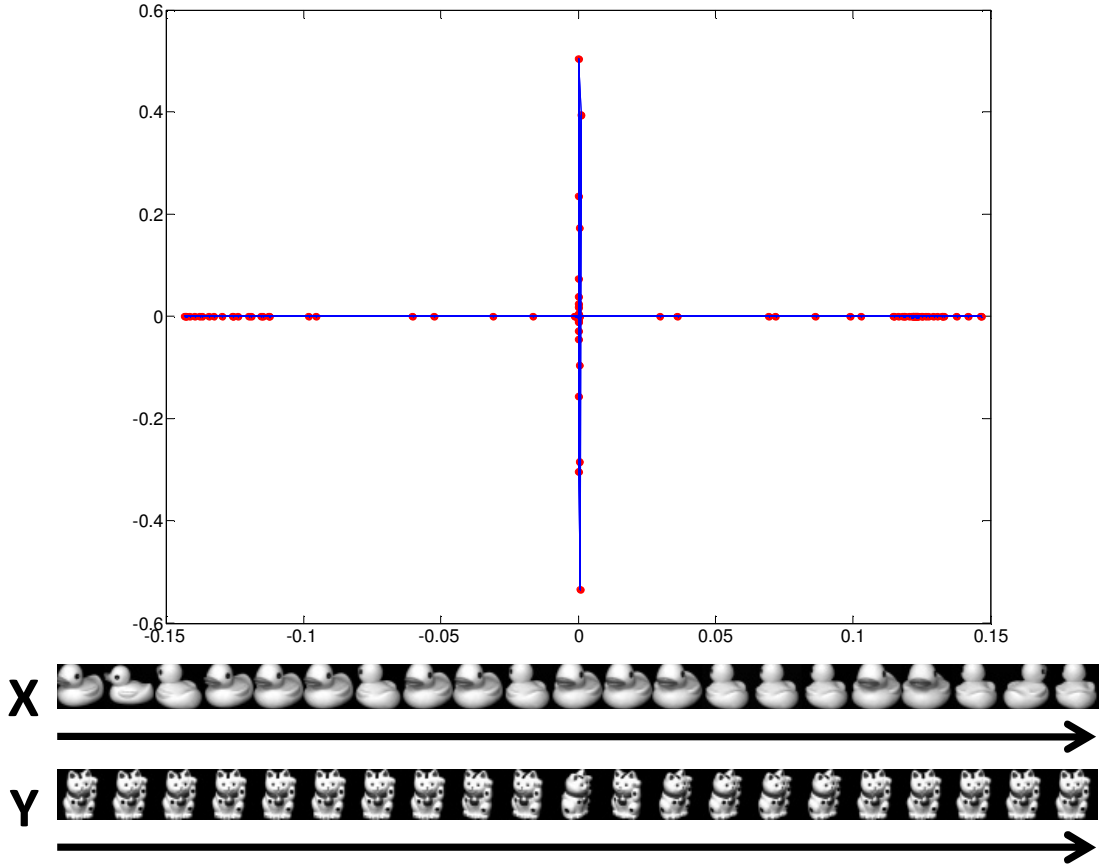

Figure 3.8: (COIL-20) Two-dimensional embedding (with neighborhood graph of the original data) of 144 32×32 images of two objects (a toy cat and a toy duck) via double shrinkage-LLE. The images were sampled from a toy cat and a toy duck rendered with different poses. 90% of the images have sparse representations in the two-dimensional subspace and thus are projected on the two coordinate axes X and Y. We sample 21 images from each axis and show them on the top and right of this figure, respectively.

for example. We plot the explained variance vs. cardinality curves for all the solutions on the path of the first three loading vectors and the corresponding solution paths in Figure 3.12. We use red cross and vertical red dash-dot line to mark the selected sparse solution. The variance-cardinality trade-off curves in Figure 3.12 indicate that a few sparse loading vectors with small cardinality can explain sufficient variance. In double shrinkage, the magnitudes of the trivial variables keep shrinking to zeros, while the importances of the critical variables

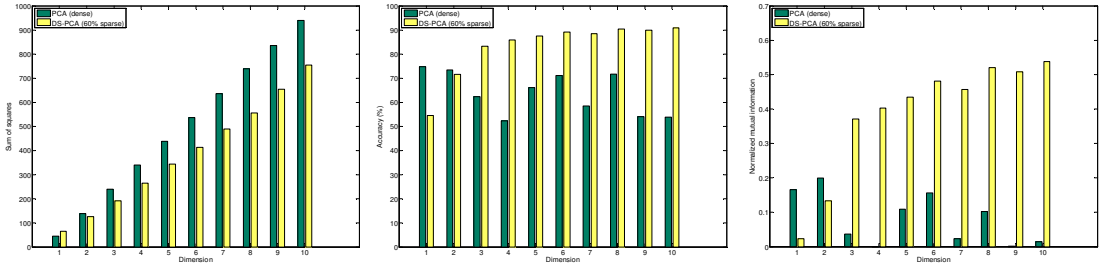

Figure 3.9: (Breast cancer) Sum of squares vs. Subspace dimensions (left), Accuracy vs. Subspace dimensions (middle), Normalized mutual information vs. Subspace dimensions (right) of clustering results on low dimensional representations of breast cancer data via PCA and double shrinkage-PCA. There are 60% samples owing zero representations on each coordinate obtained via double shrinkage.

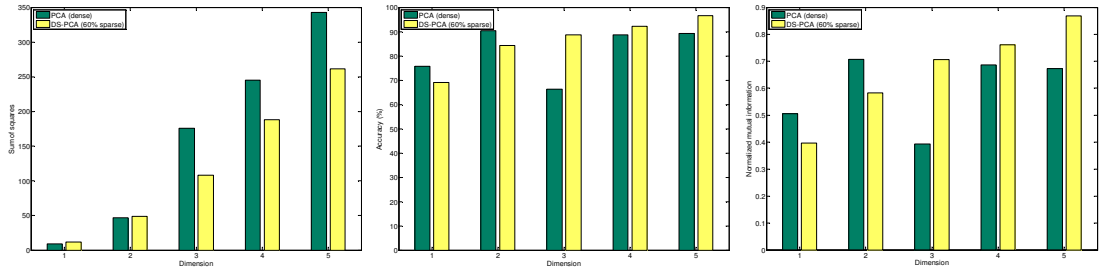

Figure 3.10: (Wine) Sum of squares vs. Subspace dimensions (left), Accuracy vs. Subspace dimensions (middle), Normalized mutual information vs. Subspace dimensions (right) of clustering results on low dimensional representations of wine data via PCA and double shrinkage-PCA. There are 60% samples owing zero representations on each coordinate obtained via double shrinkage.

keeps increasing, so the solution path of each variable in Figure 3.12 is piecewise linear and monotone in most intervals. However, double shrinkage also allows transfers between the trivial variable set and the critical variable set when the importance of a trivial variable reaches the minimal importance of the critical ones, or when the magnitude of a critical variable is shrunk to zero. Thus some solution paths in Figure 3.12 change their directions (from increasing to decreasing or from decreasing to increasing) on the way.

We then compare DSA with other sparse PCA algorithms on two gene ex-

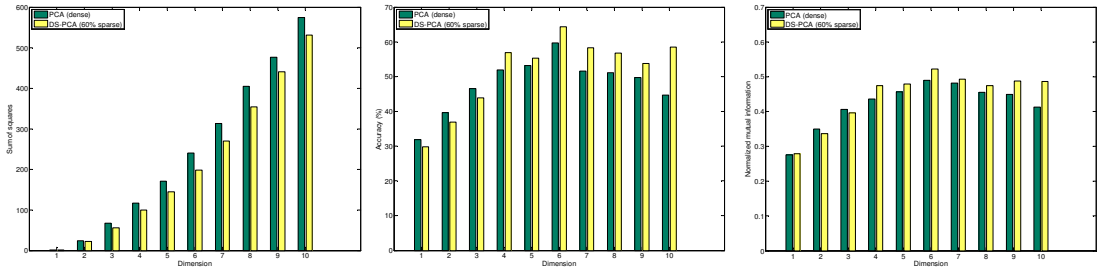

Figure 3.11: (Semeion) Sum of squares vs. Subspace dimensions (left), Accuracy vs. Subspace dimensions (middle), Normalized mutual information vs. Subspace dimensions (right) of clustering results on low dimensional representations of Semeion handwritten digit data via PCA and double shrinkage-PCA. There are 60% samples owing zero representations on each coordinate obtained via double shrinkage.

pression data sets, i.e., Colon cancer [5] and Lymphoma [4]. On both datasets, we consider the 500 genes with largest variance. For each sparse PCA method except SPC, the solution path of the first sparse loading vector (including 500 solutions) is computed. For SPC, because it adjusts the cardinality of the solution by tuning the parameter in the ℓ_1 constraint, the whole solution path is difficult to obtain. Thus we compute 10 sparse solutions with different cardinalities by adjusting the parameter c_2 of constraint $\|v\|_1 \leq c_2$ within the range $[1, \sqrt{500}]$. We show their variance vs. cardinality trade-off curves in Fig 3.13, together with the corresponding computation time. Note that SPC computes 10 sparse solutions in the shown computation time, while each of the other method computes 500 solutions to build the whole solution path. The curves demonstrate that comparing with other sparse PCA algorithms, DSA can obtain sparse solution with comparable variance in much less CPU seconds on different cardinalities.

3.6.5 Scalability study

We compare the scalability of DSA with other sparse PCA solvers on two artificial datasets, including a 100×100 Gaussian random matrix and a 500×500 Gaussian random matrix. The variance-cardinality trade-off curves of different methods and the corresponding time costs are shown in Fig 3.14. We also compute 10

Table 3.2: Total cardinality and proportion of explained variance of the first 6 sparse principal components obtained via different methods from pitprops data. The results of Sparse PCA, rSVD and Greedy SPCA are calculated from the sparse loading vectors published in [273], [201] and [176], respectively.

Method	Total Card.	Variance Prop.
Sparse PCA [273]	18	75.80%
rSVD [201]	25	79.24%
Greedy SPCA [176]	12	54.06%
	14	71.50%
Double Shrinkage	10	58.06%
	14	72.28%
	18	76.24%
	21	79.87%
	24	80.62%

solutions in each of SPC experiments and show the total time cost for obtaining the 10 solutions. The results imply that Greedy SPCA, Path SPCA and DSA have comparable performance on the explained variance of the obtained sparse loading vectors, while the explained variances of the sparse loading vector obtained by SPCA and SPC are smaller. Double shrinkage has the lowest time cost among all the algorithms. In addition, the CPU seconds of DSA increase slowly with the increasing data size compared with the other solvers. This observation is consistent with the time complexity analysis of DSA. The appealing scalability of DSA suggests its priority in solving large scale problems.

For clearer comparison, we list the time costs of different methods on the 2 gene datasets and 2 artificial datasets in Table 3.6.5. Note the time cost of SPC denotes the time for computing 10 sparse solutions rather than all the solutions on a solution path.

3.7 Conclusion

This paper proposed double shrinkage for data compression. Different from existing dimension reduction methods, double shrinkage compresses data by simulta-

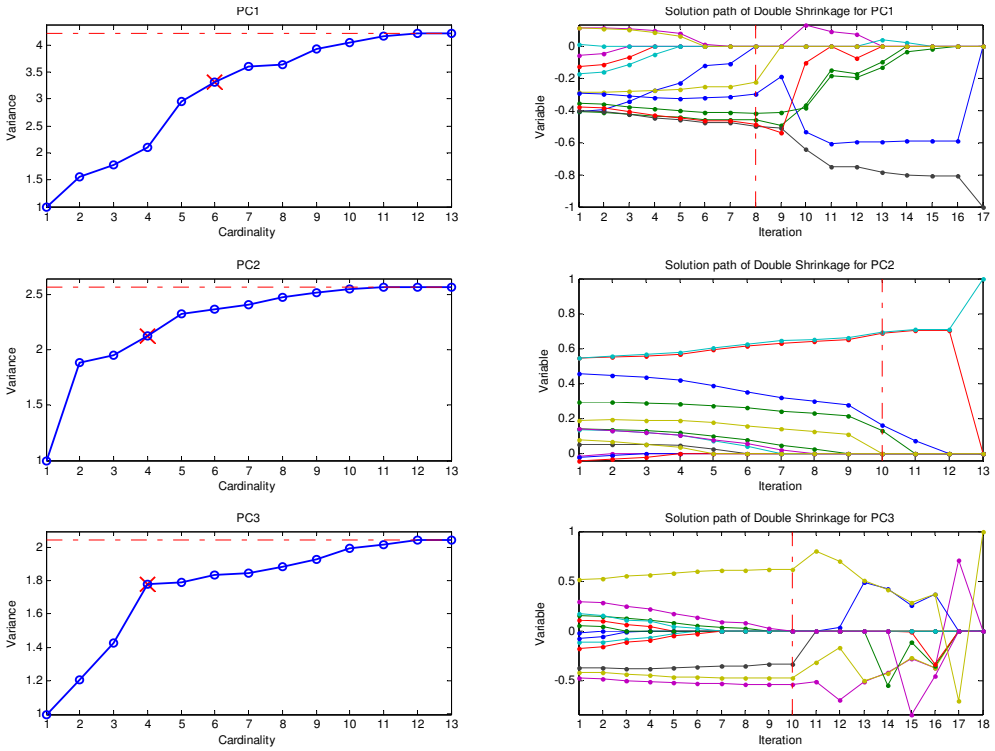

Figure 3.12: (Pitprops) Variance vs. Cardinality curves for the first 3 sparse principle components of the covariance matrix of Pitprops data obtained via double shrinkage and their corresponding solution paths. In the Variance vs. Cardinality plot, the red dash-dot line on the top of is the variance of the corresponding dense principle component, the red cross on each curve marks the corresponding selected principle component. In the solution path plot, the vertical red dash-dot line in each plot marks the step at which the sparse principle component is selected, curves with different colors represent the change of different variables.

neously shrinking both dimensionality and cardinality. It improves the low dimensional representation by exploiting the promising properties of sparsity, which has been successfully applied to problems in signal processing and statistics. Double shrinkage model (DSM) is an ℓ_1 penalized eigenvalue maximization/minimization with an unitary constraint. It consists of manifold embedding and the ℓ_1 norm penalty to shrink the data dimensionality and cardinality, respectively.

We then developed double shrinkage algorithm (DSA) to optimize DSM. DSA is a path-following algorithm that can build the whole solution path of DSM ef-

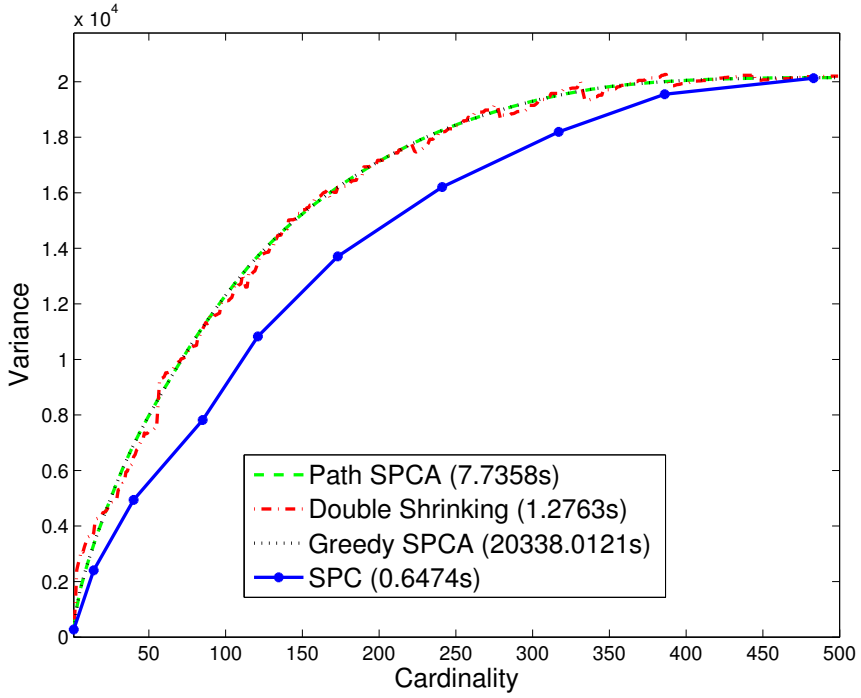

Figure 3.13: Trade-off curves between explained variance and cardinality for the first sparse principal component of colon cancer data (left) and lymphoma data (right). Different Sparse PCA methods (Greedy search, Path SPCA, SPC, Double shrinkage) are compared with each other. SPC computes 10 sparse solutions of different cardinalities, while the other methods computes 500 solutions to build their solution paths. Their corresponding time costs are listed on the bottom of each plot.

ficiently. We analyzed the essential properties of DSA. Each solution on the solution path is proved to be at least an approximation of a local optimum on the corresponding sparse level. The time complexity of each iteration round is about $\mathcal{O}({s_A}^3 + {s_B}^2)$, wherein s_A and s_B are the number of the critical variables and trivial ones, respectively. The step size of each iteration round has a closed form, and thus its computation is efficient. DSA has only one free parameter a_1 that can be conveniently determined, so it can be applied in practice conveniently. Compared against the corresponding dimension reduction method, double shrinkage has promising priorities in providing explicit interpretation to selected features, decreasing the computational costs, improving the data representation for subse-

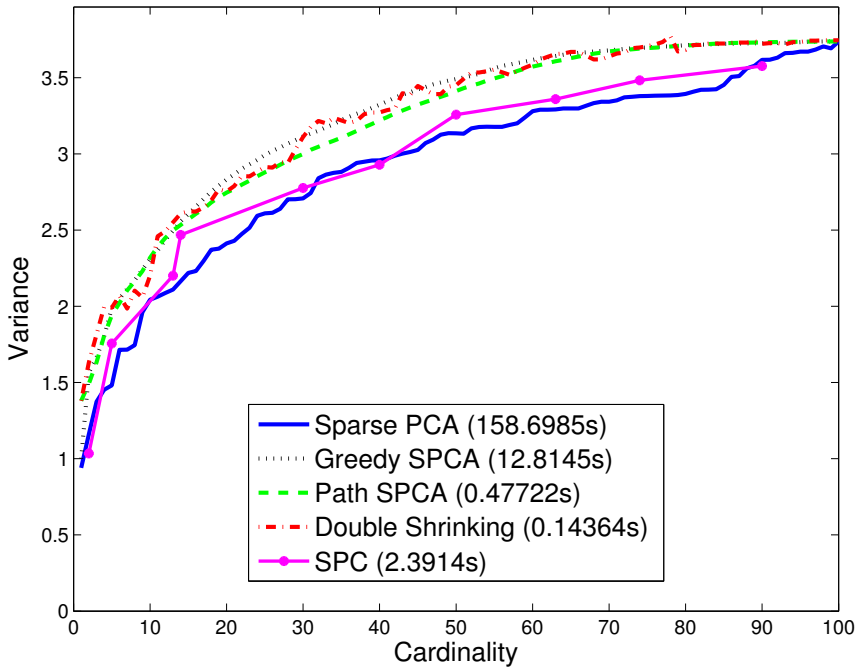

Figure 3.14: Trade-off curves between explained variance and cardinality for the first sparse principal component of a 100×100 gaussian random matrix (left) and a 500×500 gaussian random matrix (right), each entry of the matrix is sampled from an independent standard gaussian distribution. Different Sparse PCA methods (Sparse PCA, Greedy search, Path SPCA, SPC, Double shrinkage) are compared with each other. SPC computes 10 sparse solutions of different cardinalities, while the other methods computes 100 (left) or 500 (right) solutions to build their solution paths. Their corresponding time costs are listed on the bottom of each plot.

quent classification and clustering.

We applied double shrinkage to classification, manifold learning, clustering and feature selection on several datasets, such as face datasets, handwritten digit datasets, COIL-20 object dataset, datasets in UCI machine learning repository, gene expression data and artificial data. The sparse projection matrix obtained by double shrinkage produces competitive classification performance compared against the dense one obtained by the corresponding linear dimension reduction method. The selected sparse features have explicit interpretations. The sparse

Table 3.3: Time cost (CPU seconds) of Sparse PCA, Path SPCA (faster version of DSPCA), Greedy SPCA, SPC and Double Shrinkage on two gene datasets (colon cancer, lymphoma) and two artificial datasets (a 100×100 and a 500×500 Gaussian random matrix). Note the time cost of SPC denotes the time for computing 10 sparse solutions rather than all the solutions on a solution path.

Method	Colon cancer	Lymphoma	100×100	500×500
Sparse PCA	–	–	158.70	58385
Path SPCA	7.7358	6.9304	0.4772	7.7119
Greedy SPCA	20338	20743	12.815	20610
SPC	0.6474	0.6051	2.3914	7.2551
Double Shrinkage	1.2763	1.2658	0.1436	2.0399

low dimensional representations obtained by double shrinkage outperform the original dense ones in clustering. In nonlinear manifold learning, double shrinkage is able to recover the intrinsic geometric structure of data and associate semantics with the selected subspace coordinates by inheriting advantages of the corresponding manifold learning methods. Double shrinkage also has appealing performance in feature selection. Given the total cardinality, double shrinkage produces sparser loading vectors with more explained variance in less time than existing sparse PCA methods.

Double shrinkage has several important extensions. It can be conveniently extended to DSM using elastic net penalty and reweighted ℓ_1 minimization, which can produce sparse solutions with grouping effect and more zeros, respectively. The discriminative information can be encoded into the structured sparsity penalty and thus the classification performance of the sparse representation obtained via double shrinkage can be improved. Moreover, double shrinkage provides a scheme for other sparse learning problems with equality constraints, which are hard to be directly solved by existing ℓ_1 minimization algorithms.

Chapter 4

Divide-and-Conquer Anchoring for Near-separable Nonnegative Matrix Factorization and Completion

Nonnegative matrix factorization (NMF) becomes tractable in polynomial time with unique solution under *separability assumption*, which postulates all the data points are contained in the conical hull of a few anchor data points. Recently developed linear programming and greedy pursuit methods can pick out the anchors from noisy data and results in a near-separable NMF. But their efficiency could be seriously weakened in high dimensions. In this chapter, we show that the anchors can be precisely located from low-dimensional geometry of the data points even when their high dimensional features suffer from serious incompleteness. Our framework, entitled divide-and-conquer anchoring (DCA), divides the high-dimensional anchoring problem into a few cheaper sub-problems seeking anchors of data projections in low-dimensional random spaces, which can be solved in parallel by any near-separable NMF, and combines all the detected low-dimensional anchors via a fast hypothesis testing to identify the original anchors. We further develop two non-iterative anchoring algorithms in 1D and 2D spaces for data in convex hull and conical hull, respectively. These two rapid algorithms in the ul-

tra low dimensions suffice to generate a robust and efficient near-separable NMF for high-dimensional or incomplete data via DCA. Compared to existing methods, two vital advantages of DCA are its scalability for big data, and capability of handling incomplete and high-dimensional noisy data. A rigorous analysis proves that DCA is able to find the correct anchors of a rank-k matrix by solving $\mathcal{O}(k \log k)$ sub-problems. Finally, we show DCA outperforms state-of-the-art methods on various datasets and tasks.

4.1 Introduction

Nonnegative matrix factorization (NMF) [149, 50, 104, 103] decomposes a data matrix $X \in \mathbb{R}_+^{n \times m}$ containing n nonnegative m-dimensional data points $\{x_i\}_{i=1}^n$ in the form of $X = FW$, where $F \in \mathbb{R}_+^{n \times k}$ and $W \in \mathbb{R}_+^{k \times m}$ are also two nonnegative matrices and usually $k \ll \min(n, m)$. The rows of W are composed of k nonnegative basis vectors representing all the samples, while the n rows of F are nonnegative weight vectors encoding the n data points into conical combinations of the basis vectors. The nonnegativity of both F and W often results in more natural and interpretable part-based decomposition than other low-rank matrix factorizations [238], where the intrinsic parts extracted from W are usually sparse and reveals how a composite data point is generated. As a consequence, NMF has been broadly applied to numerous practical problems, such as text topic modeling, signal separation, social networks, collaborative filtering, dimension reduction, sparse coding [80] and feature selection [149]. Nonetheless, the theoretical properties of its solutions remained unclear for a long period until recently, and almost all the practical NMF algorithms [148, 158, 120, 138, 31, 67] rely on heuristics of alternating minimizing reconstruction error

$$\min_{F \in \mathbb{R}_+^{n \times k}, W \in \mathbb{R}_+^{k \times m}} \mathcal{D}(X, FW), \tag{4.1}$$

and cast the factorization into non-convex programming, where $\mathcal{D}(\cdot, \cdot)$ is a distance metric between two matrices.

4.1.1 Separable Nonnegative Matrix Factorization

The lack of rigorous analysis is the result of a fact that NMF is NP-hard [225] and highly ill-posed in its original form, because, in this case, the problem is computationally prohibitive, and neither uniqueness nor correctness guarantee can be made on its solution. It is therefore necessary to impose additional assumptions on the data points and reduce the original NMF to a tractable problem, meanwhile the generality of NMF is expected to be maintained maximally. An early effort is made in [70], which gave a *separability assumption*, under which the uniqueness of NMF solution can be achieved. Given the geometric concepts of (convex) cone, conical hull, simplex and convex hull, the separability assumption can be defined both geometrically and algebraically.

Definition 5. ***(Cone, conical hull, simplex, convex hull)*** *A (convex) cone is a non-empty convex set that is closed with respect to conical combinations of its elements. In particular, given a set of points $R = \{r_i\}_{i=1}^k$, which is comprised of k generators (rays) $r_i \in \mathbb{R}^m$, a cone $cone(R)$ can be defined by*

$$cone(R) = \left\{ \sum_{i=1}^{k} \alpha_i r_i \middle| r_i \in R, \alpha_i \in \mathbb{R}_+ \right\}. \tag{4.2}$$

And $cone(R)$ is the conical hull of R. Analogically, a simplex is a non-empty convex set that is closed with respect to convex combinations of its elements. In particular, given a set of points $V = \{v_i\}_{i=1}^k$, which is comprised of k generators (vertices) $v_i \in \mathbb{R}^m$, a simplex $\Delta(V)$ can be defined by

$$\Delta(V) = \left\{ \sum_{i=1}^{k} \alpha_i v_i \middle| v_i \in V, \alpha_i \in \mathbb{R}_+, \sum_{i=1}^{k} \alpha_i = 1 \right\}. \tag{4.3}$$

And $\Delta(V)$ is the convex hull of V. Hence, simplex defined by a convex hull is a special case of cone defined by a conical hull.

Definition 6. ***(Separability assumption)*** *All the data points in X reside in a conical hull of R, which is a subset $A \subseteq [n]$ of data points in X. Geometrically,*

the separability assumption is

$$\forall i \in [n], x_i \in cone\left(X_A\right), X_A = \{x_i\}_{i \in A}. \tag{4.4}$$

Algebraically, separability assumption means

$$X = FX_A, \Pi F = \begin{bmatrix} I_k \\ F' \end{bmatrix} \tag{4.5}$$

where I_k is a k-by-k identity matrix, $F' \in \mathbb{R}_+^{(n-k)\times k}$, and Π is a row permutation matrix.

In separability assumption, the data points in X_A are extreme rays of the cone $cone(X_A)$ because none of them, or their generators, can be expressed by conical combinations of other elements in $cone(X_A)$[1]. In addition, $cone(X_A)$ is finitely generated because all the elements in X are conical combinations of a finite set X_A. It is also pointed because its nonnegativity does not allow it containing both a vector x and $-x$. According to a basic law [179]: *a finitely generated and pointed cone $cone(R)$ possesses a finite and unique set of extreme rays R, and $cone(R)$ is the conical hull generated by these extreme rays R*, so we immediately obtain the uniqueness of solution. Note in this case, the original NP-hard NMF problem is reduced to finding the k extreme rays (or the "anchors") from data points in X, which can be solved in polynomial time.

Separability assumption selects a few data points to represent the other data points in the whole dataset. This constraint is more than merely an artificial trick: it is favored and justified by various practical applications. For example, it implies a word associated with a unique topic in topic modeling [114], an audio signal that only belongs to one source in blind source separation [50], and a highly representative user in collaborative filtering [241, 94]. Moreover, in big data challenges, it is more natural, interpretable and efficient to represent high-dimensional data by a few actual data points selected from a huge data pool rather than artificial basis vectors. In fact, such "data expresses itself" assumption has

[1]Here we assume that X includes sufficient data points and thus can completely represent $cone(X_A)$.

become a popular trend in the recent study of other related matrix factorization, such as rank revealing QR [102], CUR [75] and subspace clustering [78].

4.1.2 Related Works

A special case of separable NMF was firstly studied in [9] and [23], in which all data points are assumed to be normalized to have unit ℓ_1 norm and are included in a convex hull. So the goal is to find out the k extreme points, or the vertices of the convex hull from X. Since a vertex cannot be expressed as a convex combination of other data points, a linear programming (LP) pursuing the combination weights can be conducted on each data point to test if such expression exists [9]. If the LP is infeasible, an extreme point is detected. However, this algorithm [9] needs to solve n LP each optimizes $n-1$ variables and is not scalable for large-scale problems. A single LP method named "Hottopixx" [23] is proposed to solve the separable NMF by decomposing $X = CX$ in an elegantly designed feasibility polyhedral of C, in which an element implies a feasible F in (4.5). Hottopixx solves an LP of n^2 variables. Thus, a stochastic gradient method with only asymptotic convergence is specially developed for applying it to large-scale problems. It is also robust to small data noise, but the noise level needs to be estimated in advance. This problem considering noise is called **near-separable NMF**, which is defined as finding X_A from $X = FX_A + N$, where N is noise. A more robust and flexible LP method adopting a new LP model modified from Hottopixx is proposed in [96]. Compared to Hottopixx, it automatically detects rank k, and does not demand normalization to data points so it can address general near-separable NMF within a conical hull. Since most LP methods aim at optimizing C rather than finding X_A, post-processing that extracts X_A from C is usually indispensable.

Different from LP, another class of near-separable NMF algorithms is based on the methodology of greedy pursuit. The key idea is to select a data point on the direction of which the current residual can be decreased at a fast speed, and add the selected data point into X_A as a new anchor. The weight matrix F and the residual matrix $X - FX_A$ are updated after each update of X_A. The above procedure is iterated until reaching an error tolerance or a sufficiently large A.

Different algorithms adopt a distinct residual in their selection criterion and update of residual. For example, successful projection algorithm (SPA) [97] adopts a modified Gram-Schmidt orthogonalization with column/row pivoting. XRAY [145] selects a new anchor according to the residual of a randomly selected exterior data point, the maximum residual among all exterior points, or their averaged residual, and updates the residual matrix by solving a nonnegative least square regression. These greedy pursuit algorithms normally have smaller computational complexity than their LP rivals, but their iterative selection procedures can only be computed in single thread due to the nature of greedy method. Thus, a heavy computational burden may still arise in high dimensions for them. Although they can be applied to noisy data, it is always more difficult to achieve a rigid robustness on them than on LP methods.

4.1.3 Motivation and Main Contributions

In this chapter, we investigate how to precisely locate the anchor set X_A of near-separable NMF in two difficult cases commonly arising in big data challenges but rarely studied in previous separable NMF literature. In particular, we develop a highly scalable algorithm framework addressing high-dimensional (such as dimension reduction methods [119, 196, 210]) or incomplete data matrix with missing values (such as matrix completion methods [39, 36, 132, 232, 258]). The former case usually makes the computations of previous methods impractical, while the latter case cannot be handled by any existing algorithm.

Our framework, entitled divide-and-conquer anchoring (DCA), exploits the geometric connections between convex hull or conical hull and their projections in low dimensions, and estimates their anchors (i.e., vertices of convex hull or extreme rays of conical hull) from simple statistics to the detected anchors of data projections [226] in low-dimensional spaces. In its divide step, DCA solves a number of sub-problems in parallel. Each sub-problem aims at finding the anchors of a low-dimensional conical hull covering the random projections of the original data points. The sub-problem can be solved by arbitrary near-separable NMF, while the minimum number of sub-problems producing an accurate estimation of X_A is determined only by k. In its conquer step, DCA conducts

a fast hypothesis testing that selects anchors based on simple statistics to the collected low-dimensional anchors. We demonstrate that DCA performs much more efficiently than other methods in high dimensions because, 1) the product of sub-problem's complexity and number in DCA could be much smaller than the complexity of the original problem in high dimensions, especially when $k \ll m$; and, 2) the multiple sub-problems in DCA are solved independently so parallel or distributed computing such as MapReduce [62] can be directly leveraged to gain further acceleration.

Another primary advantage of DCA is its capability to handle incomplete data matrix with massive missing values. Since solving the sub-problems in the divide step only requires partial low-dimensional geometry information of the original data points, and the hypothesis testing in the conquer step treats anchors in each low dimensional space as a random trial in statistics, very few entries of the original data matrix are sufficient to provide an effective "sub-sampling " leading to a reliable NMF solution. Different to DCA for fully observable data, in the divide step, each low-dimensional anchor is scored by the number of available data projections in the associated space. In the conquer step, the k anchors are selected according to the proportion rather than the times of each point being selected as anchors in low dimensions, because the numbers of available projections for different data points might differ from each other due to their missing features on different dimensions. DCA still enjoys promising robustness to noise even in the incomplete case.

DCA gives each sub-problem the freedom to invoke any near-separable NMF in arbitrary lower dimensions. This property makes DCA an unified framework rather than a sole algorithm. However, we also discover that a much simpler and efficient algorithm can replace the previous sophisticated separable NMF in ultra-low dimensions and achieve a comparable performance. In particular, we build two rapid anchoring algorithms; one finds the vertices of a convex hull in 1D space, and the other finds the extreme rays of a conical hull in 2D space. We show that when the noise is not too big, DCA invoking these two simple algorithms is adequately powerful to identify all the high dimensional anchors precisely even in missing value situations.

4.2 Divide-and-Conquer Anchoring (DCA)

DCA is a divide-and-conquer framework [170] for near-separable NMF and with two steps: the divide step equals applying near-separable NMF to data random projections in multiple subspaces, whilst the conquer step is a fast hypothesis testing based on statistics of the low-dimensional anchors achieved in the divide step. Note the concept of "divide-and-conquer" for DCA is more close to that used in [170] and is slightly different from that used in algorithm design, because DCA directly divides the original problem into several solvable sub-problems rather than recursively divides those sub-problems into smaller sub-problems.

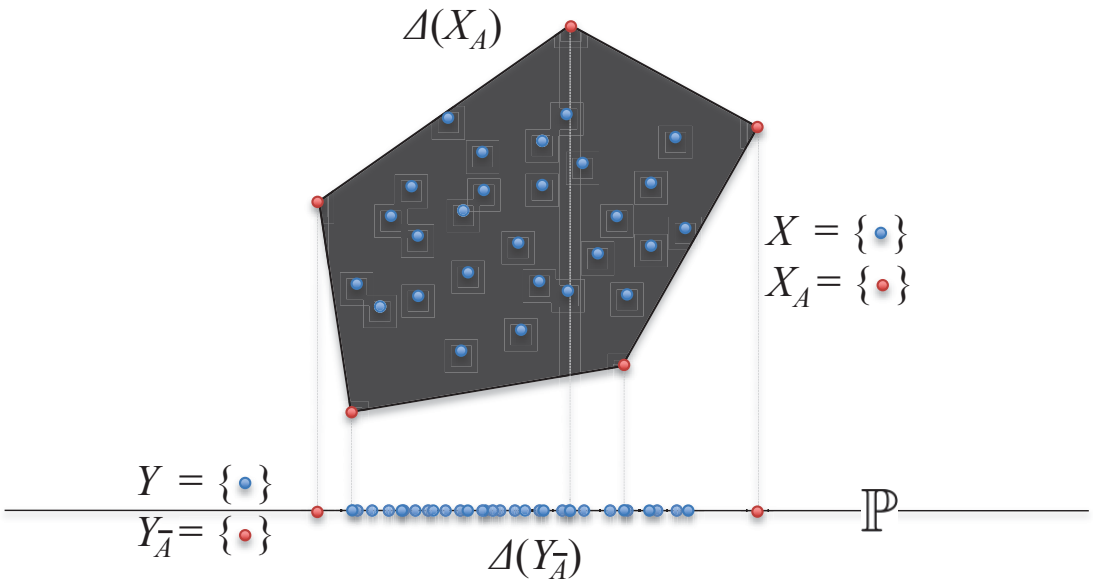

Figure 4.1: Sub-problem in the divide step of DCA: finding the low-dimensional anchors $Y_{\bar{A}}$ on hyperplane $\mathbb{P}$ when all data points X are contained in a *convex hull* of k anchors (vertices) X_A.

4.2.1 Divide Step: Anchoring on Low-dimensional Projections

In each sub-problem, DCA projects all the data points (i.e., the row vectors) in X to a randomly generated d-dimensional hyperplane $\mathbb{P} = \mathbb{P}(B)$. This ran-

dom $\mathbb{P}$ denotes a subspace $B = [\beta_1; \beta_2; \cdots; \beta_d] \in \mathbb{R}^{d \times m}$ spanned by d random vectors $\{\beta_i\}_{i=1}^{d}$ uniformly sampled from the unit hypersphere $\mathbb{S}^{m-1}$ in $\mathbb{R}^m$. The projections of data matrix X onto $\mathbb{P}$ is

$$Y = XB^T, Y \in \mathbb{R}^{n \times d}. \tag{4.6}$$

Note in separable NMF, all data points in X are contained in the conical hull $cone(X_A)$ of an unknown subset X_A whose elements (rows) are selected from the rows of X. The divide step of DCA is based on a key fact: although the nonnegativity of X has not been retained in Y, the geometry of conical hull $cone(X_A)$ is partially preserved in Y. This fact enables us to bridge the high dimensional anchors and the anchors in $\mathbb{P}$. Specifically, if x_i denotes the i^{th} row of X and y_i denotes the i^{th} row of Y, $[n] = \{1, 2, \cdots, n\}$, we have the following theorem.

Theorem 6. ***(Geometry preserved in low dimensions)*** *If* $\forall i \in [n], x_i \in cone(X_A)$*, then for any hyperplane* $\mathbb{P}$ *of arbitrary dimensions* d*,* $\forall i \in [n], y_i \in cone(Y_{\bar{A}})$*, and* $\bar{A} \subseteq A$ *always holds in this case.*

The above theorem indicates that the n random projections Y of n data points X are contained in a conical hull of several anchors in $\mathbb{R}^d$ too. In addition, these anchors are rows of Y, and must be the random projections of anchors in X. In the noiseless case, this fact immediately implies a criterion to pick out anchors, i.e., *a data point whose random projection is an anchor in certain low-dimensional space should also be an anchor in the original space*. Here, the low-dimensional anchors in Y can be extracted by arbitrary separable NMF with computations far less than applying the same NMF algorithm to X. Illustrations of the projection geometry when X is included in a convex hull and a conical hull are given in Figure 4.1 and Figure 4.2, respectively.

However, the anchors of Y in merely one hyperplane cannot always reveal all the anchors in X. Because the projections of some anchors in X are possible to be interior points within the conical hull in low dimensions. It happens when the angle between the anchor and the hyperplane belongs to a specific region. This problem can be solved by repeating the low-dimensional anchor searching on multiple different hyperplanes, then any anchor in X could have chance to be

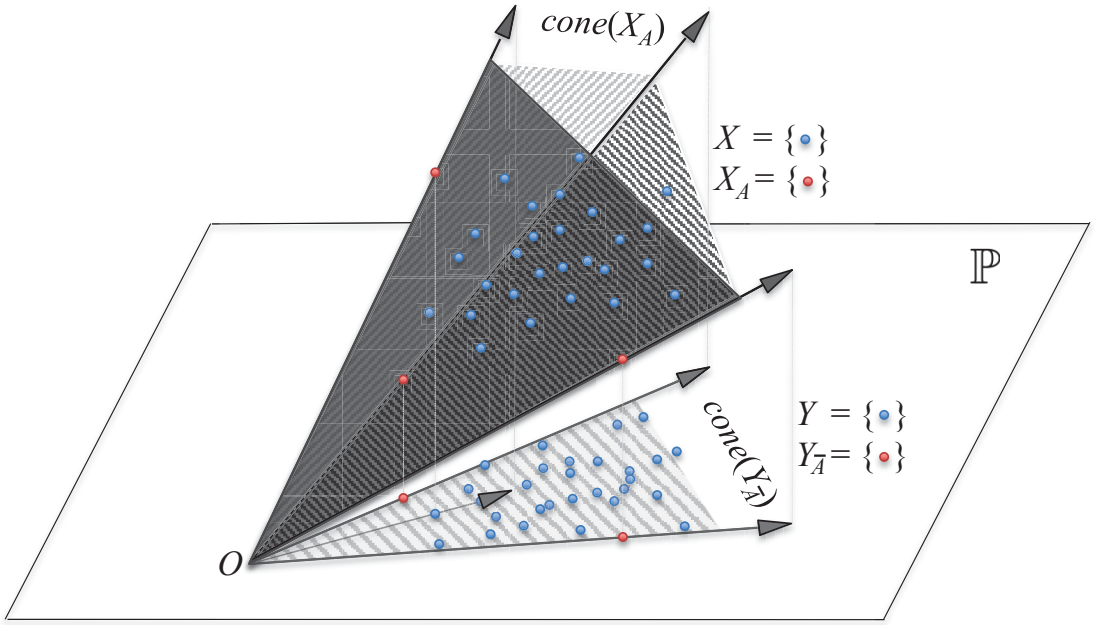

Figure 4.2: Sub-problem in the divide step of DCA: finding the low-dimensional anchors $Y_{\bar{A}}$ on hyperplane $\mathbb{P}$ when all data points X are contained in a *conical hull* of k anchors (extreme rays) X_A.

projected as a low-dimensional anchor. This equals to solving multiple independent sub-problems. If the output of separable NMF in each sub-problem is the low-dimensional anchors's indexes $\bar{A} := \text{SNMF}(Y)$, we use superscript to index different sub-problem, and assume there are s sub-problems, the divide step of DCA is

$$\bar{A}^i := \text{SNMF}\left(X(B^i)^T\right), i = 1, 2, \cdots, s. \tag{4.7}$$

In DCA, solving a sufficient number (a later analysis will show the number is $\mathcal{O}(k \log k)$) of sub-problems in parallel is able to guarantee that the collected low-dimensional anchors includes the projections of all the anchors with a high probability. In addition, the non-anchors cannot be selected in the sub-problems for noiseless cases, and can be selected but with much smaller probability than the actual anchors for noisy cases. These facts enable us to distinguish the anchors and non-anchors by combining the solutions of sub-problems in the conquer step of DCA.

4.2.2 Conquer Step: Hypothesis Testing

The conquer step of DCA is composed of a hypothesis testing that accepts or rejects each data point associated with a detected low-dimensional anchor (from the s sub-problems) as an anchor. Since the anchors are usually detected in sub-problems with higher probability than non-anchors, the hypothesis testing can then be reduced to picking out the k data points whose random projections are most frequently selected as anchors in all the sub-problems. Let $I(i \in \bar{A}^j) : i \to \{0, 1\}$ be the indicator function for the event that data index i is within $\bar{A}^j$ of the j^{th} sub-problem. For fully observable X, DCA estimates the k elements in A as the k data indexes associated with the top k largest $\sum_{j=1}^{s} I(i \in \bar{A}^j)$, i.e.,

$$A := \arg \max_{H \subseteq [n], |H|=k} \sum_{i \in H} \sum_{j=1}^{s} I(i \in \bar{A}^j), \tag{4.8}$$

where $|H|$ denotes the number of elements in set H. In the noiseless case, the non-anchors cannot to be selected in sub-problems, so the above conquer step (4.8) is reduced to finding all the elements appearing at least once in the solutions of sub-problems, i.e.,

$$A := \bigcup_{i=1}^{s} \bar{A}^i. \tag{4.9}$$

In some applications, the rank k is unknown and needs to be determined automatically. When the noise is not overwhelming, a large gap can be observed between anchor and non-anchor on their statistics $\sum_{j=1}^{s} I(i \in \bar{A}^j)$. Hence, a tolerance τ can be pre-defined to detect such gap in the sorted $\sum_{j=1}^{s} I(i \in \bar{A}^j)$ of all data points and automatically identify k. Let p be the new index set after sorting $\sum_{j=1}^{s} I(i \in \bar{A}^j)$ of all $i \in [n]$ in descending order (i.e., $p_l = i$ means the i^{th} data point has the l^{th} largest $\sum_{j=1}^{s} I(i \in \bar{A}^j)$ among the n data points), A can be estimated without knowing k by

$$\begin{aligned} &A := p_{[l^\star]}, l^\star = \min \left\{ l : g(p_l) - g(p_{l+1}) \geq \tau s \right\}, \\ &g(p_l) = \sum_{j=1}^{s} I(p_l \in \bar{A}^j). \end{aligned} \tag{4.10}$$

4.2.3 DCA for Incomplete Data Matrix

An appealing property of DCA that makes it different to other methods is its capability to handle incomplete matrix with massive missing values. This is possible for DCA because it merely requires to know partial geometry of the low-dimensional projections of data points rather than complete data points in high dimensions.

In this case, for each sub-problem, the random vectors $\{\beta_i\}_{i=1}^{d}$ spanning the subspace B of a hyperplane $\mathbb{P}(B)$ should be sparse and share the same support set Ω_B. Moreover, the random projection of a data point x_i on the hyperplane $\mathbb{P}(B)$ is legal for use only when the dimensions of x_i on Ω_B are not missing, i.e., when $\text{supp}(x_i) \subseteq \Omega_B$. In addition, Ω_B must ensure a sufficiently large set $\{i : \text{supp}(x_i) \subseteq \Omega_B\}$ in X whose data dimensions on Ω_B are not missing. Because this can guarantee a sufficiently large probability to involve anchors in the set. When the missing values in X are distributed at random, Ω_B can be selected as a small set composed of one or two randomly chosen dimensions. In contrast, when the observable set $\text{supp}(x_i)$ share some structural patterns across n data points, it is preferred to extract the dimensions that are frequently observed together in most data points to form an Ω_B.

Therefore, in the divide step of DCA for incomplete data, each sub-problem applies near-separable NMF to the low-dimensional random projections of a subset of data points, such that

$$\begin{aligned} &\bar{A}^i := \text{SNMF}\left(X_{\Phi_{B^i},\Omega_{B^i}}(B^i)^T\right), \\ &\Phi_{B^i} = \{i : \text{supp}(x_i) \subseteq \Omega_{B^i}\}, i = 1, 2, \cdots, s, \end{aligned} \tag{4.11}$$

where $X_{\Phi_{B^i},\Omega_{B^i}}$ denotes the sub-matrix of X built from rows indexed by Φ_{B^i} and columns indexed by Ω_{B^i}.

The matrix incompleteness also causes a difference in the conquer step. On the one hand, since the subsets of data points involved in the s sub-problems are of different sizes $|\Phi_{B^i}|$, and a larger subset indicates a higher confidence of finding real anchors rather than an interior point due to the absence of real anchors in the subset, we score the indicator function $I(i \in \bar{A}^j)$ in (4.8) by $|\Phi_{B^j}|/n$. On the other hand, different data points are involved in a different number of sub-problems, so

it is more robust to compare their proportions of being selected as anchors across the involved sub-problems, rather than their times of being selected as anchors. Therefore, the conquer step for incomplete data is

$$\begin{aligned} A &:= \arg \max_{H\subseteq[n],|H|=k} \sum_{i\in H} f(i), \\ f(i) &= \frac{1}{\sum_{j=1}^{s} I(i\in\Phi_{B^j})} \sum_{j=1}^{s} \frac{|\Phi_{B^j}|}{n} I(i \in \bar{A}^j). \end{aligned} \tag{4.12}$$

Note the times of data point x_i being involved in sub-problems $\sum_{j=1}^{s} I(i \in \Phi_{B^j})$ and its times of being selected $\sum_{j=1}^{s} I(i \in \bar{A}^j)$ cannot be confused in the above procedure.

Analogically, when the rank k is unknown and needs to be determined automatically for the incomplete data, a tolerance τ is applied to detect the gap between anchors and non-anchors on their statistics such that

$$A := p_{[l^\star]}, l^\star = \min\left\{l : f(p_l) - f(p_{l+1}) \geq \tau s\right\}. \tag{4.13}$$

In our empirical study, we discover that the number of observable entries in X required by DCA to accurately estimate anchors X_A could be much less than the number of observations required by matrix completion to recover the the matrix. This interesting phenomenon has a heuristic explanation that picking out the indexes of k anchors from $[n]$ should be easier than recovering the exact values of all missing entries. But a more important application of DCA implied by this fact is that we can apply DCA to the highly incomplete rating matrix in recommendation system [163], and find out the representative users. The ratings of other users can then be predicted after collecting ratings only from representative users.

In Algorithm 4, we give a detailed description of the DCA framework, and show how to handle data matrix in noiseless, noisy, complete and incomplete

situations by different strategies.

Algorithm 4: Divide-and-Conquer Anchoring (DCA)

Input: X, rank k or tolerance τ, d, K, parameters for separable NMF SNMF

Output: Index set A of anchors in X

Divide Step (in parallel):;

for $i \leftarrow 1$ **to** s **do**

1) *Complete data*:;

Uniformly sample d random vectors $\{\beta_i\}_{j=1}^{d}$ on the unit hypersphere $\mathbb{S}^{m-1}$, let $B^i = [\beta_1; \beta_2; \cdots; \beta_d]$;

Apply SNMF to $X(B^i)^T$ and let $\bar{A}^i := \text{SNMF}\left(X(B^i)^T\right)$;

2) *Incomplete data*:;

Select a subset Ω_{B^i} of m dimensions, uniformly sample d random vectors $\{\beta_i\}_{j=1}^{d}$ on the unit hypersphere $\mathbb{S}^{|\Omega_{B^i}|-1}$, let $B^i = [\beta_1; \beta_2; \cdots; \beta_d]$;

Apply SNMF to $X_{\Phi_{B^i},\Omega_{B^i}}(B^i)^T$ and let $\bar{A}^i := \text{SNMF}\left(X_{\Phi_{B^i},\Omega_{B^i}}(B^i)^T\right)$, $\Phi_{B^i} = \{i : \text{supp}(x_i) \subseteq \Omega_{B^i}\}$;

end

Conquer Step:;

1) *Noiseless and complete data*:;

$A := \bigcup_{i=1}^{s} \bar{A}^i$;

2) *Noisy and complete data*:;

$A := \arg \max_{H \subseteq [m], |H|=k} \sum_{i \in H} g(i), g(i) = \sum_{j=1}^{s} I(i \in \bar{A}^j)$;

or automatically determine k by

$A := p_{[l^\star]}, l^\star = \min\{l : g(p_l) - g(p_{l+1}) \geq \tau s\}$;

3) *Noisy and incomplete data*:;

$A := \arg \max_{H \subseteq [n], |H|=k} \sum_{i \in H} f(i)$;

$f(i) = \frac{1}{\sum_{j=1}^{s} I(i \in \Phi_{B^j})} \sum_{j=1}^{s} \frac{|\Phi_{B^j}|}{n} I(i \in \bar{A}^j)$;

or automatically determine k by

$A := p_{[l^\star]}, l^\star = \min\{l : f(p_l) - f(p_{l+1}) \geq \tau s\}$;

4.2.4 Analysis: the Number of Sub-problems

We now analyze in what situation the proposed DCA framework is able to identify all the anchors X_A successfully.

Lemma 1. *Let $B = \{\beta_i\}_{i=1}^d$ be d random directions sampled uniformly from $\mathbb{S}^{m-1}$. Assume the index set for the detected anchors on hyperplane defined by B to be $\bar{A}$. Denoted by*

$$p_i^* = \Pr[i \in \bar{A}], \tag{4.14}$$

i.e., p_i^* is the probability for the random projection y_i of data point x_i being identified as an anchor in a sub-problem. Since the detected low-dimensional anchors must be the random projections of the actual anchors (Theorem 6), we have

$$\sum_{i=1}^{k} p_i^* = 1. \tag{4.15}$$

Based on the above lemma, we can provide the following theorem about the minimum number of sub-problems that can guarantee an accurate anchor estimation.

Theorem 7. ***(Identifiability)*** *Suppose $\min_{i\in A} p_i^* \geq \alpha/k > 0$, $i = 1, 2, ..., k$. Given the statistics from the solutions of s sub-problems finding low-dimensional anchors,*

$$g(i) = \sum_{j=1}^{s} I(i \in \bar{A}^j) \tag{4.16}$$

by Theorem 6, clearly we have

$$g(i : i \in A) \geq 0,\ \sum_{i\in A} g(i) = \sum_{i=1}^{s} |\bar{A}^i| \text{ and } g(i : i \notin A) = 0; \tag{4.17}$$

further, it holds with probability at least $1 - k\exp\left(-\frac{\alpha s}{3k}\right)$ that

$$\min_{i\in A} g(i) > 0. \tag{4.18}$$

Proof. For $\forall i \in A$, we introduce a random variable $\xi_t = I(i \in \bar{A}^j)$, then $\Pr(\xi_t =$

1) = p_i^*. By Chernoof bound, we have

$$\Pr\left(\sum_{t=1}^{s}\xi_t < (1-\delta)sp_i^*\right) \leq \exp\left(-\frac{\delta^2 sp_i^*}{2+\delta}\right),\ \forall\delta \geq 0. \tag{4.19}$$

Since $g(i) = \sum_{t=1}^{s}\xi_t$, and let $\delta = 1$, we have

$$\Pr(g(i) = 0) \leq \exp\left(-\frac{sp_i^*}{3}\right). \tag{4.20}$$

This yields

$$\begin{aligned}\Pr(\min_{i\in A} g(i) > 0) &= 1 - \Pr(\cup_{i\in A} g(i) = 0)\\ &\geq 1 - \textstyle\sum_{i\in A}\Pr(g(i) = 0)\\ &\geq 1 - \textstyle\sum_{i\in A}\exp\left(-\frac{sp_i^*}{3}\right)\\ &\geq 1 - k\exp\left(-\frac{s\min_{j\in A} p_j^*}{3}\right).\end{aligned} \tag{4.21}$$

Since $\min_{j\in A} p_j^* \geq \alpha/k$, this completes the proof. □

Therefore, as long as the number of sub-problems $s >> \frac{3}{\alpha}k\log k$, we have $\min_{i\in A} g(i) > 0$, which indicates that all the k vertices can be successfully identified by DCA.

4.3 Rapid Anchoring in 1D or 2D Space

In this section, we consider two extreme cases of DCA, i.e., the sub-problems seek anchors in a 1D or 2D space. Instead of adopting the sophisticated near-separable NMF proposed before, we build two simple and efficient non-iterative algorithms that locate the 1D or 2D anchors after a computation of $\mathcal{O}(n)$ flops. In the framework of DCA, they enjoy a comparable performance as state-of-the-art, but are much more efficient, especially in big data challenge.

4.3.1 Seeking Vertices of Convex Hull in 1D Space

A number of previous works [9, 23] tackle the separable NMF in a special case where all the data points are normalized to have unit ℓ_1 norm and are included in a convex hull. The framework of DCA allows us to divide the original anchoring problem in m-dimensional space to $\mathcal{O}(k \log k)$ sub-problems, each of which aims at finding the anchors in a 1D space. According to Figure 4.1, the convex hull in 1D space is a box on the real axis, and the associated anchors are the two vertices of the box. Therefore, uniformly sample s random vectors $\{\beta_i\}_{i=1}^s$ on the hypersphere $\mathbb{S}^{m-1}$, each sub-problem is reduced to finding the maximum and minimum of the n data projections on each of the s random directions, i.e.,

$$\bar{A}^i := \left\{ \arg\max_j x_j \beta_i^T, \arg\min_j x_j \beta_i^T \right\}, i = 1, 2, \cdots, s. \tag{4.22}$$

When β_i is randomly selected from the set of m unit vectors (only one element is 1 and others are all 0), the above sub-problem can be further simplified to finding the maximum and minimum on a randomly selected dimension. Since finding the maximum or minimum in n values are linear in computation, DCA using the above procedure merely has a time complexity of $\mathcal{O}(nk \log k)$ for single thread computing.

4.3.2 Seeking Extreme Rays of Conical Hull in 2D Space

If we drop the normalization restriction of data points, and generalize the convex hull to a conical hull, the separable NMF problem turns back to the general one we studied in former sections. A simple algorithm can still be built even for this general situation. In particular, we divide the original anchoring problem in m-dimensional space to $\mathcal{O}(k \log k)$ sub-problems, each of which aims at finding the anchors in a 2D space. According to Figure 4.2, the conical hull in 2D space is a region bounded by two rays with initial points at the origin $(0, 0)$, and the associated anchors are the 2D random projections of two data points with the largest angle between their rays. Therefore, in the divide step of DCA, we uniformly sample s pairs of random vectors $\{\beta_{i1}, \beta_{i2}\}_{i=1}^s$ on the hypersphere $\mathbb{S}^{m-1}$, each sub-problem is reduced to finding the maximum and minimum angle

between the 2D random projections of the n data points and the horizontal axis in a 2D plane, whose coordinates are $\{\beta_{i1}, \beta_{i2}\}$, i.e.,

$$\begin{aligned} \bar{A}^i := \big\{ & \arg\max_j \arctan 2\left(x_j\beta_{i1}^T, x_j\beta_{i2}^T\right), \\ & \arg\min_j \arctan 2\left(x_j\beta_{i1}^T, x_j\beta_{i2}^T\right) \big\}, \\ i = 1, 2, \cdots, s. & \end{aligned} \tag{4.23}$$

The function $\arctan 2(\cdot, \cdot)$ is defined as

$$\arctan 2(y, x) = \begin{cases} \arctan(y/x), & x > 0; \\ \arctan(y/x) + \pi, & y \geq 0, x < 0; \\ \arctan(y/x) - \pi, & y < 0, x < 0; \\ \pi/2, & y > 0, x = 0; \\ -\pi/2, & y < 0, x = 0. \end{cases} \tag{4.24}$$

Note that the random vectors $\{\beta_{i1}, \beta_{i2}\}$ can also be selected from the set of unit vectors, so the number of all possible pairs is $m(m-1)/2$. In this case, each sub-problem is reduced to finding the two extreme rays in a 2D space spanned by two randomly selected dimensions. Due to the efficiency of this random selection of dimension pairs, the overall time complexity of DCA adopting the above divide step is also $\mathcal{O}(nk \log k)$ for single thread computing.

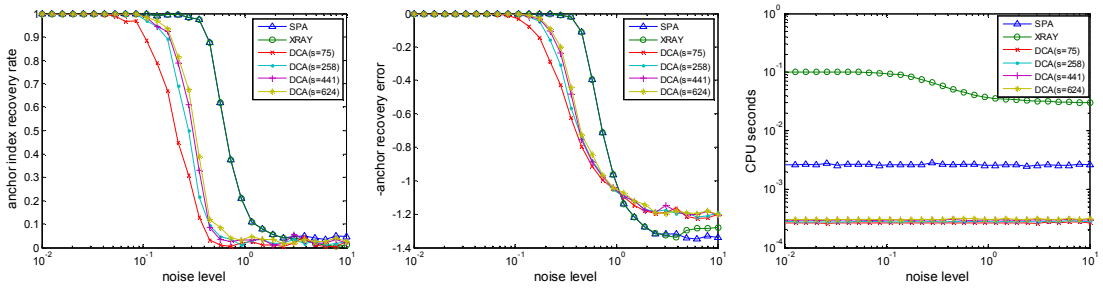

Figure 4.3: Finding the anchors of full observable data points (a complete 300×500 matrix of rank 10) in a conical hull of anchors on 30 noise levels and 4 sub-problem amounts (only for DCA). Each point in the plots is obtained by averaging the results of 20 random trails on 20 different matrices. DCA invoking 2D rapid anchoring in Section 4.3 is compared to SPA [97] and XRAY [145].

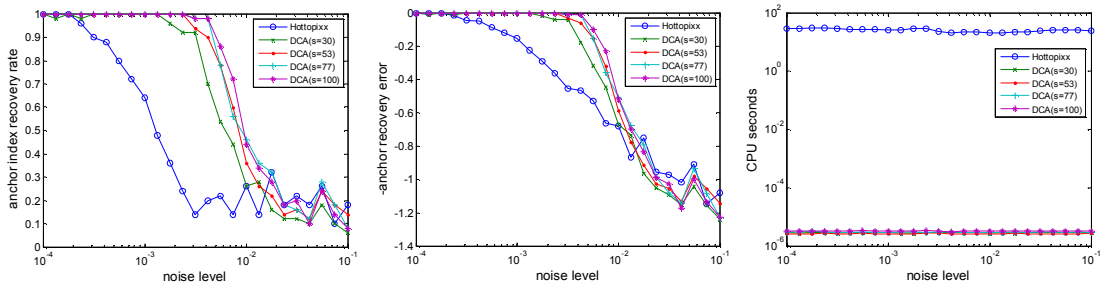

Figure 4.4: Finding the anchors of full observable data points (a complete 50×100 matrix of rank 10 each row is normalized to have unit ℓ_1 norm) in a convex hull of anchors on 25 noise levels and 4 sub-problem amounts (only for DCA). Each point in the plots is obtained by averaging 5 random trails on 5 different matrices. DCA invoking 1D rapid anchoring in Section 4.3 is compared to LP based method Hottopixx [23].

Remark: the above two simple algorithms do not require any iterative computations, and have much smaller time complexity than former near-separable NMF (most have time complexity $\Omega(nm^2)$ or $\mathcal{O}(mnr)$), and are able to achieve comparable accuracy when the data noise is not too big. However, this does not mean former algorithms do not have advantage in DCA. In contrast, they can significantly improve DCA in serious noisy case due to their capability for solving near-separable NMF in higher dimension $|\Omega_B| > 2$. This can be explained by two major reasons 1) much more random hyperplanes can be sampled to produce a more precise and reliable anchor estimation, because $|\Omega_B| > 2$ implies $\binom{m}{|\Omega_B|}$ available hyperplanes; and, 2) more actual anchors can still remain to be anchors after projected onto a hyperplane of higher dimension, and their projections are more easier to be distinguished from those of non-anchors especially in the noisy case.

4.4 Numerical Results

In this section, we report the experimental results of DCA on both synthetic and real datasets, and give thorough comparisons between 3 state-of-the-art near-separable NMF algorithms and DCA. All the experiments were run by MATLAB

on a multi-core workstation, and only single thread computing is applied for fair comparison. CPU second is used to measure the efficiency of all algorithms, and two other metrics are used to evaluate the anchor recovery precision. In particular, for a trial with real anchor index set A and its estimate $\hat{A}$, the anchor index recovery rate ρ is defined as

$$\rho = \frac{|A \cap \hat{A}|}{|A|}, \tag{4.25}$$

and the $-$anchor recovery error η is defined as

$$\eta = -\frac{1}{|A|} \sum_{i \in A} \min_{j \in \hat{A}} \frac{\|x_i - x_j\|_2}{\|x_i\|_2}. \tag{4.26}$$

Larger ρ and η implies more accurate recovery of anchors. The precision of multiple trials are measured by their average ρ and η. Note these two metrics behave differently: a large ρ implies exact localization of anchor indexes, while a large η can cover a sub-optimal case when data points close to the actual anchors are selected as anchors.

4.4.1 Empirical Study on Synthetic Data

Four groups of experiments are conducted on two kinds of synthetic datasets: the first two groups address the problem of finding anchors from full observable (complete) data points in a conical hull and a convex hull, respectively; the rest two groups address the problem of finding anchors from data points with massive missing values and in a conical hull, for varying ratio of observable entries and varying matrix rank, respectively. Each experiment evaluates algorithms on varying noise level, and on varying numbers of sub-problems for DCA. Each evaluation metric reported in the experiment results is the average of multiple trials under the same setting.

The synthetic data matrix in the experiments is generated in the form of $X = FX_A + N$, where $F = [I_k; UV]$, and the entries of noise N are generated by i.i.d. Gaussian $\mathcal{N}(0, \sigma^2)$ with noise level denoted by σ. In the convex hull case, the entries of U and V are generated by i.i.d. uniform distribution in $[0, 1]$ at

first and then their rows are normalized to have unit ℓ_1 norm. In this case, the clear data UV automatically have normalized rows. In the conical hull case, the entries of basis vectors V are generated by i.i.d. uniform distribution in range $[0, 1]$, while each row of weight matrix U are generated according to a Dirichlet distribution whose k parameters are chosen uniformly in $[0, 1]$.

The results of the first two groups of experiments are given in Figure 4.3 and Figure 4.4 respectively, and the results of the rest two groups are shown in Figure 4.5.

In the fully observable data case, DCA performs slightly worse than SPA and XRAY on the same noise level, but outperforms Hottopixx on both recovery rate and error. It can be seen that DCA is able to precisely find out most anchor indexes even when the noise level is close to 1. Although SPA and XRAY can handle a bit more noise, DCA is sufficiently robust for most practical applications. In addition, the anchor recovery error of DCA is equal to or smaller than that of SPA or XRAY. This indicates that DCA can find a data point closer to the actual anchor than the two methods, when the noise is too large to allow any algorithm to find the real anchor. Furthermore, the results show two vital advantages of DCA: 1) very few sub-problems randomly selected from the huge complete set (of size 124750 for Figure 4.3 and 100 for Figure 4.4) are able to produce an accurate solution. This is consistent with our theoretical analysis in Section 4.2.4; and, 2) DCA is much faster than other methods even when it is run by a single thread. It can be expected that a parallel or distributed DCA is able to provide a substantially efficient algorithm that cannot be offered by any previous method for big data challenges.

In the incomplete data case, DCA exhibits remarkable capability and successfully locates most anchors even when 90% entries in the matrix are missing and the rest observed ones are corrupted by relatively large noise. Note there are merely 125 sub-problems involved in the divide step of DCA. This indicates that DCA does not use all the information encoded in the observed entries, and can still achieve promising performance. Compared to similar plots reported in matrix completion (MC) literatures [258][39], we further find out that the sampling ratio required by DCA to recover anchors are much less than that required by MC to recover the matrix. But their goals in massive applications are identical, i.e.,

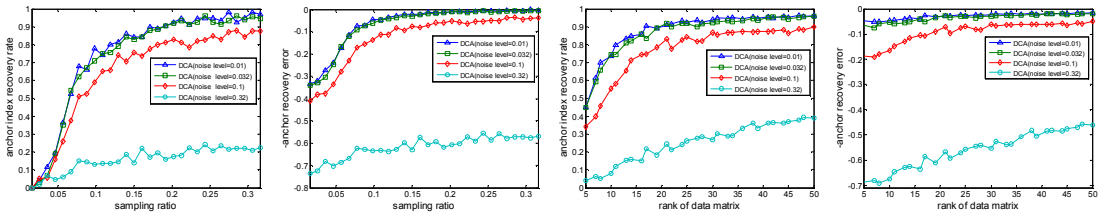

Figure 4.5: Finding the anchors of data points with massive missing values (an incomplete 50×100 matrix each entry is observed with probability *sampling ratio*) in a conical hull of anchors via solving 125 sub-problems by DCA on 4 noise levels. The left two plots show the results when sampling ratio varies between $[0.01, 0.31]$ and the rank k is fixed to 10, while the two plots on the right show the results when rank k varies between $[5, 50]$ and the sampling ratio is fixed to 0.15. Each point in the plots is obtained by averaging 20 random trails on 20 different matrices. The divide step of DCA uses 2D rapid anchoring in Section 4.3.

to represent a huge number of data points by a few basis. This means DCA can replace MC in various situations (analogous to the relation between PCA [119] and CUR [75]). Another interesting phenomenon of DCA shown in Figure 4.5 is that its performance improves with the increase of rank, which is opposite to what we usually observed in MC. This is interpretable, because to select a fewer anchors from a great number of data points and ensure that they can express the whole dataset is much difficult than to pick out more redundant data points, which can easily lead to a small reconstruction error. Hence more information from data is required in the former case.

4.4.2 Collaborative Filtering by Finding Representative Users

We show an interesting application of DCA in recommendation systems, which were mentioned in Section 4.2.3 and cannot be handled by other separable NMF. In particular, we assume that there exist some representative users whose ratings' conical combination is able to express all the other users' ratings. This assumption generates an extremely efficient method to predict unknown ratings in the user-item rating matrix, that is, we can detect the representative users at first and only collect ratings from them. Then we can predict the other users' missing ratings

by the ratings of these users. This problem can be rapidly solved by DCA for incomplete data, in which the rating matrix is X and the ratings of representative users are the anchors X_A.

We apply this method to MovieLens [1] dataset, which includes 100k, 1M and 10M ratings to movies in its three subsets, and compare its reconstruction error and speed with state-of-the-art matrix completion algorithm GreB [258]. Since the ratings of the detected representative users cannot be known from the real dataset, we assign the results of matrix completion to them. The results in Table 4.1 show that DCA outperforms GreB on both prediction accuracy and computational efficiency.

Table 4.1: Normalized mean absolute error (NMAE), root mean square error (RMSE) and CPU seconds of DCA and matrix completion on MovieLens. $n/m/k$ of 3 datasets: 100k(943/1682/10), 1M(6040/3952/10), 10M(69878/10677/10). Result format: NMAE/RMSE/CPU seconds.

	GreB	DCA
100k	0.15/0.94/0.57	0.11/0.87/0.12
1M	0.12/0.86/3.54	0.09/0.80/1.36
10M	0.11/1.04/20.67	0.08/0.95/5.98

4.4.3 Reconstruction of Images, Texts and Handwritten Digits

A primary goal of separable NMF is to express the whole dataset by the extracted anchors. We apply SPCA, XRAY and DCA (conical hull case) to three real datasets, i.e., Grolier encyclopedia article dataset (15276 words' counts in 30991 articles) [2], and MIT scene dataset with gist features (2688 images with 512 features) [3], and MNIST dataset of handwritten digits (70000 28×28 images for digits from 0 to 9) [4]. We do not evaluate LP based methods here due to its large

[1] http://www.grouplens.org/node/73
[2] http://www.cs.nyu.edu/ roweis/data.html
[3] http://people.csail.mit.edu/torralba/code/spatialenvelope/
[4] http://yann.lecun.com/exdb/mnist/

Table 4.2: Reconstruction error and CPU seconds of SPA, XRAY and DCA on three datasets. The rank k for reconstructing them is $30, 50, 50$. Result format: ℓ_2 error/CPU seconds.

	SPA	XRAY	DCA
MIT scene	0.3351/0.085	0.3133/104.19	0.2766/0.038
MNIST	569.21/1.891	0.6128/233.69	0.5792/0.928
Grolier	0.5041/3.777	0.3589/17.89	0.2907/1.303

time costs on these large-scale datasets. In each experiment, after obtaining the anchors X_A, a nonnegative least square regression is used to calculate the weight matrix F' from X_A and X. We show the average normalized ℓ_2 reconstruction error $\|x_i - \hat{x}_i\|_2/\|x_i\|_2$ and CPU seconds of different methods in Table 4.2. DCA achieves the best performance on both accuracy and efficiency in most trials.

Chapter 5

Randomized and Greedy Strategies for Bilateral Low-rank Approximation

5.1 Bilateral Random Projections (BRP)

Low-rank structure has been profoundly studied in data mining and machine learning. In this chapter, we show a dense matrix X's low-rank approximation can be rapidly built from its left and right random projections $Y_1 = XA_1$ and $Y_2 = X^T A_2$, or bilateral random projection (BRP). We then show power scheme can further improve the precision. The deterministic, average and deviation bounds of the proposed method and its power scheme modification are proved theoretically. The effectiveness and the efficiency of BRP based low-rank approximation is empirically verified on both artificial and real datasets.

5.1.1 Introduction

Recent research about low-rank structure concentrate on developing fast approximation and building meaningful decompositions. Two appealing representatives are the randomized approximate matrix decomposition [108] and column selection [65]. The former proves that a matrix can be well approximated by its projection to the column space of its random projections. This rank-revealing method pro-

vides a fast approximation of SVD/PCA. The latter proves that a column subset of a low-rank matrix can span its whole range. The low-rank approximation based on random projections for streaming data is also studied in [53].

In this chapter, we consider the problem of fast low-rank approximation. Given r bilateral random projections (BRP) of an $m \times n$ dense matrix X (w.l.o.g, $m \geq n$), i.e., $Y_1 = XA_1$ and $Y_2 = X^T A_2$, wherein $A_1 \in \mathbb{R}^{n\times r}$ and $A_2 \in \mathbb{R}^{m\times r}$ are random matrices,

$$L = Y_1 \left(A_2^T Y_1\right)^{-1} Y_2^T \tag{5.1}$$

is a fast rank-r approximation of X. The computation of L includes an inverse of an $r \times r$ matrix and three matrix multiplications. Thus, for a dense X, $2mnr$ floating-point operations (flops) are required to obtain BRP, $r^2(2n+r)+mnr$ flops are required to obtain L. The computational cost is much less than SVD based approximation. The L in (5.1) has been proposed in [85] as a recovery of a rank-r matrix X from Y_1 and Y_2, where A_1 and A_2 are independent Gaussian/SRFT random matrices. However, we propose that L is a tight rank-r approximation of a full rank matrix X, when A_1 and A_2 are correlated random matrices updated from Y_2 and Y_1, respectively. We then apply power scheme [195] to L for improving the approximation precision, especially when the eigenvalues of X decay slowly.

Theoretically, we prove the deterministic bound, average bound and deviation bound of the approximation error in BRP based low-rank approximation and its power scheme modification. The results show the error of BRP based approximation is close to the error of SVD approximation under mild conditions. Comparing with randomized SVD in [108] that extracts the column space from unilateral random projections, the BRP based method estimates both column and row spaces from bilateral random projections.

We give an empirical study of BRP on both artificial data and face image dataset. The results show its effectiveness and efficiency in low-rank approximation and recovery.

5.1.2 Bilateral random projections (BRP) based low-rank approximation

We first introduce the bilateral random projections (BRP) based low-rank approximation and its power scheme modification. The approximation error bounds of these two methods are discussed at the end of this section.

5.1.2.1 Low-rank approximation with closed form

In order to improve the approximation precision of L in (5.1) when A_1 and A_2 are standard Gaussian matrices, we use the obtained right random projection Y_1 to build a better left projection matrix A_2, and use Y_2 to build a better A_1. In particular, after $Y_1 = XA_1$, we update $A_2 = Y_1$ and calculate the left random projection $Y_2 = X^T A_2$, then we update $A_1 = Y_2$ and calculate the right random projection $Y_1 = XA_1$. A better low-rank approximation L will be obtained if the new Y_1 and Y_2 are applied to (5.1). This improvement requires additional flops of mnr in BRP calculation.

5.1.2.2 Power scheme modification

When singular values of X decay slowly, (5.1) may perform poorly. We design a modification for this situation based on the power scheme [195]. In the power scheme modification, we instead calculate the BRP of a matrix $\tilde{X} = (XX^T)^q X$, whose singular values decay faster than X. In particular, $\lambda_i(\tilde{X}) = \lambda_i(\tilde{X})^{2q+1}$. Both X and $\tilde{X}$ share the same singular vectors. The BRP of $\tilde{X}$ is:

$$Y_1 = \tilde{X}A_1, Y_2 = \tilde{X}^T A_2. \tag{5.2}$$

According to (5.1), the BRP based r rank approximation of $\tilde{X}$ is:

$$\tilde{L} = Y_1 \left(A_2^T Y_1\right)^{-1} Y_2^T. \tag{5.3}$$

In order to obtain the approximation of X with rank r, we calculate the QR decomposition of Y_1 and Y_2, i.e.,

$$Y_1 = Q_1 R_1, Y_2 = Q_2 R_2. \tag{5.4}$$

The low-rank approximation of X is then given by:

$$L = \left(\tilde{L}\right)^{\frac{1}{2q+1}} = Q_1 \left[R_1 \left(A_2^T Y_1\right)^{-1} R_2^T\right]^{\frac{1}{2q+1}} Q_2^T. \tag{5.5}$$

The power scheme modification (5.5) requires an inverse of an $r \times r$ matrix, an SVD of an $r \times r$ matrix and five matrix multiplications. Therefore, for dense X, $2(2q+1)mnr$ flops are required to obtain BRP, $r^2(m+n)$ flops are required to obtain the QR decompositions, $2r^2(n+2r)+mnr$ flops are required to obtain L. The power scheme modification reduces the error of (5.1) by increasing q. When the random matrices A_1 and A_2 are built from Y_1 and Y_2, mnr additional flops are required in the BRP calculation.

5.1.3 Approximation error bounds

We analyze the error bounds of the BRP based low-rank approximation (5.1) and its power scheme modification (5.5).

The SVD of an $m \times n$ (w.l.o.g, $m \geq n$) matrix X takes the form:

$$X = U\Lambda V^T = U_1 \Lambda_1 V_1^T + U_2 \Lambda_2 V_2^T, \tag{5.6}$$

where Λ_1 is an $r \times r$ diagonal matrix which diagonal elements are the first largest r singular values, U_1 and V_1 are the corresponding singular vectors, Λ_2, U_2 and V_2 forms the rest part of SVD. Assume that r is the target rank, A_1 and A_2 have $r+p$ columns for oversampling. We consider the spectral norm of the approximation error E for (5.1):

$$\begin{aligned} \|X - L\| &= \left\| X - Y_1 \left(A_2^T Y_1\right)^{-1} Y_2^T \right\| \\ &= \left\| \left[I - XA_1 \left(A_2^T X A_1\right)^{-1} A_2^T\right] X \right\|. \end{aligned} \tag{5.7}$$

The unitary invariance of the spectral norm leads to

$$\begin{aligned}\|X - L\| &= \left\| U^T \left[I - XA_1 \left(A_2^T X A_1 \right)^{-1} A_2^T \right] X \right\| \\ &= \left\| \Lambda \left[I - V^T A_1 \left(A_2^T X A_1 \right)^{-1} A_2^T U \Lambda \right] \right\|. \end{aligned} \tag{5.8}$$

In low-rank approximation, the left random projection matrix A_2 is built from the left random projection $Y_1 = XA_1$, and then the right random projection matrix A_1 is built from the left random projection $Y_2 = X^T A_2$. Thus $A_2 = Y_1 = XA_1 = U\Lambda V^T A_1$ and $A_1 = Y_2 = X^T A_2 = X^T X A_1 = V\Lambda^2 V^T A_1$. Hence the approximation error given in (5.8) has the following form:

$$\left\| \Lambda \left[I - \Lambda^2 V^T A_1 \left(A_1^T V \Lambda^4 V^T A_1 \right)^{-1} A_1^T V \Lambda^2 \right] \right\|. \tag{5.9}$$

The following Theorem 8 gives the bound for the spectral norm of the deterministic error $\|X - L\|$.

Theorem 8. (Deterministic error bound) *Given an $m \times n$ $(m \geq n)$ real matrix X with singular value decomposition $X = U\Lambda V^T = U_1 \Lambda_1 V_1^T + U_2 \Lambda_2 V_2^T$, and chosen a target rank $r \leq n - 1$ and an $n \times (r + p)$ $(p \geq 2)$ standard Gaussian matrix A_1, the BRP based low-rank approximation (5.1) approximates X with the error upper bounded by*

$$\|X - L\|^2 \leq \left\| \Lambda_2^2 \left(V_2^T A_1 \right) \left(V_1^T A_1 \right)^{\dagger} \Lambda_1^{-1} \right\|^2 + \|\Lambda_2\|^2 .$$

See Section 5.1.4 for the proof of Theorem 8.

If the singular values of X decay fast, the first term in the deterministic error bound will be very small. The last term is the rank-r SVD approximation error. Therefore, the BRP based low-rank approximation (5.1) is nearly optimal.

Theorem 9. (Deterministic error bound, power scheme) *Frame the hypotheses of Theorem 8, the power scheme modification (5.5) approximates X with*

the error upper bounded by

$$\|X - L\|^2 \leq \left(\left\| \Lambda_2^{2(2q+1)} \left(V_2^T A_1 \right) \left(V_1^T A_1 \right)^{\dagger} \Lambda_1^{-(2q+1)} \right\|^2 + \left\| \Lambda_2^{2q+1} \right\|^2 \right)^{1/(2q+1)}.$$

See Section 5.1.4 for the proof of Theorem 9.

If the singular values of X decay slowly, the error produced by the power scheme modification (5.5) is less than the BRP based low-rank approximation (5.1) and decreasing with the increasing of q.

The average error bound of BRP based low-rank approximation is obtained by analyzing the statistical properties of the random matrices that appear in the deterministic error bound in Theorem 8.

Theorem 10. (Average error bound) *Frame the hypotheses of Theorem 8,*

$$\mathbb{E}\|X - L\| \leq \left(\sqrt{\frac{1}{p-1} \sum_{i=1}^{r} \frac{\lambda_{r+1}^2}{\lambda_i^2}} + 1 \right) |\lambda_{r+1}| + \frac{\mathrm{e}\sqrt{r+p}}{p} \sqrt{\sum_{i=r+1}^{n} \frac{\lambda_i^2}{\lambda_r^2}}.$$

See Section 5.1.4 for the proof of Theorem 10.

The average error bound will approach to the SVD approximation error $|\lambda_{r+1}|$ if $|\lambda_{r+1}| \ll |\lambda_{i:i=1,\cdots,r}|$ and $|\lambda_r| \gg |\lambda_{i:i=r+1,\cdots,n}|$.

The average error bound for the power scheme modification is then obtained from the result of Theorem 10.

Theorem 11. (**Average error bound, power scheme**) *Frame the hypotheses of Theorem 8, the power scheme modification (5.5) approximates X with the*

expected error upper bounded by

$$E\|X-L\| \leq \left[\left(\sqrt{\frac{1}{p-1}\sum_{i=1}^{r}\frac{\lambda_{r+1}^{2(2q+1)}}{\lambda_i^{2(2q+1)}}}+1\right)|\lambda_{r+1}^{2q+1}| + \frac{\mathrm{e}\sqrt{r+p}}{p}\sqrt{\sum_{i=r+1}^{n}\frac{\lambda_i^{2(2q+1)}}{\lambda_r^{2(2q+1)}}}\right]^{1/(2q+1)}.$$

See Section 5.1.4 for the proof of Theorem 11.

Compared the average error bounds of the BRP based low-rank approximation with its power scheme modification, the latter produces less error than the former, and the error can be further decreased by increasing q.

The deviation bound for the spectral norm of the approximation error can be obtained by analyzing the deviation bound of $\left\|\Lambda_2^2\left(V_2^TA_1\right)\left(V_1^TA_1\right)^{\dagger}\Lambda_1^{-1}\right\|$ in the deterministic error bound and by applying the concentration inequality for Lipschitz functions of a Gaussian matrix.

Theorem 12. (**Deviation bound**) *Frame the hypotheses of Theorem 8. Assume that $p \geq 4$. For all $u, t \geq 1$, it holds that*

$$\|X-L\| \leq \left(1+t\sqrt{\frac{12r}{p}}\left(\sum_{i=1}^{r}\lambda_i^{-1}\right)^{\frac{1}{2}}+\frac{\mathrm{e}\sqrt{r+p}}{p+1}\cdot tu\lambda_r^{-1}\right)\lambda_{r+1}^2+\frac{\mathrm{e}\sqrt{r+p}}{p+1}\cdot t\lambda_r^{-1}\left(\sum_{i=r+1}^{n}\lambda_i^2\right)^{\frac{1}{2}}.$$

except with probability $\mathrm{e}^{-u^2/2}+4t^{-p}+t^{-(p+1)}$.

See Section 5.1.4 for the proof of Theorem 12.

5.1.4 Proofs of error bounds

5.1.4.1 Proof of Theorem 8

The following lemma and propositions from [108] will be used in the proof.

Lemma 2. *Suppose that* $M \succeq 0$. *For every* A, *the matrix* $A^T M A \succeq 0$. *In particular,*

$$M \preceq N \quad \Rightarrow \quad A^T M A \preceq A^T N A. \tag{5.10}$$

Proposition 1. *Suppose* $\mathrm{range}(N) \subset \mathrm{range}(M)$. *Then, for each matrix* A, *it holds that* $\|\mathcal{P}_N A\| \leq \|\mathcal{P}_M A\|$ *and that* $\|(I - \mathcal{P}_M)A\| \leq \|(I - \mathcal{P}_N)A\|$.

Proposition 2. *Suppose that* $M \succeq 0$. *Then*

$$I - (I + M)^{-1} \preceq M. \tag{5.11}$$

Proposition 3. *We have* $\|M\| \leq \|A\| + \|C\|$ *for each partitioned positive semidefinite matrix*

$$M = \begin{bmatrix} A & B \\ B^T & C \end{bmatrix}. \tag{5.12}$$

The proof of Theorem 8 is given below.

Proof. Since an orthogonal projector projects a given matrix to the range (column space) of a matrix M is defined as $\mathcal{P}_M = M(M^T M)^{-1} M^T$, the deterministic error (5.9) can be written as

$$\|E\| = \left\|\Lambda \left(I - \mathcal{P}_M\right)\right\|, \; M = \Lambda^2 V^T A_1. \tag{5.13}$$

By applying Proposition 1 to the error (5.13), because $\mathrm{range}(M(V_1^T A_1)^\dagger \Lambda_1^{-2}) \subset \mathrm{range}(M)$, we have

$$\|E\| = \left\|\Lambda \left(I - \mathcal{P}_M\right)\right\| \leq \left\|\Lambda \left(I - \mathcal{P}_N\right)\right\|, \tag{5.14}$$

where

$$N = \begin{bmatrix} \Lambda_1^2 V_1^T A_1 \\ \Lambda_2^2 V_2^T A_1 \end{bmatrix} (V_1^T A_1)^\dagger \Lambda_1^{-2} = \begin{bmatrix} I \\ H \end{bmatrix}. \tag{5.15}$$

Thus $(I - \mathcal{P}_N)$ can be written as

$$I - \mathcal{P}_N = \begin{bmatrix} I - \left(I + H^T H\right)^{-1} & -\left(I + H^T H\right)^{-1} H^T \\ -H\left(I + H^T H\right)^{-1} & I - H\left(I + H^T H\right)^{-1} H^T \end{bmatrix}$$

For the top-left block in (5.16), Proposition 2 leads to $I - \left(I + H^T H\right)^{-1} \preceq H^T H$. For the bottom-right block in (5.16), Lemma 2 leads to $I - H\left(I + H^T H\right)^{-1} H^T \preceq I$. Therefore,

$$I - \mathcal{P}_N \preceq \begin{bmatrix} H^T H & -\left(I + H^T H\right)^{-1} H^T \\ -H\left(I + H^T H\right)^{-1} & I \end{bmatrix}$$

By applying Lemma 2, we have

$$\Lambda\left(I - \mathcal{P}_N\right)\Lambda \preceq \begin{bmatrix} \Lambda_1^T H^T H \Lambda_1 & -\Lambda_1^T \left(I + H^T H\right)^{-1} H^T \Lambda_2 \\ -\Lambda_2^T H \left(I + H^T H\right)^{-1} \Lambda_1 & \Lambda_2^T \Lambda_2 \end{bmatrix}$$

According to Proposition 3, the spectral norm of $\Lambda(I - \mathcal{P}_N)$ is bounded by

$$\begin{aligned} \left\|\Lambda\left(I - \mathcal{P}_N\right)\right\|^2 &= \left\|\Lambda\left(I - \mathcal{P}_N\right)\Lambda\right\| \\ &\leq \left\|\Lambda_2^2 \left(V_2^T A_1\right)\left(V_1^T A_1\right)^{\dagger} \Lambda_1^{-1}\right\|^2 + \left\|\Lambda_2\right\|^2. \end{aligned} \tag{5.16}$$

By substituting (5.16) into (5.14), we obtain the deterministic error bound. This completes the proof. □

5.1.4.2 Proof of Theorem 9

The following proposition from [108] will be used in the proof.

Proposition 4. *Let $\mathcal{P}$ be an orthogonal projector, and let A be a matrix. For each nonnegative q,*

$$\left\|\mathcal{P}A\right\| \leq \left\|\mathcal{P}\left(AA^T\right)^q A\right\|^{1/(2q+1)}. \tag{5.17}$$

The proof of Theorem 9 is given below.

Proof. The power scheme modification (5.5) applies the BRP based low-rank approximation (5.1) to $\tilde{X} = (XX^T)^q X = U\Lambda^{2q+1}V^T$ rather than X. In this case,

the approximation error is

$$\|\tilde{X} - \tilde{L}\| = \left\|\Lambda^{2q+1}\left(I - \mathcal{P}_M\right)\right\|, \; M = \Lambda^{2(2q+1)}V^T A_1. \tag{5.18}$$

According to Theorem 8, the error is upper bounded by

$$\begin{aligned}&\left\|\tilde{X} - \tilde{L}\right\|^2 \leq \\ &\left\|\Lambda_2^{2(2q+1)}\left(V_2^T A_1\right)\left(V_1^T A_1\right)^{\dagger}\Lambda_1^{-(2q+1)}\right\|^2 + \left\|\Lambda_2^{2q+1}\right\|^2.\end{aligned} \tag{5.19}$$

The deterministic error bound for the power scheme modification is obtained by applying Proposition 4 to (5.19). This completes the proof. □

5.1.4.3 Proof of Theorem 10

The following propositions from [108] will be used in the proof.

Proposition 5. *Fix matrices S, T, and draw a standard Gaussian matrix G. Then it holds that*

$$\mathbb{E}\left\|SGT^T\right\| \leq \|S\|\|T\|_F + \|S\|_F\|T\|. \tag{5.20}$$

Proposition 6. *Draw an $r \times (r+p)$ standard Gaussian matrix G with $p \geq 2$. Then it holds that*

$$\mathbb{E}\|G^{\dagger}\|_F^2 = \frac{r}{p-1}, \mathbb{E}\|G^{\dagger}\| \leq \frac{\mathrm{e}\sqrt{r+p}}{p}. \tag{5.21}$$

The proof of Theorem 10 is given below.

Proof. The distribution of a standard Gaussian matrix is rotational invariant. Since 1) A_1 is a standard Gaussian matrix and 2) V is an orthogonal matrix, $V^T A_1$ is a standard Gaussian matrix, and its disjoint submatrices $V_1^T A_1$ and $V_2^T A_1$ are standard Gaussian matrices as well.

Theorem 8 and the Hölder's inequality imply that

$$\begin{aligned}\mathbb{E}\|X-L\| &\leq \mathbb{E}\left(\left\|\Lambda_2^2\left(V_2^TA_1\right)\left(V_1^TA_1\right)^\dagger\Lambda_1^{-1}\right\|^2+\|\Lambda_2\|^2\right)^{1/2}\\ &\leq \mathbb{E}\left\|\Lambda_2^2\left(V_2^TA_1\right)\left(V_1^TA_1\right)^\dagger\Lambda_1^{-1}\right\|+\|\Lambda_2\|. \end{aligned} \tag{5.22}$$

We condition on $V_1^TA_1$ and apply Proposition 5 to bound the expectation w.r.t. $V_2^TA_1$, i.e.,

$$\begin{aligned}&E\left\|\Lambda_2^2\left(V_2^TA_1\right)\left(V_1^TA_1\right)^\dagger\Lambda_1^{-1}\right\|\\ &\leq \mathbb{E}\left(\left\|\Lambda_2^2\right\|\left\|\left(V_1^TA_1\right)^\dagger\Lambda_1^{-1}\right\|_F+\left\|\Lambda_2^2\right\|_F\left\|\left(V_1^TA_1\right)^\dagger\Lambda_1^{-1}\right\|\right)\\ &\leq \left\|\Lambda_2^2\right\|\left(\mathbb{E}\left\|\left(V_1^TA_1\right)^\dagger\Lambda_1^{-1}\right\|_F^2\right)^{1/2}+\\ &\left\|\Lambda_2^2\right\|_F\cdot\mathbb{E}\left\|\left(V_1^TA_1\right)^\dagger\right\|\cdot\left\|\Lambda_1^{-1}\right\|. \end{aligned} \tag{5.23}$$

The Frobenius norm of $(V_1^TA_1)^\dagger\Lambda_1^{-1}$ can be calculated as

$$\begin{aligned}\left\|\left(V_1^TA_1\right)^\dagger\Lambda_1^{-1}\right\|_F^2 &= \text{trace}\left[\Lambda_1^{-1}\left(\left(V_1^TA_1\right)^\dagger\right)^T\left(V_1^TA_1\right)^\dagger\Lambda_1^{-1}\right]\\ &= \text{trace}\left[\left(\left(\Lambda_1V_1^TA_1\right)\left(\Lambda_1V_1^TA_1\right)^T\right)^{-1}\right].\end{aligned}$$

Since 1) $V_1^TA_1$ is a standard Gaussian matrix and 2) Λ_1 is a diagonal matrix, each column of $\Lambda_1V_1^TA_1$ follows r-variate Gaussian distribution $\mathcal{N}_r(\mathbf{0},\Lambda_1^2)$. Thus the random matrix $\left(\left(\Lambda_1V_1^TA_1\right)\left(\Lambda_1V_1^TA_1\right)^T\right)^{-1}$ follows the inverted Wishart distribution $\mathcal{W}_r^{-1}(\Lambda_1^{-2},r+p)$. According to the expectation of inverted Wishart distribution [177], we have

$$\begin{aligned}&\mathbb{E}\left\|\left(V_1^TA_1\right)^\dagger\Lambda_1^{-1}\right\|_F^2\\ &= \mathbb{E}\ \text{trace}\left[\left(\left(\Lambda_1V_1^TA_1\right)\left(\Lambda_1V_1^TA_1\right)^T\right)^{-1}\right]\\ &= \text{trace}\ \mathbb{E}\left[\left(\left(\Lambda_1V_1^TA_1\right)\left(\Lambda_1V_1^TA_1\right)^T\right)^{-1}\right]\\ &= \frac{1}{p-1}\sum_{i=1}^r\lambda_i^{-2}. \end{aligned} \tag{5.24}$$

We apply Proposition 6 to the standard Gaussian matrix $V_1^T A_1$ and obtain

$$\mathbb{E}\left\|(V_1^T A_1)^\dagger\right\| \leq \frac{\mathrm{e}\sqrt{r+p}}{p}. \tag{5.25}$$

Therefore, (5.23) can be further derived as

$$\begin{aligned} & E\left\|\Lambda_2^2\left(V_2^T A_1\right)(V_1^T A_1)^\dagger \Lambda_1^{-1}\right\| \\ & \leq \lambda_{r+1}^2 \cdot \sqrt{\frac{1}{p-1}\sum_{i=1}^{r}\lambda_i^{-2}} + \sqrt{\sum_{i=r+1}^{n}\lambda_i^2}\cdot\frac{\mathrm{e}\sqrt{r+p}}{p}\cdot|\lambda_r^{-1}| \\ & = |\lambda_{r+1}|\sqrt{\frac{1}{p-1}\sum_{i=1}^{r}\frac{\lambda_{r+1}^2}{\lambda_i^2}} + \frac{\mathrm{e}\sqrt{r+p}}{p}\sqrt{\sum_{i=r+1}^{n}\frac{\lambda_i^2}{\lambda_r^2}}. \end{aligned} \tag{5.26}$$

By substituting (5.26) into (5.22), we obtain the average error bound

$$\begin{aligned} \mathbb{E}\|X-L\| \leq & \left(\sqrt{\frac{1}{p-1}\sum_{i=1}^{r}\frac{\lambda_{r+1}^2}{\lambda_i^2}}+1\right)|\lambda_{r+1}|+ \\ & \frac{\mathrm{e}\sqrt{r+p}}{p}\sqrt{\sum_{i=r+1}^{n}\frac{\lambda_i^2}{\lambda_r^2}}. \end{aligned} \tag{5.27}$$

This completes the proof. □

5.1.4.4 Proof of Theorem 11

The proof of Theorem 11 is given below.

Proof. By using Hölder's inequality and Theorem 9, we have

$$\begin{aligned} \mathbb{E}\|X-L\| & \leq \left(\mathbb{E}\|X-L\|^{2q+1}\right)^{1/(2q+1)} \\ & \leq \left(\mathbb{E}\left\|\tilde{X}-\tilde{L}\right\|\right)^{1/(2q+1)}. \end{aligned} \tag{5.28}$$

We apply Theorem 10 to $\tilde{X}$ and $\tilde{L}$ and obtain the bound of $\mathbb{E}\|\tilde{X}-\tilde{L}\|$, noting

that $\lambda_i(\tilde{X}) = \lambda_i(X)^{2q+1}$.

$$\mathbb{E}\left\|\tilde{X} - \tilde{L}\right\| = \left(\sqrt{\frac{1}{p-1}\sum_{i=1}^{r}\frac{\lambda_{r+1}^{2(2q+1)}}{\lambda_i^{2(2q+1)}}} + 1\right)|\lambda_{r+1}^{2q+1}| + \frac{\mathrm{e}\sqrt{r+p}}{p}\sqrt{\sum_{i=r+1}^{n}\frac{\lambda_i^{2(2q+1)}}{\lambda_r^{2(2q+1)}}}. \tag{5.29}$$

By substituting (5.29) into (5.28), we obtain the average error bound of the power scheme modification shown in Theorem 11. This completes the proof. □

5.1.4.5 Proof of Theorem 12

The following propositions from [108] will be used in the proof.

Proposition 7. *Suppose that h is a Lipschitz function on matrices:*

$$|h(X) - h(Y)| \leq L\|X - F\|_F \quad \textit{for all } X, Y. \tag{5.30}$$

Draw a standard Gaussian matrix G. Then

$$\Pr\left\{h(G) \geq \mathbb{E}h(G) + Lt\right\} \leq \mathrm{e}^{-t^2/2}. \tag{5.31}$$

Proposition 8. *Let G be a $r \times (r+p)$ standard Gaussian matrix where $p \geq 4$. For all $t \geq 1$,*

$$\Pr\left\{\left\|G^\dagger\right\|_F \geq \sqrt{\frac{12r}{p}} \cdot t\right\} \leq 4t^{-p} \quad \text{and}$$
$$\Pr\left\{\left\|G^\dagger\right\| \geq \frac{\mathrm{e}\sqrt{r+p}}{p+1} \cdot t\right\} \leq t^{-(p+1)}. \tag{5.32}$$

The proof of Theorem 12 is given below.

Proof. According to the deterministic error bound in Theorem 8, we study the deviation of $\left\|\Lambda_2^2\left(V_2^T A_1\right)\left(V_1^T A_1\right)^\dagger \Lambda_1^{-1}\right\|$. Consider the Lipschitz function $h(X) = \left\|\Lambda_2^2 X\left(V_1^T A_1\right)^\dagger \Lambda_1^{-1}\right\|$, its Lipschitz constant L can be estimated by using the

triangle inequality:

$$\begin{aligned}
|h(X) - h(Y)| &\leq \left\| \Lambda_2^2 \left(X - Y\right) \left(V_1^T A_1\right)^\dagger \Lambda_1^{-1} \right\| \\
&\leq \left\|\Lambda_2^2\right\| \|X - Y\| \left\| \left(V_1^T A_1\right)^\dagger \right\| \left\|\Lambda_1^{-1}\right\| \\
&\leq \left\|\Lambda_2^2\right\| \left\| \left(V_1^T A_1\right)^\dagger \right\| \left\|\Lambda_1^{-1}\right\| \|X - Y\|_F \,. \qquad (5.33)
\end{aligned}$$

Hence the Lipschitz constant satisfies $L \leq \left\|\Lambda_2^2\right\| \left\| \left(V_1^T A_1\right)^\dagger \right\| \left\|\Lambda_1^{-1}\right\|$. We condition on $V_1^T A_1$ and then Proposition 5 implies that

$$\begin{aligned}
\mathbb{E}\left[h\left(V_2^T A_1\right) \mid V_1^T A_1\right] \leq &\left\|\Lambda_2^2\right\| \left\| \left(V_1^T A_1\right)^\dagger \right\|_F \left\|\Lambda_1^{-1}\right\|_F + \\
&\left\|\Lambda_2^2\right\|_F \left\| \left(V_1^T A_1\right)^\dagger \right\| \left\|\Lambda_1^{-1}\right\| .
\end{aligned}$$

We define an event T as

$$\begin{aligned}
T = \Bigg\{ &\left\| \left(V_1^T A_1\right)^\dagger \right\|_F \leq \sqrt{\frac{12r}{p}} \cdot t \quad \text{and} \\
&\left\| \left(V_1^T A_1\right)^\dagger \right\| \leq \frac{\mathrm{e}\sqrt{r+p}}{p+1} \cdot t \Bigg\} . \qquad (5.34)
\end{aligned}$$

According to Proposition 8, the event T happens except with probability

$$\Pr\left\{\overline{T}\right\} \leq 4t^{-p} + t^{-(p+1)}. \qquad (5.35)$$

Applying Proposition 7 to the function $h\left(V_2^T A_1\right)$, given the event T, we have

$$\begin{aligned}
\Pr\Big\{ &\left\| \Lambda_2^2 \left(V_2^T A_1\right) \left(V_1^T A_1\right)^\dagger \Lambda_1^{-1} \right\| > \\
&\left\|\Lambda_2^2\right\| \left\| \left(V_1^T A_1\right)^\dagger \right\|_F \left\|\Lambda_1^{-1}\right\|_F + \\
&\left\|\Lambda_2^2\right\|_F \left\| \left(V_1^T A_1\right)^\dagger \right\| \left\|\Lambda_1^{-1}\right\| + \\
&\left\|\Lambda_2^2\right\| \left\| \left(V_1^T A_1\right)^\dagger \right\| \left\|\Lambda_1^{-1}\right\| \cdot u \mid T \Big\} \leq \mathrm{e}^{-u^2/2}. \qquad (5.36)
\end{aligned}$$

According to the definition of the event T and the probability of $\overline{T}$, we obtain

$$\begin{aligned}
&\Pr\left\{\left\|\Lambda_2^2\left(V_2^TA_1\right)\left(V_1^TA_1\right)^\dagger\Lambda_1^{-1}\right\| > \right.\\
&\quad \left\|\Lambda_2^2\right\|\left\|\Lambda_1^{-1}\right\|_F\sqrt{\frac{12r}{p}}\cdot t+\left\|\Lambda_2^2\right\|_F\left\|\Lambda_1^{-1}\right\|\frac{\mathrm{e}\sqrt{r+p}}{p+1}\cdot t\\
&\quad \left.+\left\|\Lambda_2^2\right\|\left\|\Lambda_1^{-1}\right\|\frac{\mathrm{e}\sqrt{r+p}}{p+1}\cdot tu\right\}\leq\\
&\quad \mathrm{e}^{-u^2/2}+4t^{-p}+t^{-(p+1)}.
\end{aligned}$$

Therefore,

$$\begin{aligned}
&\Pr\left\{\left\|\Lambda_2^2\left(V_2^TA_1\right)\left(V_1^TA_1\right)^\dagger\Lambda_1^{-1}\right\|+\|\Lambda_2\| > \right.\\
&\quad \left(1+t\sqrt{\frac{12r}{p}}\left(\sum_{i=1}^{r}\lambda_i^{-1}\right)^{1/2}+\frac{\mathrm{e}\sqrt{r+p}}{p+1}\cdot tu\lambda_r^{-1}\right)\lambda_{r+1}^2+\\
&\quad \left.\frac{\mathrm{e}\sqrt{r+p}}{p+1}\cdot t\lambda_r^{-1}\left(\sum_{i=r+1}^{n}\lambda_i^2\right)^{1/2}\right\}\leq\\
&\quad \mathrm{e}^{-u^2/2}+4t^{-p}+t^{-(p+1)}. \qquad (5.37)
\end{aligned}$$

Since Theorem 8 implies $\|X-L\|\leq\left\|\Lambda_2^2\left(V_2^TA_1\right)\left(V_1^TA_1\right)^\dagger\Lambda_1^{-1}\right\|+\|\Lambda_2\|$, we obtain the deviation bound in Theorem 12. This completes the proof. □

5.1.5 Empirical Study

We first evaluate the efficiency of the BRP based low-rank approximation (5.1) for exact recovery of low-rank matrices. We consider square matrices of dimension n from 500 to 30000 with rank r from 50 to 500. Each matrix is generated by AB, wherein A and B are both $n\times r$ standard Gaussian matrices. Figure 5.1 shows that the recovery time is linearly increased w.r.t n. This is consistent with the $r^2(2n+r)+mnr$ flops required by (5.1). The relative error of each recovery is less than 10^{-14}. It also shows that a 30000×30000 matrix with rank 500 can be exactly recovered within 200 CPU seconds. This suggests the advantage of (5.1) for large-scale applications.

We then evaluate the effectiveness of (5.1) and its power scheme modification

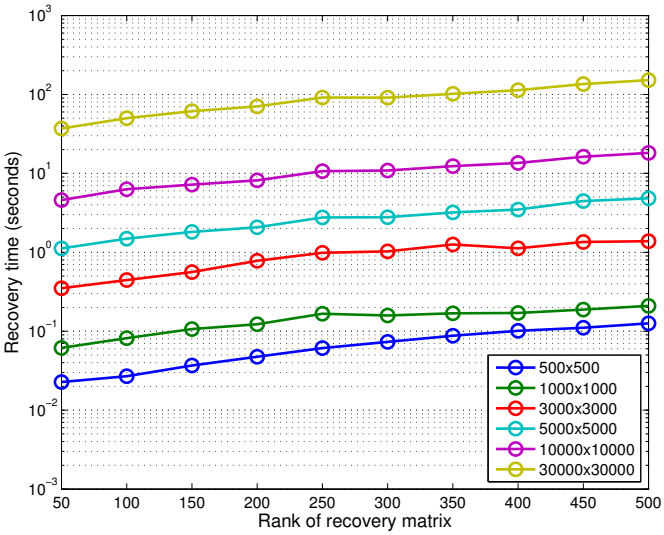

Figure 5.1: low-rank matrix recovery via BRP: the recovery time for matrices of different size and different rank.

(5.5) in low-rank approximation of full rank matrix with slowly decaying singular values. We generate a square matrix with size 1000, whose entries are independently sampled from a standard normal distribution with mean 0 and variance 1, and then apply (5.1) ($q = 0$) and (5.5) with $q = 1, 2, 3$ to obtain approximations with rank varying from 1 to 600. We show the relative errors in Figure 5.2 and the relative error of the corresponding SVD approximation as a baseline. The results suggest that our method can obtain a nearly optimal approximation when q is sufficiently large (e.g., 2).

At last, we evaluate the efficiency and effectiveness of BRP on low-rank compression of human face images from dataset FERET [188]. We randomly selected 700 face images of 100 individuals from FERET and built a 700×1600 data matrix, wherein the 1600 features are the 40×40 pixels of each image. We then obtain two rank-60 compressions of the data matrix by using SVD and the power modification of BRP based low-rank approximation (5.5) with $q = 1$, respectively. The compressed images and the corresponding time costs are shown in Figure 5.3 and its caption. It indicates that our method is able to produce compression with competitive quality in considerably less time than SVD.

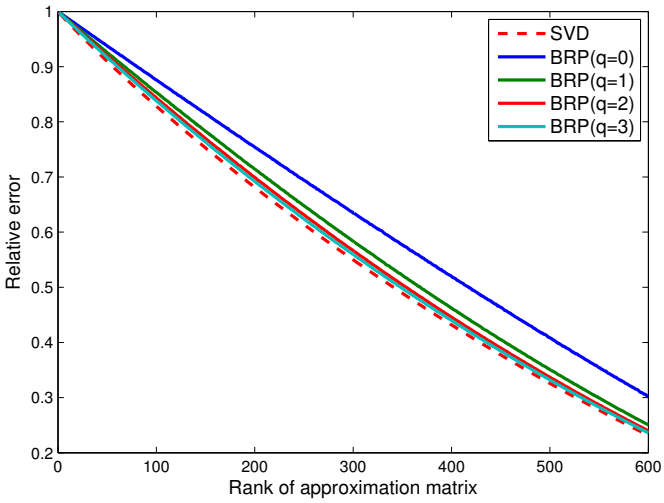

Figure 5.2: low-rank approximation via BRP: the relative approximation error for a 1000×1000 matrix with standard normal distributed entries on different rank.

5.2 Greedy Bilateral Sketch (GreBske)

GreBske is an application of "greedy bilateral (GreB)" paradigm, which will be given in Chapter 7.

5.2.1 Low-rank Approximation

Real world data matrix is hardly to be exactly low-rank. However, approximating it by a low-rank one presents a good trade-off between accuracy and time/space costs, especially when its singular values decay fast. The low-rank approximation [238] can replace the original matrix in least square regression and matrix product, and also provides a low-dimensional representation which can boosts the classification and clustering performance. Although the low-rank approximation is provably optimal when constructed from SVD, the expensive time cost makes SVD prohibitive to large matrix. Thus many faster Monte Carlo algorithms [75][53][183] have been proposed at the cost of producing sufficiently small error to SVD with high probability. Typically, they build the low-rank approximation from random selected columns or rows [27], or linear projections (which are called "sketch") on certain random matrix [108]. Some of them compute bilateral

Figure 5.3: low-rank image compression via BRP on FERET: BRP compresses 700 40×40 face images sampled from 100 individuals to a 700×1600 matrix with rank 60. Upper row: Original images. Middle row: images compressed by SVD (6.59s). Bottom row: images compressed by BRP (0.36s).

sketches [85][263] for both column space and row space in building the approximation. In this case, the matrix X is approximated as $X = USV$, wherein $U = XA_1$ and $V = A_2^T X$ are right and left sketches, A_1 and A_2 are random matrices.

In this chapter, we omit S and consider optimizing U and V:

$$\begin{aligned} &\min_{U,V} \|X - UV\|_F^2 \\ &\text{s.t.} \quad rank(U) = rank(V) \leq r. \end{aligned} \tag{5.38}$$

5.2.2 Greedy Bilateral Sketch

Alternately optimizing U and V in (5.38) immediately yields the following updating rules, note subscript in $\cdot_k$ denotes the variable in the k^{th} iterate and $(\cdot)^\dagger$ stands for the Moore-Penrose pseudo-inverse:

$$\begin{cases} U_k = XV_{k-1}^T \left(V_{k-1}V_{k-1}^T\right)^\dagger, \\ V_k = \left(U_k^T U_k\right)^\dagger U_k^T X. \end{cases} \tag{5.39}$$

It can be observed that the object value in (5.38) is merely determined by the matrix product UV rather than individual U or V, and different (U, V) pair can produce the same UV. It is then of interest to find a pair of (U, V) that have the same product as (U_k, V_k) in (5.39) but can be computed in less time than U_k and

V_k. For this purpose, we investigate the product U_kV_k

$$U_kV_k = U_k \left(U_k^T U_k\right)^\dagger U_k^T X = \mathcal{P}_{U_k} X. \tag{5.40}$$

This implies that the product U_kV_k equals to the orthogonal projection of X onto the column space of U_k. According to (5.39), the column space of U_k can be represented by arbitrary orthonormal basis for the columns of XV_{k-1}^T. For example, we can compute it as Q via fast QR decomposition $XV_{k-1}^T = QR$. In this case, the product U_kV_k can be equivalently computed as $U_kV_k = \mathcal{P}_Q X = QQ^T X$. Therefore, U_k and V_k in (5.39) can be replaced by Q and $Q^T X$ respectively, while the product U_kV_k and the corresponding object value are kept the same. This gives a faster updating procedure

$$\begin{cases} U_k = Q, \mathrm{QR}\left(XV_{k-1}^T\right) = QR, \\ V_k = Q^T X. \end{cases} \tag{5.41}$$

This alternating update can be viewed as mutually adaptive optimization of right sketch XV^T and left sketch $Q^T X$ for X, where the right projection matrix V^T is the former left sketch, and the left projection matrix Q is the orthonormal basis of the former right sketch. This mutually adaptive optimization of left and right factors will appear in GreBcom and GreBsmo as well.

Since the right and left sketches respectively describe the column and row spaces, which largely decide the approximation precision, we can temporarily ignore the QR decomposition in order to see how the column/row space is tracked within this scheme. Specifically, we start from a random matrix V and repeat $U_k = XV_{k-1}, V_k = U_k^T X$ for K times, the resulting $V_K = V(X^T X)^K$ are exactly the randomized SVD under power scheme [195] given in [108]. The K-order matrix exponential accelerates the decaying of singular values and thus improves the approximation precision. Similar theoretical analysis as [108] supports that (5.39) achieves the same accuracy as power scheme randomized SVD.

Different from power scheme whose speed would be quickly limited with the increasing of power K, GreBske invokes the updates in (5.39) with a greedy incremental rank r for both U and V. In particular, GreBske starts from a $V \in \mathbb{R}^{n \times r_0}$ with a small integer r_0, iterates (5.39) for K times, and then augment

the rank of V to $r_1 = r_0 + \Delta r$ by adding Δr extra rows to V, where Δr is the rank step size. In GreB, the Δr rows are selected greedily as the top Δr row basis on which the object decreases fastest. Accordingly, they maximizes the magnitude of the partial derivative of the object w.r.t. UV, which is

$$\frac{\partial \|X - UV\|_F^2}{\partial UV} = X - UV. \tag{5.42}$$

Hence the Δr rows are the top Δr right singular vectors of the fat matrix $U^T(X - UV)$, which can be quickly obtained by a small SVD or its faster approximation like random projections $A^T U^T(X - UV)$, wherein A is a $r_i \times \Delta r$ random matrix. The rank r stops augmenting when reaching certain error tolerance.

In GreBske, the top r_i row basis are successfully obtained when optimizing V of $r_i + \Delta r$ rows. The essential task of the updates is to optimize the added Δr rows, while the first r_i rows take part in the update merely for keep the incoherence between rows. So it converges faster than simultaneously optimizing the whole r rows. In addition, the newly added Δr rows are initialized as the fastest decreasing directions, so its distance to the optimal solution is shortened. The computation in GreBske is dominated by the two matrix multiplications that take $2mnr_i$ flops. It can be further speeded up if designing sparse updates of U and V, which will be studied in future works.

The detailed algorithm of GreBske is given in Algorithm 5.

Algorithm 5: Greedy Bilateral Sketch (GreBske)

Input: Object function f; rank step size Δr; power K; tolerance τ; observations of data matrix X
Output: low-rank matrix UV (and sparse S)
Initialize $V \in \mathbb{R}^{r_0 \times n}$ (and S);
while *residual error* $\leq \tau$ **do**
  for $k \leftarrow 1$ **to** K **do**
    Greedy Bilateral Sketch: sequentially compute (5.41);
  end
  Calculate the top Δr right singular vectors v (or Δr-dimensional random projections) of $\partial f / \partial V$ given in (5.42);
  Set $V := [V; v]$;
end

5.2.3 Empirical Study

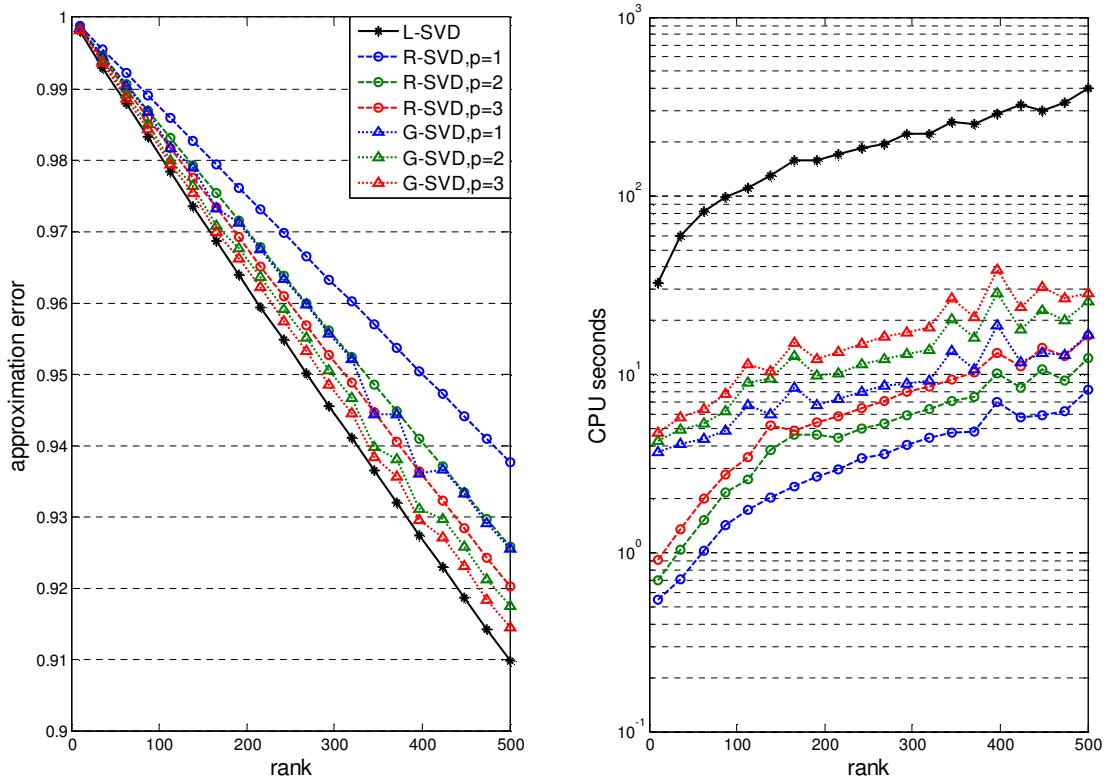

Figure 5.4: Low-rank approximation performed by Lanczos method (L-SVD), randomized SVD (R-SVD) and GreBske (G-SVD) on $10^4 \times 10^4$ matrix whose entries are sampled from i.i.d. normal distribution, p (K in G-SVD) is the power parameter.

The approximation accuracy and time cost of GreBske is evaluated on the task of approximating a randomly generated $10^4 \times 10^4$ matrix X, with thorough comparison to the results obtained by Lanczos algorithm for SVD and randomized SVD [108], which are two popular approximation algorithms of SVD. Each entry of the matrix is sampled from an i.i.d. standard Gaussian distribution $\mathcal{N}(0, 1)$. We uniformly select 20 values from 1 to 500 as the rank parameters. Then the associated 20 low-rank approximations are computed by the three different algorithms. Randomized SVD and GreBske of different power parameters are tested. We show the approximation error $\|\hat{X} - X\|_F / \|X\|_F$ and CPU seconds for each approximation $\hat{X}$ in the two right plots of Figure 5.2.3. It can been

verified that Lanczos method achieves the smallest error yet along with expensive computations, while randomized SVD has the fastest speed yet largest error. GreBske has error very close to that of Lanczos method, but its computational time is comparable with that of randomized SVD. Thus it provides a good trade-off between speed and accuracy, which is highly preferred in real applications.

It is necessary to discuss the connections and differences between BRP and GreBske, because they share the goal of solving low-rank matrix approximation problem. Their most critical differences are their strategies to find out the column space of a matrix. While BRP use random projections, GreBske adopts greedy optimization. Randomized and greedy strategies have their advantages and disadvantages comparing to each other. Greedy strategy usually provides better approximation accuracy, but its computation cannot be paralleled like randomized strategy so its time cost is higher. Although we did not show direct comparison between these two proposed methods, we compare the randomized SVD and GreBske in Figure 5.2.3, and the results validate above analysis. Both BRP and randomized SVD use random projections to build the low-rank approximation, but BRP builds a closed-form solution from left and right random projections. They share similar approximation error, but BRP has simpler implementation and faster speed.

Chapter 6

GO Decomposition and Randomized Low-rank + Sparse Decomposition

Low-rank and sparse structures have been profoundly studied in matrix completion and compressed sensing. In this chapter, we develop "Go Decomposition" (GoDec) to efficiently and robustly estimate the low-rank part L and the sparse part S of a matrix $X = L + S + G$ with noise G. GoDec alternatively assigns the low-rank approximation of $X - S$ to L and the sparse approximation of $X - L$ to S. The algorithm can be significantly accelerated by bilateral random projections (BRP) in Chapter 5. We also propose GoDec for matrix completion as an important variant in Chapter 8. We prove that the objective value $\|X - L - S\|_F^2$ converges to a local minimum, while L and S linearly converge to local optimums. Theoretically, we analyze the influence of L, S and G to the asymptotic/convergence speeds in order to discover the robustness of GoDec. Empirical studies validate the efficiency, robustness and effectiveness of GoDec comparing with representative matrix decomposition and completion tools, e.g., Robust PCA and OptSpace.

6.1 Introduction

It has proven in compressed sensing [71] that a sparse signal can be exactly recovered from a small number of its random measurements, and in matrix completion [137] that a low-rank matrix can be exactly completed from a few of its entries sampled at random. When signals are neither sparse nor low-rank, its low-rank and sparse structure can be explored by either approximation or decomposition.

Recent research about exploring low-rank and sparse structures [269] concentrates on developing fast approximations and meaningful decompositions. Two appealing representatives are the randomized approximate matrix decomposition [108] and the robust principal component analysis (RPCA) [37]. The former proves that a matrix can be well approximated by its projection onto the column space of its random projections. This rank-revealing method provides a fast approximation of SVD/PCA. The latter proves that the low-rank and the sparse components of a matrix can be exactly recovered if it has a unique and precise "low-rank+sparse" decomposition. RPCA offers a blind separation of low-rank data and sparse noises.

We study the approximated "low-rank+sparse" decomposition of a matrix X, i.e.,

$$X = L + S + G, \text{rank}(L) \leq r, \text{card}(S) \leq k, \tag{6.1}$$

where G is the noise. This problem is intrinsically different from RPCA that assumes $X = L + S$. In this chapter, we develop "Go Decomposition" (GoDec) to estimate the low-rank part L and the sparse part S from X. We show that BRP can significantly accelerate GoDec.

In particular, GoDec alternatively assigns the r-rank approximation of $X - S$ to L and assigns the sparse approximation with cardinality k of $X - L$ to S. The updating of L is obtained via singular value hard thresholding of $X - S$, while the updating of S is obtained via entry-wise hard thresholding [29] of $X - L$. The term "Go" is owing to the similarities between L/S in the GoDec iteration rounds and the two players in the game of go. BRP based low-rank approximation is applied to accelerating the r-rank approximation of $X - S$ in GoDec. We show GoDec can be extended to solve matrix completion problem with competitive robustness and efficiency.

We theoretically analyze the convergence of GoDec. The objective value (decomposition error) $\|X-L-S\|_F^2$ monotonically decreases and converges to a local minimum. Since the updating of L and S in GoDec is equivalent to alternatively projecting L or S onto two smooth manifolds, we use the framework proposed in [153] to prove the asymptotical property and linear convergence of L and S. The asymptotic and convergence speeds are mainly determined by the angle between the two manifolds. We discuss how L, S and G influence the speeds via influencing the cosine of the angle. The analyses show the convergence of GoDec is robust to the noise G.

Both GoDec and RPCA can explore the low-rank and sparse structures in X, but they are intrinsically different. RPCA assumes $X = L + S$ (S is sparse noise) and exactly decomposes X into L and S without predefined rank(L) and card(S). However, GoDec produces approximated decomposition of a general matrix X whose exact RPCA decomposition does not exist due to the additive noise G and pre-defined rank(L) and card(S). In practice, rank(L) and card(S) are preferred to be restricted in order to control the model complexity. Another major difference is that GoDec directly constrains the rank range of L and the cardinality range of S, while RPCA minimizes their corresponding convex polytopes, i.e., the nuclear norm of L and ℓ_1 norm of S. Chandrasekaran et al. [43] proposed an exact decomposition based on a different assumption but the same optimization procedure used in RPCA. Stable principal component pursuit [271] is an extension of RPCA to handle noise by minimizing the nuclear norm and ℓ_1 norm. Therefore, they are different from GoDec. In addition, GoDec can be extended to solve matrix completion problems (Chapter 8) because it is able to control the support set of S, while RPCA cannot because the support set of S is automatically determined.

GoDec has low computational cost in "low-rank+sparse" decomposition and matrix completion tasks. It is powerful in background modeling of videos and shadow/light removal of images. For example, it processes a 200 frame video with 256×320 resolution within 200 seconds, while RPCA requires $1,800+$ seconds.

In this chapter, a standard Gaussian matrix is a random matrix whose entries are independent standard normal variables; the SVD of a matrix X is $U\Lambda V^T$ and λ_i or $\lambda_i(X)$ stands for the i^{th} largest singular value of X; $\mathcal{P}_\Omega(\cdot)$ is the projection

of a matrix to an entry set Ω; and the QR decomposition of a matrix results in Q and R.

6.2 Go Decomposition (GoDec)

The approximated "low-rank+sparse" decomposition problem stated in (6.1) can be solved by minimizing the decomposition error:

$$\begin{aligned} \min_{L,S} \quad & \|X - L - S\|_F^2 \\ s.t. \quad & \text{rank}\,(L) \leq r, \\ & \text{card}\,(S) \leq k. \end{aligned} \tag{6.2}$$

6.2.1 Naïve GoDec

We propose the naïve GoDec algorithm in this section. The optimization problem of GoDec (6.2) can be solved by alternatively solving the following two subproblems until convergence:

$$\begin{cases} L_t = \arg \min\limits_{\text{rank}(L) \leq r} \|X - L - S_{t-1}\|_F^2; \\ S_t = \arg \min\limits_{\text{card}(S) \leq k} \|X - L_t - S\|_F^2. \end{cases} \tag{6.3}$$

Although both subproblems (6.3) have nonconvex constraints, their global solutions L_t and S_t exist.

In particular, the two subproblems in (6.3) can be solved by updating L_t via singular value hard thresholding of $X - S_{t-1}$ and updating S_t via entry-wise hard thresholding of $X - L_t$, respectively, i.e.,

$$\begin{cases} L_t = \sum\limits_{i=1}^{r} \lambda_i U_i V_i^T, \text{svd}\,(X - S_{t-1}) = U \Lambda V^T; \\ S_t = \mathcal{P}_\Omega\,(X - L_t)\,, \Omega : \left|(X - L_t)_{i,j \in \Omega}\right| \neq 0 \\ \quad \text{and} \ \geq \left|(X - L_t)_{i,j \in \overline{\Omega}}\right|, |\Omega| \leq k. \end{cases} \tag{6.4}$$

The main computation in the naïve GoDec algorithm (6.4) is the SVD of $X - S_{t-1}$ in the updating L_t sequence. SVD requires $\min\left(mn^2, m^2 n\right)$ flops, so it is

impractical when X is of large size.

6.2.2 Fast GoDec via BRP based approximation

Since BRP based low-rank approximation is near optimal and efficient, we replace SVD with BRP in naïve GoDec in order to significantly reduce the time cost.

We summarize GoDec using BRP based low-rank approximation (5.1) and power scheme modification (5.5) in Algorithm 1. When $q = 0$, For dense X, (5.1) is applied. Thus the QR decomposition of Y_1 and Y_2 in Algorithm 1 are not performed, and L_t is updated as $L_t = Y_1 \left(A_2^T Y_1\right)^{-1} Y_2^T$. In this case, Algorithm 6 requires $r^2(2n + r) + 4mnr$ flops per iteration. When integer $q > 0$, (5.5) is applied and Algorithm 1 requires $r^2(m + 3n + 4r) + (4q + 4)mnr$ flops per iteration.

Algorithm 6: GO Decomposition (GoDec)

Input: X, r, k, ϵ, q
Output: L, S
Initialize $L_0 := X$, $S_0 := \mathbf{0}$, $t := 0$;
while $\|X - L_t - S_t\|_F^2 / \|X\|_F^2 > \epsilon$ **do**
  $t := t + 1$;
  $\tilde{L} = \left[(X - S_{t-1})(X - S_{t-1})^T\right]^q (X - S_{t-1})$;
  $Y_1 = \tilde{L} A_1$, $A_2 = Y_1$;
  $Y_2 = \tilde{L}^T Y_1 = Q_2 R_2$, $Y_1 = \tilde{L} Y_2 = Q_1 R_1$;
  If rank $\left(A_2^T Y_1\right) < r$ **then** $r :=$ rank $\left(A_2^T Y_1\right)$, go to the first step; **end**;
  $L_t = Q_1 \left[R_1 \left(A_2^T Y_1\right)^{-1} R_2^T\right]^{1/(2q+1)} Q_2^T$;
  $S_t = \mathcal{P}_\Omega (X - L_t)$, Ω is the nonzero subset of the first k largest entries of $|X - L_t|$;
end

6.3 Convergence of GoDec

In this section, we analyze the convergence properties of GoDec. In particular, we first prove that the objective value $\|X - L - S\|_F^2$ (decomposition error) converges

to a local minimum. Then we demonstrate the asymptotic properties of GoDec and prove that the solutions L and S respectively converge to local optimums with linear rate less than 1, by using the framework presented in [153]. The influence of L, S and G to the asymptotic/convergence speeds is analyzed. The speeds will be slowed by augmenting the magnitude of noise part $\|G\|_F^2$. However, the convergence will not be harmed unless $\|G\|_F^2 \ll \|L\|_F^2$ or $\|G\|_F^2 \ll \|S\|_F^2$.

We have the following theorem about the convergence of the objective value $\|X - L - S\|_F^2$ in (6.2).

Theorem 13. (**Convergence of objective value**). *The alternative optimization (6.3) produces a sequence of* $\|X-L-S\|_F^2$ *that converges to a local minimum.*

Proof. Let the objective value $\|X - L - S\|_F^2$ after solving the two subproblems in (6.3) be E_t^1 and E_t^2, respectively, in the t^{th} iteration. On the one hand, we have

$$E_t^1 = \|X - L_t - S_{t-1}\|_F^2, E_t^2 = \|X - L_t - S_t\|_F^2. \tag{6.5}$$

The global optimality of S_t yields $E_t^1 \geq E_t^2$. On the other hand,

$$E_t^2 = \|X - L_t - S_t\|_F^2, E_{t+1}^1 = \|X - L_{t+1} - S_t\|_F^2. \tag{6.6}$$

The global optimality of L_{t+1} yields $E_t^2 \geq E_{t+1}^1$. Therefore, the objective values (decomposition errors) $\|X - L - S\|_F^2$ keep decreasing throughout GoDec (6.3):

$$E_1^1 \geq E_1^2 \geq E_2^1 \geq \cdots \geq E_t^1 \geq E_t^2 \geq E_{t+1}^1 \geq \cdots \tag{6.7}$$

Since the objective of (6.2) is monotonically decreasing and the constraints are satisfied all the time, (6.3) produces a sequence of objective values that converge to a local minimum. This completes the proof. □

The asymptotic property and the linear convergence of L and S in GoDec are demonstrated based on the framework proposed in [153]. We firstly consider L. From a different prospective, GoDec algorithm shown in (6.4) is equivalent to iteratively projecting L onto one manifold $\mathcal{M}$ and then onto another manifold $\mathcal{N}$. This kind of optimization method is the so called "alternating projections on manifolds". To see this, in (6.4), by substituting S_t into the next updating of

L_{t+1}, we have:

$$L_{t+1} = \mathcal{P}_{\mathcal{M}}\left(X - \mathcal{P}_{\Omega}\left(X - L_t\right)\right) = \mathcal{P}_{\mathcal{M}}\mathcal{P}_{\mathcal{N}}\left(L_t\right), \tag{6.8}$$

Both $\mathcal{M}$ and $\mathcal{N}$ are two C^k-manifolds around a point $\overline{L} \in \mathcal{M} \cap \mathcal{N}$:

$$\begin{cases} \mathcal{M} = \left\{H \in \mathbb{R}^{m\times n} : \operatorname{rank}\left(H\right) = r\right\}, \\ \mathcal{N} = \left\{X - \mathcal{P}_{\Omega}\left(X - H\right) : H \in \mathbb{R}^{m\times n}\right\}. \end{cases} \tag{6.9}$$

According to the above definitions, any point $L \in \mathcal{M} \cap \mathcal{N}$ satisfies:

$$L = \mathcal{P}_{\mathcal{M}\cap\mathcal{N}}\left(L\right) \Rightarrow \tag{6.10}$$

$$L = X - \mathcal{P}_{\Omega}\left(X - L\right), \operatorname{rank}\left(L\right) = r. \tag{6.11}$$

Thus any point $L \in \mathcal{M} \cap \mathcal{N}$ is a local solution of L in (6.2).

We define the angle between two manifolds $\mathcal{M}$ and $\mathcal{N}$ at point L as the angle between the corresponding tangent spaces $T_{\mathcal{M}}(L)$ and $T_{\mathcal{N}}(L)$. The angle is between 0 and $\pi/2$ with cosine:

$$c\left(\mathcal{M}, \mathcal{N}, L\right) = c\left(T_{\mathcal{M}}(L), T_{\mathcal{N}}(L)\right). \tag{6.12}$$

In addition, if $\mathbb{S}$ is the unit sphere in $\mathbb{R}^{m\times n}$, the angle between two subspaces M and N in $\mathbb{R}^{m\times n}$ is defined as the angle between 0 and $\pi/2$ with cosine:

$$c\left(M, N\right) = \max\Big\{\langle x, y\rangle : x \in \mathbb{S} \cap M \cap \left(M \cap N\right)^{\perp}, \\ y \in \mathbb{S} \cap N \cap \left(M \cap N\right)^{\perp}\Big\}.$$

We give the following proposition about the angle between two subspaces M and N:

Proposition 9. *Following the above definition of the angle between two subspaces M and N, we have*

$$c\left(M, N\right) = \max\big\{\langle x, y\rangle : x \in \mathbb{S} \cap M \cap N^{\perp}, \\ y \in \mathbb{S} \cap N \cap M^{\perp}\big\}.$$

The angle between $\mathcal{M}$ and $\mathcal{N}$ is used in the asymptotical property and the linear convergence rate of "alternating projections on manifolds" algorithms.

Theorem 14. (**Asymptotic property** [153]). *Let $\mathcal{M}$ and $\mathcal{N}$ be two transverse C^2-manifolds around a point $\overline{L} \in \mathcal{M} \cap \mathcal{N}$. Then*

$$\limsup_{L \to \overline{L}, L \notin \mathcal{M} \cap \mathcal{N}} \frac{\|\mathcal{P}_{\mathcal{M}} \mathcal{P}_{\mathcal{N}}(L) - \mathcal{P}_{\mathcal{M} \cap \mathcal{N}}(L)\|}{\|L - \mathcal{P}_{\mathcal{M} \cap \mathcal{N}}(L)\|} \leq c\left(\mathcal{M}, \mathcal{N}, \overline{L}\right).$$

A refinement of the above argument is

$$\limsup_{L \to \overline{L}, L \notin \mathcal{M} \cap \mathcal{N}} \frac{\|(\mathcal{P}_{\mathcal{M}} \mathcal{P}_{\mathcal{N}})^n(L) - \mathcal{P}_{\mathcal{M} \cap \mathcal{N}}(L)\|}{\|L - \mathcal{P}_{\mathcal{M} \cap \mathcal{N}}(L)\|} \leq c^{2n-1}$$

for $n = 1, 2, ...$ and $c = c\left(\mathcal{M}, \mathcal{N}, \overline{L}\right)$.

Theorem 15. (**Linear convergence of variables** [153]). *In $\mathbb{R}^{m \times n}$, let $\mathcal{M}$ and $\mathcal{N}$ be two transverse manifolds around a point $\overline{L} \in \mathcal{M} \cap \mathcal{N}$. If the initial point $L_0 \in \mathbb{R}^{m \times n}$ is close to $\overline{L}$, then the method of alternating projections*

$$L_{t+1} = \mathcal{P}_{\mathcal{M}} \mathcal{P}_{\mathcal{N}}(L_t), (t = 0, 1, 2, ...)$$

is well-defined, and the distance $d_{\mathcal{M} \cap \mathcal{N}}(L_t)$ from the iterate L_t to the intersection $\mathcal{M} \cap \mathcal{N}$ decreases Q-linearly to zero. More precisely, given any constant c strictly larger than the cosine of the angle of the intersection between the manifolds, $(\mathcal{M}, \mathcal{N}, \overline{L})$, if L_0 is close to $\overline{L}$, then the iterates satisfy

$$d_{\mathcal{M} \cap \mathcal{N}}(L_{t+1}) \leq c \cdot d_{\mathcal{M} \cap \mathcal{N}}(L_t), (t = 0, 1, 2, ...)$$

Furthermore, L_t converges linearly to some point $L^* \in \mathcal{M} \cap \mathcal{N}$, i.e., for some constant $\alpha > 0$,

$$\|L_t - L^*\| \leq \alpha c^t, (t = 0, 1, 2, ...).$$

Since GoDec algorithm can be written as the form of alternating projections on two manifolds $\mathcal{M}$ and $\mathcal{N}$ given in (6.9) and they satisfy the assumptions of Theorem 14 and Theorem 15, L in GoDec converges to a local optimum with linear rate. Similarly, we can prove the linear convergence of S.

Since cosine $c(\mathcal{M}, \mathcal{N}, \overline{L})$ in Theorem 14 and Theorem 15 determines the asymptotic and convergence speeds of the algorithm. We discuss how L, S and G influence the asymptotic and convergence speeds via analyzing the relationship between L, S, G and $c(\mathcal{M}, \mathcal{N}, \overline{L})$.

Theorem 16. (**Asymptotic and convergence speed**). *In GoDec, the asymptotical improvement and the linear convergence of L and S stated in Theorem 14 and Theorem 15 will be slowed by augmenting*

$$\begin{aligned}
&\text{For } L: \frac{\|\Delta_L\|_F}{\|L+\Delta_L\|_F}, \Delta_L = (S+G) - \mathcal{P}_\Omega\left(S+G\right), \\
&\text{For } S: \frac{\|\Delta_S\|_F}{\|S+\Delta_S\|_F}, \Delta_S = (L+G) - \mathcal{P}_\mathcal{M}\left(L+G\right).
\end{aligned}$$

However, the asymptotical improvement and the linear convergence will not be harmed and is robust to the noise G unless when $\|G\|_F \gg \|S\|_F$ and $\|G\|_F \gg \|L\|_F$, which lead the two terms increasing to 1.

Proof. GoDec approximately decomposes a matrix $X = L + S + G$ into the low-rank part L and the sparse part S. According to the above analysis, GoDec is equivalent to alternating projections of L on $\mathcal{M}$ and $\mathcal{N}$, which are given in (6.9). According to Theorem 14 and Theorem 15, smaller $c(\mathcal{M}, \mathcal{N}, \overline{L})$ produces faster asymptotic and convergence speeds, while $c(\mathcal{M}, \mathcal{N}, \overline{L}) = 1$ is possible to make L and S stopping converging. Below we discuss how L, S and G influence $c(\mathcal{M}, \mathcal{N}, \overline{L})$ and further influence the asymptotic and convergence speeds of GeDec.

According to (6.12), we have

$$c\left(\mathcal{M}, \mathcal{N}, \overline{L}\right) = c\left(T_\mathcal{M}(\overline{L}), T_\mathcal{N}(\overline{L})\right). \tag{6.13}$$

Substituting the equation given in Proposition 9 into the right-hand side of the above equation yields

$$\begin{aligned}
c\left(\mathcal{M}, \mathcal{N}, \overline{L}\right) = \max\big\{\langle x, y\rangle : x \in \mathbb{S} \cap T_\mathcal{M}(\overline{L}) \cap N_\mathcal{N}(\overline{L}), \\
y \in \mathbb{S} \cap T_\mathcal{N}(\overline{L}) \cap N_\mathcal{M}(\overline{L})\big\}.
\end{aligned} \tag{6.14}$$

The normal spaces of manifolds $\mathcal{M}$ and $\mathcal{N}$ on point $\overline{L}$ is respectively given by

$$\begin{aligned} N_{\mathcal{M}}(\overline{L}) &= \left\{ y \in \mathbb{R}^{m \times n} : u_i^T y v_j = 0, \overline{L} = UDV^T \right\}, \\ N_{\mathcal{N}}(\overline{L}) &= \left\{ X - \mathcal{P}_\Omega \left(X - \overline{L} \right) \right\}, \end{aligned} \tag{6.15}$$

where $\overline{L} = UDV^T$ represents the eigenvalue decomposition of $\overline{L}$, $U = [u_1, ..., u_r]$ and $V = [v_1, ..., v_r]$. Assume $X = \overline{L} + \overline{S} + \overline{G}$, wherein $\overline{G}$ is the noise corresponding to $\overline{L}$, we have

$$\begin{aligned} &\overline{L} = X - \left(\overline{S} + \overline{G} \right), \\ &\hat{L} = X - \mathcal{P}_\Omega \left(\overline{S} + \overline{G} \right), \Rightarrow \\ &\hat{L} = \overline{L} + \left[\left(\overline{S} + \overline{G} \right) - \mathcal{P}_\Omega \left(\overline{S} + \overline{G} \right) \right] = \overline{L} + \Delta. \end{aligned} \tag{6.16}$$

Thus the normal space of manifold $\mathcal{N}$ is

$$N_{\mathcal{N}}(\overline{L}) = \left\{ \overline{L} + \Delta \right\}. \tag{6.17}$$

Since the tangent space is the complement space of the normal space, by using the normal space of $\mathcal{M}$ in (6.15) and the normal space of $\mathcal{N}$ given in (6.17), we can verify

$$N_{\mathcal{N}}(\overline{L}) \subseteq T_{\mathcal{M}}(\overline{L}), N_{\mathcal{M}}(\overline{L}) \subseteq T_{\mathcal{N}}(\overline{L}). \tag{6.18}$$

By substituting the above results into (6.14), we obtain

$$\begin{aligned} c\left(\mathcal{M}, \mathcal{N}, \overline{L} \right) = \max \big\{ \langle x, y \rangle : x \in \mathbb{S} \cap N_{\mathcal{N}}(\overline{L}), \\ y \in \mathbb{S} \cap N_{\mathcal{M}}(\overline{L}) \big\}. \end{aligned} \tag{6.19}$$

Hence we have

$$\begin{aligned} \langle x, y \rangle &= \operatorname{tr}\left(VDU^T y + \Delta^T y \right) \\ &= \operatorname{tr}\left(DU^T y V \right) + \operatorname{tr}\left(\Delta^T y \right) = \operatorname{tr}\left(\Delta^T y \right). \end{aligned} \tag{6.20}$$

The last equivalence is due to $u_i^T y v_j = 0$ in (6.15). Thus

$$c\left(\mathcal{M}, \mathcal{N}, \overline{L} \right) = \max \left\{ \langle x, y \rangle \right\} \leq \max \left\{ \langle D_\Delta, D_y \rangle \right\}, \tag{6.21}$$

where the diagonal entries of D_Δ and D_y are composed by eigenvalues of Δ and y, respectively. The last inequality is obtained by considering the case when x and y have identical left and right singular vectors. Because $\overline{L}+\Delta, y \in \mathbb{S}$ infers $\|\overline{L}+\Delta\|_F^2 = \|y\|_F^2 = 1$, we have

$$\begin{aligned} c\left(\mathcal{M},\mathcal{N},\overline{L}\right) &\leq \max\left\{\langle D_\Delta, D_y\rangle\right\} \\ &\leq \|D_\Delta\|_F \|D_y\|_F \leq \|D_\Delta\|_F . \end{aligned} \tag{6.22}$$

Since c in Theorem 15 can be selected as any constant that is strictly larger than $c\left(\mathcal{M},\mathcal{N},\overline{L}\right) \leq \|D_\Delta\|_F$, we can choose $c = c\left(\mathcal{M},\mathcal{N},\overline{L}\right)+\Delta_c \leq \|D_\Delta\|_F$. In Theorem 14, the cosine $c\left(\mathcal{M},\mathcal{N},\overline{L}\right)$ is directly used.

Therefore, the asymptotic and convergence speeds of L will be slowed by augmenting $\|\Delta\|_F$, and vice versa. However, the asymptotical improvement and the linear convergence will not be jeopardized unless $\|\Delta\|_F = 1$. For general $L+\Delta$ that is not normalized onto the sphere $\mathbb{S}$, $\|\Delta\|_F$ should be replaced by $\|\Delta\|_F/\|L+\Delta\|_F$.

For the variable S, we can obtain an analogous result via an analysis in a similar style as above. For general $L+\Delta$ without normalization, the asymptotic/convergence speed of S will be slowed by augmenting $\|\Delta\|_F/\|S+\Delta\|_F$, and vice versa, wherein

$$\Delta = (L+G) - \mathcal{P}_{\mathcal{M}}\left(L+G\right). \tag{6.23}$$

The asymptotical improvement and the linear convergence will not be jeopardized unless $\|\Delta\|_F/\|S+\Delta\|_F = 1$.

This completes the proof. □

Theorem 16 reveals the influence of the low-rank part L, the sparse part S and the noise part G to the asymptotic/convergence speeds of L and S in GoDec. Both Δ_L and Δ_S are the element-wise hard thresholding error of $S+G$ and the singular value hard thresholding error of $L+G$, respectively. Large errors will slow the asymptotic and convergence speeds of GoDec. Since $S-\mathcal{P}_\Omega(S) = 0$ and $L-\mathcal{P}_{\mathcal{M}}(L) = 0$, the noise part G in Δ_L and Δ_S can be interpreted as the perturbations to S and L and deviates the two errors from 0. Thus noise G with large magnitude will decelerate the asymptotical improvement and the

linear convergence, but it will not ruin the convergence unless $\|G\|_F \gg \|S\|_F$ or $\|G\|_F \gg \|L\|_F$. Therefore, GoDec is robust to the additive noise in X and is able to find the approximated $L+S$ decomposition when noise G is not overwhelming.

6.4 Experiments

This section evaluates both the effectiveness and the efficiency of the BRP based low-rank approximation and GoDec for computer vision applications, low-rank+sparse decomposition and matrix completion. We run all the experiments in MatLab on a server with dual quad-core 3.33 GHz Intel Xeon processors and 32 GB RAM. The relative error $\|X - \hat{X}\|_F^2 / \|X\|_F^2$ is used to evaluate the effectiveness, wherein X is the original matrix and $\hat{X}$ is an estimate/approximation.

6.4.1 RPCA vs. GoDec

Table 6.1: Relative error and time cost of RPCA and GoDec in low-rank+sparse decomposition tasks. The results separated by "/" are RPCA and GoDec, respectively.

size(X) (square)	rank(L) (1)	card(S) (10^4)	rel.error(X) (10^{-8})	rel.error(L) (10^{-8})	rel.error(S) (10^{-6})	time (seconds)
500	25	1.25	3.70/1.80	1.50/1.20	2.00/0.95	6.07/2.83
1000	50	5.00	4.98/4.56	1.82/1.85	5.16/4.90	20.96/12.71
2000	100	20.0	8.80/1.13	3.10/1.10	1.81/1.24	101.74/74.16
3000	250	45.0	6.29/4.98	5.09/5.05	33.9/55.3	562.09/266.11
5000	400	125	63.1/24.4	30.2/29.3	54.2/18.8	2495.31/840.39
10000	500	600	6.18/3.04	2.27/2.88	58.3/36.6	9560.74/3030.15

Since RPCA and GoDec are related in their motivations, we compare their relative errors and time costs on square matrices with different sizes, different ranks of low-rank components and different cardinality of sparse components. For a matrix $X = L + S + G$, its low-rank component is built as $L = AB$, wherein both A and B are $n \times r$ standard Gaussian matrices. Its sparse part is built as $S = \mathcal{P}_\Omega(D)$, wherein D is a standard Gaussian matrix and Ω is an entry set of size k drawn uniformly at random. Its noise part is built as $G =$

$10^{-3} \cdot F$, wherein F is a standard Gaussian matrix. In our experiments, we compare RPCA [1] (`inexact_alm_rpca`) with GoDec (Algorithm 6 with $q = 2$). Since both algorithms adopt the relative error of X as the stopping criterion, we use the same tolerance $\epsilon = 10^{-7}$. Table 6.1 shows the results and indicates that both algorithms are successful in recovering the correct "low-rank+sparse" decompositions with relative error less than 10^{-6}. GoDec usually produces less relative error with much less CPU seconds than RPCA. The improvement of accuracy is due to that the model of GoDec in (6.1) is more general than that of RPCA by considering the noise part. The improvement of speed is due to that BRP based low-rank approximation significantly saves the computation of each iteration round.

6.4.2 Background modeling

Figure 6.1: Background modeling results of four 200-frame surveillance video sequences in $X = L + S$ mode. Top left: lobby in an office building (resolution 128×160, learning time 39.75 seconds). Top right: shopping center (resolution 256×320, learning time 203.72 seconds). Bottom left: Restaurant (resolution 120×160, learning time 36.84 seconds). Bottom right: Hall of a business building (resolution 144×176, learning time 47.38 seconds).

Background modeling [48] is a challenging task to reveal the correlation be-

[1] http://watt.csl.illinois.edu/perceive/matrix-rank

tween video frames, model background variations and foreground moving objects. A video sequence satisfies the low-rank+sparse structure, because backgrounds of all the frames are related, while the variation and the moving objects are sparse and independent. We apply GoDec (Algorithm 2 with $q = 2$) to four surveillance videos [1], respectively. The matrix X is composed of the first 200 frames of each video. For example, the second video is composed of 200 frames with the resolution 256×320, we convert each frame as a vector and thus the matrix X is of size 81920×200. We show the decomposition result of one frame in each video sequence in Figure 6.1. The background and moving objects are precisely separated (the person in L of the fourth sequence does not move throughout the video) without losing details. The results of the first sequence and the fourth sequence are comparable with those shown in [37]. However, compared with RPCA (36 minutes for the first sequence and 43 minutes for the fourth sequence) [37], GoDec requires around 50 seconds for each of both. Therefore, GoDec makes large-scale applications available.

6.4.3 Shadow/Light removal

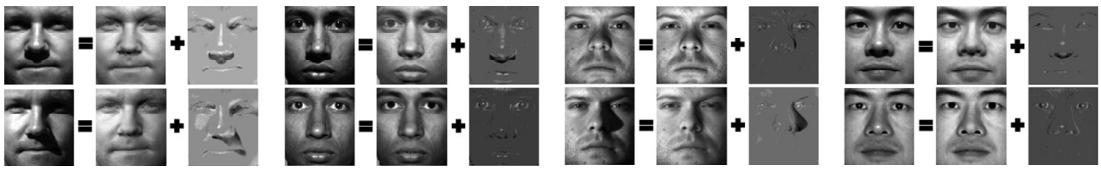

Figure 6.2: Shadow/light removal of face images from four individuals in Yale B database in $X = L + S$ mode. Each individual has 64 images with resolution 192×168 and needs 24 seconds learning time.

Shadow and light in training images always pull down the quality of learning in computer vision applications. GoDec can remove the shadow/light noises by assuming that they are sparse and the rest parts of the images are low-rank. We apply GoDec (Algorithm 2 with $q = 2$) to face images of four individuals in the Yale B database [2]. Each individual has 64 images with resolution 192×168 captured under different illuminations. Thus the matrix X for each individual is of

[1] http://perception.i2r.a-star.edu.sg/bk_model/bk_index.html
[2] http://cvc.yale.edu/projects/yalefacesB/yalefacesB.html

size 32760×64. We show the GoDec of eight example images (2 per individual) in Figure 6.2. The real face of each individual are remained in the low rank component, while the shadow/light noises are successfully removed from the real face images and stored in the sparse component. The learning time of GoDec for each individual is less than 30 seconds, which encourages for large-scale applications, while RPCA requies around 685 seconds.

6.5 Conclusion

In this chapter, we first proposed a bilateral random projections (BRP) based low-rank approximation with fast speed and nearly optimal error bounds. We then develop "Go Decomposition" (GoDec) to estimate the low-rank part L and the sparse part S of a general matrix $X = L + S + G$, wherein G is noise. GoDec is significantly accelerated by using BRP based approximation. The discussions of asymptotic and convergence speeds indicate that GoDec is robust to noise G.

GoDec achieves better decomposition accuracy and much faster speed than existing robust PCA algorithms such as PCP in lots of practical problems. This is because that the sparse outlier and dense noise assumption in GoDec fits real data better than sparse random noise assumption in original robust PCA model. In addition, the alternating projection accelerated by BRP provides a simple and efficient algorithm with very competitively small time cost. However, in the cases where the sparse random noise is more reasonable, PCP might has better accuracy. But according to our empirical study on different real data, GoDec usually outperforms PCP in both accuracy and speed.

Chapter 7

Greedy Bilateral Paradigm and Greedy Low-rank + Sparse Decomposition

Recovering a large low-rank matrix from highly corrupted, incomplete or sparse outlier overwhelmed observations is the crux of various intriguing statistical problems. We explore the power of "greedy bilateral (GreB)" paradigm in reducing both time and sample complexities for solving these problems. GreB models a low-rank variable as a bilateral factorization, and updates the left and right factors in a mutually adaptive and greedy incremental manner. We detail how to model and solve low-rank approximation (GreBske in Chapter 5), matrix completion (GreBcom in Chapter 8) and robust PCA in GreB's paradigm. On their MATLAB implementations, approximating a noisy $10^4 \times 10^4$ matrix of rank 500 with SVD accuracy takes 6s; MovieLens10M matrix of size 69878×10677 can be completed in 10s from 30% of 10^7 ratings with RMSE 0.86 on the rest 70%; the low-rank background and sparse moving outliers in a 120×160 video of 500 frames are accurately separated in 1s. This brings 30 to 100 times acceleration in solving these popular statistical problems.

7.1 Introduction

Explosion of information introduces dramatic increasing data collected from Internet and digital sensors. These data are usually featured by their high-dimension, huge volume, serious incompleteness, dominating noise and complicated structure, which bring intriguing new challenges to compressive acquisition, signal processing and machine learning. Because traditional methods are limited by their time/sample complexities, storage and denoising capability. This fact has driven the recent exploitation of data intrinsic redundancy and structures. For a single signal or individual instance, the redundancy is usually exhibited by its sparsity, i.e., it is nonzero only on a small set of entries. Compressed sensing [71][41] recovers a signal from its highly compressed measurements, via solving a undetermined linear system, by leveraging the sparsity. For multiple instances, or more specifically a matrix, the redundancy is often identified by its low-rank structure, i.e., all the instances lie in a subspace spanned by a small number of bases. Low-rank structure can be analogized to sparsity due to its sparse spectrum. Equivalently saying, the matrix X can be written as the sum of a few rank-1 matrices such that $X = \sum_{i=1}^{r} U_i V_i$, wherein U_i is a column vector and V_i is a row vector.

Low-rank structure arises in a wide range of fundamental problems. In this chapter, we mainly focus on three representatives catching substantial interests in several important applications: low-rank approximation [108], matrix completion [39] and robust PCA [37]. In this chapter, the low-rank matrix variable in these problems is modeled in a bilateral factorization form UV for the purpose of developing SVD-free algorithms.

We describe and analyze a general scheme called "greedy bilateral (GreB)" paradigm for solving the mainstream low-rank matrix recovery problems. GreB starts from U and V respectively containing a very few (e.g., one) columns and rows, and optimizes them alternately. Their updates are based on observation that the object value is determined by the product UV rather than individual U or V. Thus we can choose a different pair (U, V) producing the same UV but computed faster than the one derived by alternating least squares like in IRLS-M [88] and ALS [244]. In GreB, the updates of U and V can be viewed as

mutually adaptive update of the left and right sketches of the low-rank matrix. Such updates are repeated until the object convergence, then a few more columns (or rows) are concatenated to the obtained U (or V), and the alternating updates are restarted on a higher rank. Here, the added columns (or rows) are selected in a greedy manner. Specifically, they are composed of the rank-1 column (or row) directions on which the object decreases fastest. GreB incrementally increases the rank until when UV is adequately consistent with the observations.

GreB's greedy strategy avoids the failures brought by possible biased rank estimation. Moreover, greedy selecting optimization directions from 1 to r is faster than updating r directions in all iterates like in LMaFit [228] and [265]. In addition, the lower rank solution before each rank increment is invoked as the "warm start" of the next higher rank optimization and thus speed up convergence. Furthermore, its mutually adaptive updates of U and V yields a simple yet efficient SVD-free implementation. Under GreB paradigm, the overall time complexity of matrix completion is $\mathcal{O}(\max\{\|\Omega\|_0 r^2, (m+n)r^3\})$ (Ω-sampling set, $m \times n$-matrix size, r-rank), while the overall complexities of low-rank approximation and noisy robust PCA are $\mathcal{O}(mnr^2)$. An improvement on sample complexity can also be justified in our experiments.

7.2 Background and Problem Formulation

For data with sparse outliers or partially contaminated by noise of overwhelming magnitude, sheer low-rank assumption cannot fully capture its complex structure. A more general assumption $X = L + S$ is applied in this case [43], i.e., the data matrix X can be decomposed as the sum of a low-rank matrix L and a sparse matrix S. L explains the components that lie in a subspace and smoothly change across different instances, while S contains the spiky anomalies that are rarely shared by different instances. This model is called "robust PCA" [37] due to its robustness to sparse noise S when recovering principle components of L from X, and can be applied to video surveillance, graphical mode selection, image alignment, multi-label learning [259], etc.

PCP [37] recovers L and S from X by minimizing sum of the trace norm of L and the ℓ_1 norm of S. It can be proved that the solution to this convex relaxation

is the exact recovery if $X = L+S$ indeed exists and L and S are sufficiently incoherent [43][37]. That is, L obeys the incoherence property in (8.1) and thus is not sparse, while S has nonzero entries uniformly selected at random and thus is not low-rank. Popular optimization algorithms such as augmented Lagrangian multiplier, accelerated proximal gradient method and accelerated projected gradient method [46] have been applied. But full SVD as a costly subroutine is required to be repeatedly invoked in any of them.

Despite the strong theoretical guarantee of robust PCA, the exact decomposition $X = L + S$ does not always exist for real data matrix X. Thus a more adaptive model $X = L + S + G$ is preferred, where $L + S$ approximates X and G is the dense noise. Such noisy robust PCA becomes the central interests of many recent works including stable PCP [271], GoDec [265] and DRMF [233]. Direct rank constraint for L and cardinality constraint for S have been employed in order to replace the full SVD with truncated SVD or faster low-rank approximation. They also face the rank estimation problem when determining the rank constraint r.

In this chapter, we formulate the noisy robust PCA by replacing L with its bilateral factorization $L = UV$ and regularizing the ℓ_1 norm of S's entries:

$$\begin{aligned} &\min\nolimits_{U,V,S} \|X - UV - S\|_F^2 + \lambda\|\text{vec}(S)\|_1 \\ &\text{s.t.} \quad rank(U) = rank(V) \leq r. \end{aligned} \tag{7.1}$$

The ℓ_1 regularization induces soft-thresholding in updating S, which is faster than sorting caused by cardinality constraint in GoDec and DRMF.

7.3 Greedy Bilateral (GreB) Paradigm

In this section, we sequentially detail how to solve low-rank approximation, matrix completion and noisy robust PCA in GreB's paradigm. The resulting three algorithms are named as "greedy bilateral sketch (GreBske)", "greedy bilateral completion (GreBcom)" and "greedy bilateral smoothing (GreBsmo)" respectively according to their normal usages in practical applications. We summarize all the three algorithms in GreB's paradigm given in Algorithm 7.

Algorithm 7: Greedy Bilateral (GreB) Paradigm

Input: Object function f; rank step size Δr; power K; tolerance τ; observations of data matrix X
Output: low-rank matrix UV (and sparse S)
Initialize $V \in \mathbb{R}^{r_0 \times n}$ (and S);
while *residual error* $\leq \tau$ **do**
  for $k \leftarrow 1$ **to** K **do**
    Greedy Bilateral Sketch: sequentially compute (5.41);
    Greedy Bilateral Completion: sequentially compute (8.8);
    Greedy Bilateral Smoothing: sequentially compute (7.4);
  end
  Calculate the top Δr right singular vectors v (or Δr-dimensional random projections) of $\partial f / \partial V$ (given in (5.42), (8.9) and (7.5) for different problems);
  Set $V := [V; v]$;
end

7.4 Greedy Bilateral Smoothing

Alternately optimizing U, V and S in (7.1) immediately yields the following updating rules:

$$\begin{cases} U_k = (X - S_{k-1}) V_{k-1}^T \left(V_{k-1} V_{k-1}^T\right)^\dagger, \\ V_k = \left(U_k^T U_k\right)^\dagger U_k^T (X - S_{k-1}), \\ S_k = \mathcal{S}_\lambda (X - U_k V_k), \end{cases} \tag{7.2}$$

where $\mathcal{S}_\lambda$ is an element-wise soft thresholding operator with threshold λ such that

$$\mathcal{S}_\lambda X = \{\operatorname{sgn}(X_{ij}) \max(|X_{ij}| - \lambda, 0) : (i,j) \in [m] \times [n]\}. \tag{7.3}$$

The same trick of replacing the (U, V) pair with a faster computed one is applied and produce

$$\begin{cases} U_k = Q, \operatorname{QR}\left((X - S_{k-1}) V_{k-1}^T\right) = QR, \\ V_k = Q^T (X - S_{k-1}), \\ S_k = \mathcal{S}_\lambda (X - U_k V_k), \end{cases} \tag{7.4}$$

The above procedure can be performed in $3mnr_i + mr_i^2$ flops for $U \in \mathbb{R}^{m \times r_i}$ and $V \in \mathbb{R}^{r_i \times n}$.

In GreBsmo, (7.4) is iterated as a subroutine of GreB's greedy incremental paradigm. In particular, the updates in (7.4) are iterated for K times or until the object converging, then Δr rows are added into V as the new directions for decreasing the object value. In order to achieve the fastest decreasing directions, we greedily select the added Δr rows as the top Δr right singular vectors of the partial derivative

$$\frac{\partial \|X - UV - S\|_F^2}{\partial V} = X - UV - S. \tag{7.5}$$

We also allow to approximate row space of the singular vectors via random projections [108]. The selected Δr rows maximize the magnitude of the above partial derivative and thus lead to the most rapid decreasing of the object value, a.k.a., the decomposition error. GreBsmo repeatedly increases the rank until a sufficiently small decomposition error is achieved. So the rank of the low-rank component is adaptively estimated in GreBsmo and does not relies on initial estimation.

We report the phase diagram of GreBsmo in Figure 7.1 from results on randomly generated matrix that is the sum of a low-rank part and a sparse part. The low-rank part is generated as the product of two Gaussian matrices and the sparse part has a Bernoulli model generated support set on which ± 1 values are randomly assigned. The phase transition phenomenon is in consistency with existing low-rank and sparse decomposition algorithms. It also shows that GreBsmo is able to gain accurate separation of L even if its rank is close to $0.4n$, given the sparse part has an adequately sparse support set. This is competitive to published result [37]. Interestingly, the phase transition curve has a regular shape and implies a theoretical analysis to its behavior is highly possible in future studies.

7.5 Analysis

It is not direct to analyze the theoretical guarantee of GreB due to its combination of alternating minimization and greedy forward selection. Hence, we consider analyzing its convergence behavior by leveraging the results from GECO [200]

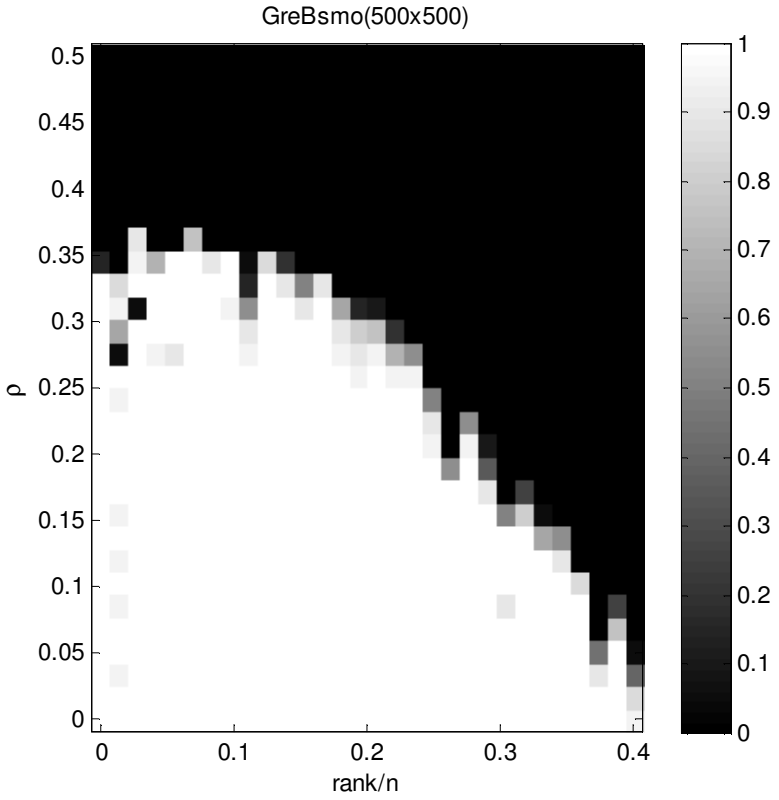

Figure 7.1: Phase diagram for GreBsmo on 500×500 matrices. Low-rank component is generated as $L = UV$, where entries of U and V are sampled from $\mathcal{N}(0, 1/n)$. Entries of sparse component S are sampled as 1 or -1 with probability $\rho/2$ and 0 with probability $1 - \rho$. On the 30×30 grid of sparsity-rank/n plane, 20 trials are performed for each (ρ, r) pair. L is said to be successfully recovered if its rel. err.$\leq 10^{-2}$. The phase diagram shows the successful recovery rate for each (ρ, r) pair.

analysis. This is reasonable because they share the same objective function yet different optimization variables. In particular, the risk function in GECO is $R(A) = R(A(\lambda)) = f(\lambda)$, where $A = \sum_i \lambda_i U_i V_i$. It can be seen that the variable A in GECO is able to be written as $A = UV$ without any loss of generality. Therefore, for the same selection of $R(A)$, we can compare the objective value of GECO and GreB at arbitrary step of their algorithm. This results in the following theorem.

Theorem 17. *Assume $R(A)$ is a β-smooth function according to GECO [200] and $\epsilon > 0$, and $F(U, V) = R(UV)$ is the objective function of GreB. Given a rank*

constraint r to A and a tolerance parameter $\tau \in [\,0, 1\,)$. Let $A^* = U^*V^*$ is the solution of GreB. Then for all matrices $A = UV$ with

$$\|UV\|_{tr}^2 \leq \frac{\epsilon(r+1)(1-\tau)^2}{2\beta} \tag{7.6}$$

we have $F(U^*, V^*) \leq F(U, V) + \epsilon$.

Proof. According to Lemma 3 in GECO [200], let $\epsilon_i = f(\lambda^{(i)}) - f(\bar{\lambda})$, where $\lambda^{(i)}$ is the value of λ at the beginning of iteration i and $\bar{\lambda}$ fulfills $f(\lambda) > f(\bar{\lambda})$, we have

$$f(\lambda^{(i)}) - \min_{\eta} f(\lambda^{(i)} + \eta e^{u,v}) \geq \frac{\epsilon_i^2(1-\tau)^2}{2\beta\|A\|_{tr}^2}. \tag{7.7}$$

At the end of iteration i, the objective value of GreB equals $R(UV)$, while GECO optimizes λ over the support of $\text{span}(U) \times \text{span}(V)$ (i.e., optimizes S when fixing U and V). We use the same notation $\cdot^{(i)}$ to denote the variable in iteration i. This yields

$$\begin{aligned} F(U^{(i)}, V^{(i)}) = \quad & R(U^{(i)}V^{(i)}) \geq \\ & \min_{S} R(U^{(i)}SV^{(i)}) = f(\lambda^{(i)}). \end{aligned} \tag{7.8}$$

At the beginning of iteration $i+1$, both GECO and GreB computes the direction (u, v) along which the object declines fastest. However, GECO adds both u and v to the ranges of U and V, while GreB only adds v to V and then optimizes U when fixing V. Because the range of U in GreB is optimized rather than previously fixed, we have

$$\begin{aligned} F(U^{(i+1)}, V^{(i+1)}) = \quad & \min_{U} F(U, [V^{(i+1)}; v]) \leq \\ & \min_{\eta} f(\lambda^{(i)} + \eta e^{u,v}). \end{aligned} \tag{7.9}$$

Plug (7.8) and (7.9) into (7.7), we gain a similar result:

$$F(U, V) - \min_{U} F(U, [V; v]) \geq \frac{\epsilon_i^2(1-\tau)^2}{2\beta\|A\|_{tr}^2}. \tag{7.10}$$

Following the analysis after Lemma 3 in GECO [200], we can immediately obtain the results of the theorem. □

The theorem states that GreB solution is at least close to optimum as GECO. Note when sparse S is alternatively optimized with UV in GreB scheme, such as GreBcom, the theorem can still holds. This is because after optimizing S in each iteration of GreBcom, we have $\mathcal{P}_{\Omega^C}(S+UV)=0$, which enforces the objective function $\|M-UV-S\|_F^2$ degenerates to that of GECO, which is $\|P_\Omega(M-UV)\|_F^2$.

7.6 Experiments on Video Data

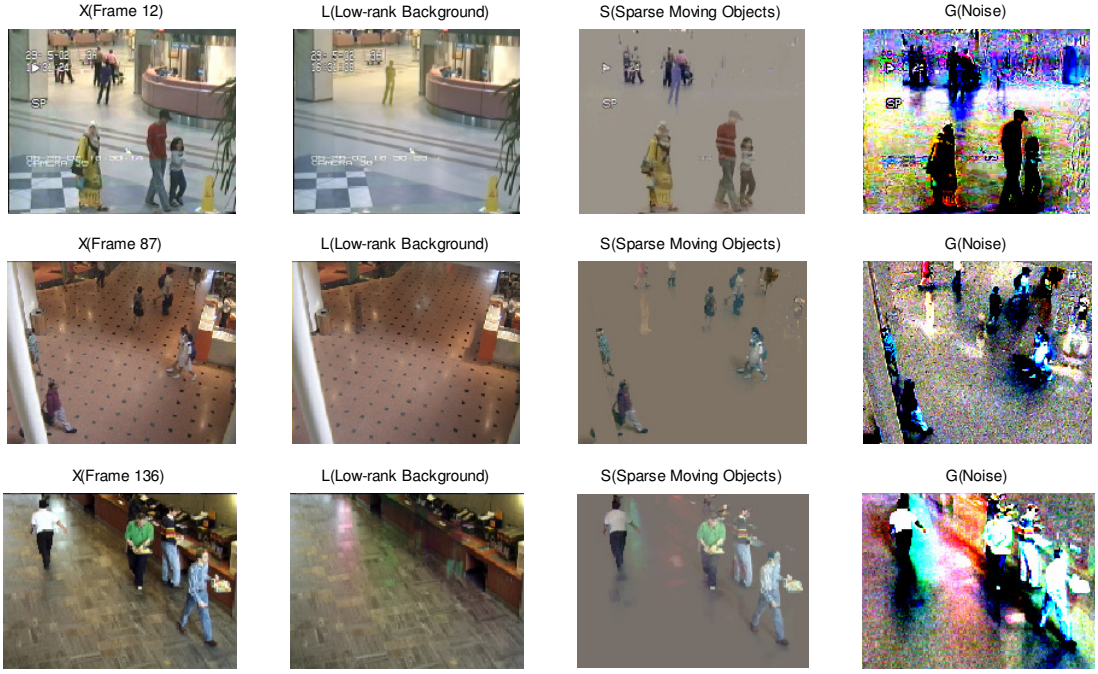

Figure 7.2: Background modeling of GreBsmo on three video sequences, top row: Hall, 144×176 pixels, 500 frames; middle row: ShoppingMall, 256×320 pixels, 253 frames; bottom row: Boostrap, 120×160 pixels, 500 frames.

In this section, we show how to use the above three algorithms developed from GreB to solve robust PCA by justifying their performance on real datasets. For simplicity, we fix the times of rank increment in GreB as 5, which implies $\Delta r = \max\{1, \lfloor rank/5 \rfloor\}$. All the experiments are performed on MATLAB.

For real data, three robust PCA algorithms, i.e., inexact augmented Lagrangian multiplier method for PCP, GoDec and GreBsmo are applied to sep-

Table 7.1: Comparison of time costs in CPU seconds of PCP, GoDec and GreBsmo in low-rank and sparse matrix decomposition task on background modeling datasets.

	PCP	GoDec	GreBsmo
Hall	$87s$	$56s$	$1.13s$
ShoppingMall	$351s$	$266s$	$3.29s$
Bootstrap	$71s$	$49s$	$0.98s$

arate the low-rank background and sparse moving objects in 3 video sequences [1]. Pixel values of each frame in the video are vectorized as a row vector in a matrix X, and the background modeling can be modeled as performing robust PCA on the matrix. We show the robust PCA decomposition results of one frame for each video sequence obtained by GreBsmo in the left plot of Figure 7.6. The decomposition includes a low-rank background, a sparse component containing moving objects, and a dense noise part. The time costs for all the three methods are listed in Table 7.1. It shows GreBsmo considerably speed up the decomposition and performs 30-100 times faster than most existing algorithms.

[1] http://perception.i2r.a-star.edu.sg/bk_model/bk_index.html

Chapter 8

Randomized and Greedy Algorithms for Matrix Completion

In this chapter, we propose two variants of GoDec in Chapter 6 and GreB in Chapter 7 for solving low-rank matrix completion problem.

8.1 Introduction to Low-rank Matrix Completion

Data is usually obtained with many missing values. The goal of matrix completion [39][36] is to recover the whole data matrix X from partial noisy entries or undersampled linear measurements $\mathcal{A}(X)$ ($\mathcal{A}$ is the sampling operator) by leveraging the connections between different instances. Such connections can be well captured by low-rank structure. Matrix completion has broad applications in various real problems of considerable importance, such as collaborative filtering in recommendation system [272] link prediction for social network, quantum state tomography and traffic distance completion. The matrix completion problem was firstly written as a rank minimization that is NP-hard both to solve and approximate. Thus trace norm (ℓ_1 norm of singular values) as the convex surrogate of rank has been minimized in many popular approaches [35][132]. For a matrix

$X \in \mathbb{R}^{m \times n}$ of rank-r with SVD decomposition $X = B\Sigma D$, the obtained global solution is provable to exactly recover X from $\mathcal{O}(nr \log^6 n)$ uniformly sampled noisy entries if it fulfills incoherence property ($\mathcal{P}_B \cdot$ denotes the orthogonal projection onto B and e_i is standard basis)

$$\max_{1 \le i \le m} \|\mathcal{P}_B e_i\|^2 \le \mu_0 r/m, \max_{1 \le i \le n} \|\mathcal{P}_D e_i\|^2 \le \mu_0 r/n, \tag{8.1}$$

entries are upper bounded as $\max_{i,j} |X_{ij}| \le \mu_1 \sqrt{r}/\sqrt{mn}$, and $\mathcal{A}$ obeys matrix restricted isometry property (RIP) with constant δ

$$(1-\delta)\|X\|_F^2 \le \|\mathcal{A}(X)\|_F^2 \le (1+\delta)\|X\|_F^2. \tag{8.2}$$

Another norm receiving much attention for encouraging low-rank solution is max-norm $\|X\|_{max} = \inf\{\|U\|_{2,\infty}\|V^T\|_{2,\infty} : X = UV\}$ [204][151] ($\|\cdot\|_{2,\infty}$ is the maximum ℓ_2 row norm of a matrix), whose theoretical recovery bound is established based on Rademacher complexity of its unit ball [89]. The max-norm can be defined as the global solution to an non-convex optimization on left and right factors U and V.

Both trace norm minimization and max-norm minimization can be formulated as semidefinite programming (SDP) which has standard solvers. Various accelerated optimization methodologies [168][151] also have been applied to them for practical purpose. However, most of them rely on costly computation of full SVD in each iterate, and thus do not scale well to large-scale problems. This fact induces a revisit of rank minimization/constraint based formulations. Forward greedy selection methods including ADMiRA [152] and GECO [200] are developed for rank minimization. GECO adopts an interesting greedy strategy: it increments the rank by 1 and adds the optimal rank-1 direction into the optimization per iterate. Incremental OptSpace [137] also has a similar scheme. We inherit a similar spirit but solve different optimization models in GreB. Error minimization with rank constraint is also considered in SVP [129]. Some of them like OptSpace and GECO model the low-rank variable as factorization USV and optimize over (U, V) pair and S. Unfortunately, truncated SVD or large matrix multiplication is still required in these approaches. In addition, the update of S

is to solve a large-scale overdetermined linear system and thus time consuming in practice. Furthermore, the rank is fixed within some algorithms, so in this case the recovery accuracy strongly relies on the quality of rank estimation. Online/stochastic gradient method [12] and aggregated (divide-and-conquer) method [170] have also been considered for pursuing approximated recovery. Since GreB can be straightforwardly transferred to or invoked as a subroutine by them to take their advantages, they will not be included in later comparison.

8.2 GoDec for matrix completion

8.2.1 Model and Algorithm

We consider the problem of exactly completing a low-rank matrix X with $\text{rank}(X) \leq r$ from a subset of its entries $Y = \mathcal{P}_\Omega(X)$, wherein Ω is the sampling index set. Different from the two conventional methods, nuclear norm minimization [36] and low-rank subspace optimization on Grassmann manifold [137], we formulate the matrix completion problem as a rank constrained optimization:

$$\begin{aligned} \min_{X,Z} \quad & \|Y - X - Z\|_F^2 \\ s.t. \quad & \text{rank}\,(X) \leq r, \\ & \text{supp}\,(Z) = \overline{\Omega}, \end{aligned} \tag{8.3}$$

where Z is an estimate of $-\mathcal{P}_{\overline{\Omega}}(X)$. Therefore, Godec algorithms can be extended to solve (8.3) after the following two slight modifications.

- Replacing X, L and S in Algorithm 6 with Y, X and Z, respectively.
- Replacing the entry set Ω used in the last step of Algorithm 6 with $\overline{\Omega}$, wherein Ω is the sampling index set in matrix completion.

The same as GoDec, its extension (8.3) for solving the matrix completion problem converges to a local optimum. Compared with the nuclear norm minimization methods, (8.3) is more efficient because it does not require time consuming SVD for X. Compared with the subspace optimization methods, GoDec avoids the unstableness and the local barriers of the optimization on Grassmann manifold.

Moreover, GoDec is parameter free (both the rank range r and the tolerance ϵ are predefined parameters) and thus it is easier to use compared with existing methods.

8.2.2 Matrix Completion Experiments of GoDec

We test the performance of GoDec in matrix completion tasks. Each test low-rank matrix is generated by $X = AB$, wherein both A and B are $n \times r$ standard Gaussian matrices. We randomly sample a few entries from X and recover the whole matrix by using Algorithm 6 (after the two modifications presented in Section 8.1.1). The experimental results are shown in Table 8.1. Compared with the published results [137] of the popular matrix completion methods, e.g., OptSpace, SVT, FPCA and ADMIRA, GoDec requires both less computational time and less samples to recover a low-rank matrix.

Table 8.1: Relative error and time cost of OptSpace and GoDec in matrix completion tasks. The results separated by "/" are SVT [35] (a nuclear norm minimization method), OptSpace [137] (a subspace optimization method on Grassmann manifold) and GoDec, respectively. See [137] for the results of the other methods, e.g., FPCA and ADMIRA.

size(X) (square)	rank(X) (1)	sampling rate (%)	rel.error(X) (10^{-5})	time (seconds)
1000	10	0.12/0.12/0.075	1.68/1.18/1.77	40/28/15.43
	50	0.39/0.39/0.18	1.62/0.92/1.11	247/212/26.36
	100	0.57/0.57/0.3	1.71/1.49/1.24	694/723/43.47
5000	10	0.024/0.024/0.021	1.76/1.51/1.39	112/252/300.96
	50	0.1/0.1/0.084	1.62/1.16/1.48	1312/850/415.96
	100	0.16/0.16/0.12	1.73/0.83/1.09	5432/3714/551.95
10000	10	0.012/0.012/0.04	1.75/0.76/0.50	221/632/1101.83
	50	0.05/0.05/0.045	1.63/1.19/1.17	2872/2585/1172.68
	100	0.08/0.08/0.075	1.76/1.46/1.84	10962/8514/1505.93

8.3 Greedy Bilateral Completion (GreBcom)

8.3.1 Model and Algorithm

In this chapter, we optimize the bilateral factorization form $X = UV$ of low-rank matrix X and address a rank constrained optimization:

$$\begin{aligned} &\min_{U,V,S} \|M - UV - S\|_F^2 \\ &\text{s.t. } \ rank(U) = rank(V) \leq r, P_\Omega S = 0, \end{aligned} \tag{8.4}$$

where $\mathcal{P}_\Omega \cdot$ denotes the projection of a matrix on an element subset $\Omega \subset [m] \times [n]$

$$\mathcal{P}_\Omega X = \begin{cases} X_{ij}, & (i,j) \in \Omega \\ 0, & (i,j) \in \Omega^C \end{cases} \tag{8.5}$$

and M consists of the observed entries and defined as $M = P_\Omega X$.

8.3.2 Greedy Bilateral Completion

Following a similar methodology as GreBske, alternately optimizing U, V and S in (8.4) immediately yields the following updating rules:

$$\begin{cases} U_k = (M - S_{k-1}) V_{k-1}^T \left(V_{k-1} V_{k-1}^T\right)^\dagger, \\ V_k = \left(U_k^T U_k\right)^\dagger U_k^T (M - S_{k-1}), \\ S_k = \mathcal{P}_\Omega^C (M - U_k V_k), \end{cases} \tag{8.6}$$

Note the the object value in (8.4) is solely determined by the matrix product UV and S rather than individual U or V, we use the same trick of replacing the (U, V) pair with a faster computed one as we did in GreBske. This yields

$$\begin{cases} U_k = Q, \text{QR}\left((M - S_{k-1}) V_{k-1}^T\right) = QR, \\ V_k = Q^T (M - S_{k-1}), \\ S_k = -\mathcal{P}_\Omega^C (U_k V_k). \end{cases} \tag{8.7}$$

Since $M - S_{k-1} = M - \mathcal{P}_\Omega(U_{k-1}V_{k-1}) + U_{k-1}V_{k-1}$, define the residual $E = M - \mathcal{P}_\Omega(U_{k-1}V_{k-1})$, the above procedure (8.7) can be further accelerated as

$$\begin{cases} U_k = Q, \mathrm{QR}\left(E_{k-1}V_{k-1}^T + U_{k-1}\left(V_{k-1}V_{k-1}^T\right)\right) = QR, \\ V_k = Q^T E_{k-1} + \left(Q^T U_{k-1}\right) V_{k-1}, \\ E_k = M - \mathcal{P}_\Omega\left(U_k V_k\right). \end{cases} \tag{8.8}$$

It is not hard to verify that computation of the above procedure requires $3|\Omega|_0 r_i + (3m + 2n)r_i^2$ flops for $U \in \mathbb{R}^{m\times r_i}$ and $V \in \mathbb{R}^{r_i \times n}$.

Similar to GreBske, GreBcom adopts a greedy strategy incrementally changing r_i to $r_i + \Delta r$ by concatenating new rows to V after executing K times of updates in (8.8). The number of iterating (8.8) can also be determined by the convergence of object, which implies a successful tracking of the first r_i basis within the row space of low-rank matrix X. The greedy selection of the Δr new rows is based on the partial derivative of the object w.r.t. UV

$$\frac{\partial \|M - UV - S\|_F^2}{\partial UV} = M - UV - S = E. \tag{8.9}$$

The Δr new rows are chosen as the Δr right singular vectors of $U^T E$ associated with the Δr largest singular values, which can be approximated by the random projections $A^T U^T E$ wherein A is a $r_i \times \Delta r$ random matrix [108]. This greedy selection enables the object decreasing fastest along the newly added Δr basis in the next round of updates (8.8). GreBcom starts from a small r_0 and increases the rank until reaching certain tolerance for the residual, when the final rank r of X is determined automatically. As well as GreBske, such greedy incremental scheme brings evident improvement in speed.

We report the phase diagram of GreBcom in Figure 8.1. The white region denotes where GreBcom can successfully recover the incomplete low-rank matrix with probability of 1, while the black region is where the algorithm fails. It can be seen that GreBcom exhibits evident phase transition property, which is a significant phenomenon verified in many matrix completion algorithms. Compared to the published phase transition plots of them, GreBcom possesses a larger region for successful recovery on the ρ-r plane.

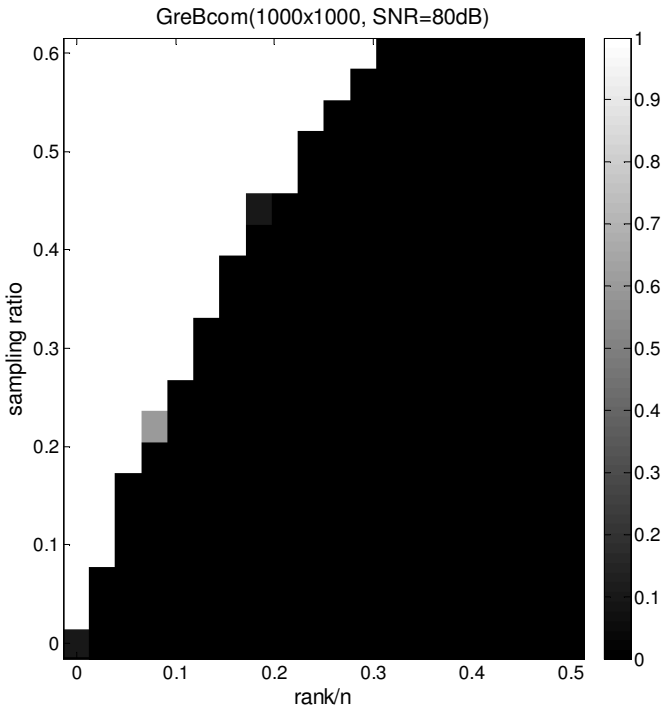

Figure 8.1: Phase diagram for GreBcom on 1000×1000 matrices. On the 20×20 grid of sampling ratio-rank/n plane, 10 trials are performed for each (ρ, r) pair. A matrix is said to be successfully recovered if rel. err.$\leq 10^{-3}$. The phase diagram shows the successful recovery rate for each (ρ, r) pair.

8.3.3 Matrix Completion Experiments of GreBcom

The performance of GreBcom in matrix completion tasks are evaluated and compared with other approaches on both artificial generated data and real data from movie (MovieLens) and joke (Jester) recommendation systems.

Numerical results on artificial data. We generate low-rank matrix $X \in \mathbb{R}^{n \times n}$ by $X = UV + Z$, wherein $U \in \mathbb{R}^{n \times r}$ and $V \in \mathbb{R}^{r \times n}$ are random matrices whose entries are sampled from i.i.d. normal distribution $\mathcal{N}(0, 1)$ and each entry of noise Z is sampled from $\mathcal{N}(0, 10^{-10})$. The observed entries are uniformly selected with probability ρ. OptSpace, SVP, ADMiRA and GreBcom are performed on large-scale matrices of $n \in [5000, 30000]$ and $r \in [10, 100]$. Their completion accuracy is measured by relative error $\|\hat{X} - X\|_F / \|X\|_F$ wherein $\hat{X}$ is the low-rank recovery, while the time cost is measured by CPU time. The results are given in Table 8.2. It shows that GreBcom achieves the lowest relative error and

Table 8.2: Relative error and time cost of OptSpace, SVP, ADMiRA and GreBcom in matrix completion tasks of different matrix size and rank. Notations: $m(n)$-square matrix size, r-rank, ρ-sampling ratio $|\Omega|_0/mn$, rel. err.-relative error, time-CPU time, "-"-does not apply due to speed or divergence.

$m(n)$	r	ρ	OptSpace		SVP		ADMiRA		GreBcom	
			rel. err.	time	rel. err.	time	rel. err.	time	rel. err.	time
5000	10	0.01	2.29×10^{-2}	304	8.51×10^{-1}	86	5.13×10^{-1}	77	2.01×10^{-2}	0.73
	50	0.04	3.41×10^{-2}	1582	6.95×10^{-1}	1362	5.36×10^{-1}	358	3.06×10^{-2}	16
	100	0.08	5.64×10^{-2}	3944	7.01×10^{-1}	24519	4.84×10^{-1}	36266	2.38×10^{-3}	116
10000	10	0.01	1.34×10^{-2}	516	4.54×10^{-1}	1322	1.22×10^{-1}	442	1.55×10^{-3}	2.17
	50	0.04	1.19×10^{-2}	2192	2.35×10^{-1}	5961	2.58×10^{-2}	186591	1.40×10^{-3}	49
	100	0.08	2.64×10^{-3}	15910	-	-	9.66×10^{-2}	755082	1.20×10^{-3}	153
20000	10	0.006	7.06×10^{-2}	1928	-	-	3.04×10^{-1}	181	1.20×10^{-3}	4.06
	50	0.025	7.66×10^{-3}	11397	-	-	4.33×10^{-2}	346651	1.20×10^{-3}	113
30000	10	0.006	8.29×10^{-2}	6121	2.43×10^{-1}	2235	4.19×10^{-1}	71	1.20×10^{-3}	18

brings substantial acceleration. The time cost of GreBcom slowly augmented with the increasing of matrix size, which indicates its promising scalability for solving large-scale problems.

Collaborative filtering. We now apply different matrix completion algorithms to real datasets MovieLens [1] and Jester [2] respectively collected from movie and joke recommendation systems. Each of the two datasets contains 3 matrices, whose rows denote users, columns denote items and the entry values are associated ratings. Ratings in MovieLens are integers between 1 and 5, while ratings in Jester have values in $[-10, 10]$. For MovieLens, we randomly select $\{10\%, 30\%, 50\%\}$ of the given observations as the training set observable to the matrix completion algorithms, while the rest are left for test set. For Jester, we randomly select $\{2, 5, 10\}$ ratings from each user as test set and treat the others as training instances. The completion accuracy is evaluated by comparing the difference between the given matrix and the completed one on the test set. In Table 8.3, the difference is measured by root mean square error (RMSE) of the test set such that $RMSE_{test} = \sqrt{\text{mean}(\hat{X}_{ij} - X_{ij}), (i,j) \in \text{test set}}$, while the CPU time indicates the required computation time. In these experiments, GreBcom exhibits evident priority in both completion accuracy and time cost over other methods. In addition, less observations are required in GreBcom to reach

[1] http://www.grouplens.org/node/73

[2] http://www.ieor.berkeley.edu/~goldberg/jester-data/

Table 8.3: $RMSE_{test}$/CPU time of OptSpace, SVP and GreBcom in matrix completion tasks on recommendation system data with different training set ratio (for MovieLens) or different number of test ratings per user (for Jester), "-"-does not apply due to speed or divergence. Size and rank information (m/n/r) of datasets: 100k(943/1682/3), 1M(6040/3952/10), 10M(69878/10677/10), J1(24983/100/10), J2(23500/100/10), J3(24938/100/10).

	OptSpace		
Movie	10%	30%	50%
100k	3.13/334s	2.82/394s	1.02/117s
1M	1.35/2241s	0.93/2550s	0.89/3383s
10M	0.92/9021s	-	-
Jester	2	5	10
J1	4.10/756s	4.10/732s	4.13/669s
J2	4.14/1058s	4.14/640s	4.16/840s
J3	4.57/340s	4.64/261s	4.51/99.31s
	SVP		
Movie	10%	30%	50%
100k	1.23/5.01s	1.06/2.19s	0.97/2.35s
1M	1.12/38s	0.98/41s	0.92/32s
10M	0.96/588s	0.87/464s	0.84/694s
Jester	2	5	10
J1	5.31/41.26s	5.09/40.12s	4.24/32.54s
J2	4.24/320s	4.23/22.14s	4.31/66.74s
J3	7.75/14.71s	7.28/13.27s	6.13/6.85s
	GreBcom		
Movie	10%	30%	50%
100k	1.01/0.04s	0.98/0.04s	0.97/0.04s
1M	0.96/1.15s	0.90/1.22s	0.89/1.89s
10M	0.88/9.61s	0.86/10.13s	0.82/25.65s
Jester	2	5	10
J1	4.01/7.29s	4.06/7.88s	4.08/6.28s
J2	4.12/10.23s	4.18/8.80s	4.09/10.06s
J3	4.91/3.15s	4.43/3.02s	4.38/1.20s

a sufficiently small RMSE, which support the effectiveness of GreBcom in real applications.

Chapter 9

Three GoDec Variants Unmixing General Incoherent Structures

Learning big data by matrix decomposition always suffers from expensive computation, mixing of complicated structures and noise. In Chapter 6, we introduce "GO decomposition (GoDec)", an alternating projection method estimating the low-rank part L and the sparse part S from data matrix $X = L + S + G$ corrupted by noise G. Two acceleration strategies are proposed to obtain scalable unmixing algorithm on big data: 1) Bilateral random projection (BRP) in Chapter 5 is developed to speed up the update of L in GoDec by a closed-form built from left and right random projections of $X - S$ in lower dimensions; 2) Greedy bilateral (GreB) paradigm in Chapter 7 updates the left and right factors of L in a mutually adaptive and greedy incremental manner, and achieve significant improvement in both time and sample complexities.

In this chapter, we attempt to solve three real application problems by more adaptive models and efficient algorithms that decompose a data matrix as the sum of semantic components with incoherent structures, which are not necessary to be low-rank or sparse. The three application problems include motion segmentation, multi-label learning and recommendation system with side information. Although these three problems are considerably different from each other, it turns out that a new insight of unmixing incoherent structures can be applied to develop effective algorithms for these problems.

In particular, we propose three nontrivial variants of GoDec that generalizes GoDec to more general data type and whose fast algorithms can be derived from the two strategies: 1) for motion segmentation, we further decompose the sparse S (moving objects) as the sum of multiple row-sparse matrices, each of which is a low-rank matrix after specific geometric transformation sequence and defines a motion shared by multiple objects; 2) for multi-label learning, we further decompose the low-rank L into subcomponents with separable subspaces, each corresponds to the mapping a single label in feature space. Then the prediction can be effectively conducted by group lasso on the subspace ensemble; 3) for estimating scoring functions of each user in recommendation system, we further decompose the low-rank L as WZ^T, where the rows of W is the linear scoring functions and the rows of Z are the items represented by available features. Empirical studies show the efficiency, robustness and effectiveness of the proposed methods in real applications.

9.1 Introduction

Complex data is usually generated by mixing several components of different structures. These structures are often compressible, and are able to provide semantic interpretations of the data content. In addition, they can reveal the difference and similarity among data samples, and thus produce robust features playing vital roles in supervised or unsupervised learning tasks. Two types of structures have drawn lots of research attentions in recent years: 1) in compressed sensing [71, 41], a sparse signal can be exactly recovered from its linear measurements at a rate significant below the Nyquist rate, in sparse coding [1, 142, 150], an overcomplete dictionary leads to sparse representations for dense signals of the same type; 2) in matrix completion [39, 36, 137, 132, 129], a low-rank matrix can be precisely rebuilt from a small portion of its entries by restricting the rows (samples) to lie in a subspace. In dimension reduction [119, 87, 196, 210, 250, 269], low-rank structure [238] has been broadly leveraged for exploring the geometry of point cloud. Although sparse and low-rank structures have been studied separately by a great number of researchers for years, the linear combination of them or their extensions is rarely explored until recently [43, 37, 121, 46]. Intuitively,

fitting data with either sparse or low-rank structure is mature technique but is inevitably restricted by the limited data types they can model, while recent study shows that the linear mixture of them is more expressive in modeling complex data from different applications.

A motivating example is robust PCA [37] (RPCA), which decomposes the data matrix X as $L + S$. The low-rank part L summarizes a subspace that is shared by all the samples and thus reveals the global smoothness, while the sparse part S captures the individual differences or abrupt changes among samples. A direct application of robust PCA is separating the sparse moving objects from the low-rank background in video sequence. Another interesting example is morphological component analysis (MCA) [25], which decompose the data into two parts that have sparse representations on two incoherent over-complete dictionaries, i.e., the first part has a very non-sparse representation on the dictionary of the second part, and vise versa. This requirement suggests that the two parts are separable on their sparse representations. Note that both RPCA and MCA can only work on data whose two building parts are incoherent, i.e., the content of one part cannot be moved to the other part without changing either of their structures (low-rank, sparse, dictionary, etc.). This incoherence condition could be viewed as a general extension of the statistical independence supporting independent component analysis (ICA) [55, 126] blindly separating non-Gaussian source signals. It leads to the identifiability of the structures in theory, and is demonstrated to be fulfilled on a wide class of real data.

However, new challenges arises when many recent studies tend to focus on big data with complex structures. Firstly, existing algorithms are computationally prohibitive to processing these data. For instance, the update of low-rank part in RPCA and in its extensions invoke a full singular value decomposition (SVD) per iterate, while MCA requires challenging ℓ_0 or ℓ_1 minimization per sample/feature and previously achieved incoherent dictionaries/transform operators encouraging sparse representations. Thus they suffer from a dramatic growth in time complexity when either feature dimensions or data samples increase. In previous methods, the structured information such as low-rank and sparse properties are always achieved at the price of time-consuming optimization, but are rarely leveraged for the purpose of improving the scalability. Recent progresses

in randomized approximation and rank-revealing algorithms shed some light on the speedup of the robust PCA typed algorithms: the subspace of the low-rank part can be estimated from random sampling of its columns/rows or projections of its columns/rows on a random ensemble with bounded precision [108, 53, 2]. However, straightforward invoking this technique in RPCA problem needs to apply it to the updated residual matrix per iterate and thus may lead to costly computation. Besides, determining the rank of the low-rank part is not a trivial problem in practice.

Secondly, the simple low-rank, sparse and sparse representation assumptions cannot fully capture the sophisticated relation, individuality and sparsity of data samples with complex structures. While low-rank structure summarizes a global linear relationship between data points, the nonlinear relationship, local geometry and correlated functions are more common in big data and more expressive for a much wider class of structures. Moreover, the sparse matrix is simply explained by random noises on random positions in the past, but current studies reveal that it may have rich structured information that could be the central interests of various applications. For instance, the sparse motions captured by RPCA on video sequence data includes immense unexplored information favored by object tracking and behavior analysis. Furthermore, although the sparse representation is more general than sparse features, its quality largely relies on whether the given dictionary or transform operator fits the nature of data well. But this is difficult to evaluate when the data is of large volume and in general type.

Thirdly, two building parts are not sufficient to cover all the mixtures of incoherent structures in big data. One the one hand, dense noise is an extra component that has to be separated from the low-rank and sparse parts in many cases where the exact decomposition $X = L + S$ does not hold. This noisy assumption has been considered in stable PCP [271], DRMF [233] and other theoretical studies [121], and its robustness and adaptiveness to a broad class of data has also been verified. But efficient algorithm for the noisy model lacks. On the other hand, further decomposing the low-rank or sparse part to multiple distinguishable sub-components is potential to tell locally spatial or temporal relations within each identifiable structure and differences between them, which usually play pivot roles in supervised and unsupervised learning tasks. Although

it appeals to be a natural extension to the two-part model in RPCA, how to formulate a proper decomposition model for learning problems and develop a practical algorithm are challenging.

9.2 Main Contributions of This Chapter

Although the randomized and greedy strategies proposed in GoDec (Chapter 6) and GreB (Chapter 7) successfully generate efficient low-rank and sparse decomposition capable to tackle large volume problem of big data, the complicated structures widely existing in big data cannot be always expressed by the sum of low-rank and sparse matrices and thus may still lead to the failure of RPCA typed models. Therefore, we address this problem by developing several GoDec's variants that unravel different combination of incoherent structures beyond low-rank and sparse matrices, where the two strategies can be still used to achieve scalable algorithms.

The first variant "shifted subspace tracking (SST)" [260] is developed for motion segmentation [231, 3, 90, 113, 236] from raw pixels of video sequence. SST further analyzes the unexplored rich structure of the sparse part S of GoDec, which could be seem as a sum mixture of several motions with distinct appearance and trajectories. In particular, SST decomposes S of GoDec into the sum of several matrices, each of whose rows are generated by imposing a smooth geometric transformation sequence to the rows of a low-rank matrix. These rows store moving object in the same motion after aligning them across different frames, while the geometric transformation sequence defines the shared trajectories and deformations of those moving objects across frames. We develop an efficient randomized algorithm extracting the motions in sequel, where the low-rank matrix for each motion is updated by BRP [263] in Chapter 5, and the geometric transformation sequence is updated in a piece-wise linear approximation manner. SST is an efficient unsupervised framework to detect, track and segment multiple motions in complex scenes via solving a simple matrix factorization model. Moreover, SST admits other nonlinear transformation on low-rank matrix and thus can explore the mixture structure of sparse component on other data types rather then video.

The second variant "multi-label subspace ensemble (MSE)" [259] extends the low-rank part L of GoDec to the sum of multiple low-rank matrices defined by distinguishable but correlated subspaces. MSE provides a novel insight into the multi-label learning (ML) problem [220, 218, 216, 187, 217], which aims at predicting multiple labels of a data sample. Most previous ML methods [193, 20, 253, 134, 219, 49] focus on training effective classifiers that establishes a mapping from feature space to label space, and take the label correlation into account in the training process. Because it has been longly believed that label correlation is useful for improving prediction performance. However, in these methods, both the label space and the model complexity will grow rapidly when increasing the number of labels and simultaneously modeling their joint correlations. This usually makes the available training samples insufficient for learning a joint prediction model.

MSE eliminates this problem by jointly learning inverse mappings that map each label to the feature space as a subspace, and formulating the prediction as finding the group sparse representation [243] of a given sample on the ensemble of subspaces. In the training stage, the training data matrix X is decomposed as the sum of several low-rank matrices and a sparse residual via a randomized optimization. Each low-rank part defines a subspace mapped by a label, and its rows are nonzero only when the corresponding samples are annotated by the label. The sparse part captures the rest contents in the features that cannot be explained by the labels. In MSE, there are only k subspaces needed to be learned, and the label correlations are fully used via considering correlation among subspaces.

The third variant "linear functional GoDec (LinGoDec)" treats the data matrix X as a rating matrix whose rows index the users, columns index the items, and entries denote the scores of items given by different users. A key problem in recommendation system is that given the features of some items, which are usually available, and the ratings of these items scored by all users, how to learn a scoring function for each user so that effective prediction of ratings can be made on new item. LinGoDec studies the case when all the scoring functions of different users are linear and related to each other. In particular, it extends the low-rank part L of GoDec to WZ^T, where W represents the linear functions and is constrained to be low-rank, while the rows of Z contain the features of

items in the training set. In addition, the sparse part S is able to detect the advertising effects or anomaly of users' ratings on specific items, because these cannot be represented by the low-rank scoring functions. LinGoDec imposes a special structured restriction to the low-rank part L. It formulates the collaborative filtering problem as supervised learning, and thus avoids time-consuming completion of the whole matrix when only a new item's scores (a new row) are needed to be predicted. In the algorithm of LinGoDec, the update of low-rank W is accomplished by invoking an elegant closed-form solution for least square rank minimization [244], which could be accelerated by BRP.

9.3 Shifted Subspace Tracking (SST) for Motion Segmentation

9.3.1 The Problem of Motion Segmentation

In video sequences, an object flow is composed of multiple motions or moving objects with the identical trajectory. Analyzing and separating the motions of different object flows is more frequently preferred than tracking a single object, because the flows can provide more semantic clues for the crowd behavior [3]. Tracking multiple object flows and motion segmentation [231][93] in complex scenes is a vital and challenging problem in a variety of computer vision tasks such as surveillance, robotics, augmented reality, medical imaging and human-computer interactions. The challenges mainly come from the complex background, occlusion, illumination variation, noise, overlapping, and intertwined trajectories of different flows [90]. Some of these problems have been well studied in the previous research works, but some are typical features of the flow tracking or motion segmentation and thus have not been fully tackled before. In this chapter, we construct a model that considers all the miscellaneous problems above. We reduce the motion detection, tracking and segmentation to a simple matrix factorization on raw pixels rather than sophisticated features of video frames, and then develop an efficient optimization algorithm to achieve the factorization result.

we can roughly categorize existing tracking approaches into two groups, i.e.,

generative model [74][54] and discriminative method [113][118]. Generative model formulates tracking as estimation of the state within a time series space state space model, and search the regions of the highest likelihood. Early works such as Kalman filter and its variants have been demonstrated to be optimal for linear Gaussian model. Another representative generative model for tracking is particle filter [74], which approximates the posterior distribution of the state space by Monte Carlo integration. Single appearance model or multiple appearance models [242] are used in these generative models. Different from generative models, discriminative classifies cast the tracking problem into a classification task whose goal is to distinguish the target object from the background. In the training stage, each pixel is represented by a feature vector, and the pixels belonging to the same target object is assigned to the same class. Hybrid approaches [242] combining both generative models and discriminative classifiers have also been proposed for solving visual tracking under significant occlusions.

Robust PCA (RPCA) [37] has been successfully applied to background modeling in video surveillance, where the backgrounds in all frames compose the rows of low-rank matrix L, whilst the moving objects or motions are captured by the sparse outlier S. However, most existing methods simply treat S as random noise and lack further study of the obtained sparse outlier, which in fact contains substantial rich information about the motions of object flows. However, most existing methods simply treat S as random noise and lack further study of the obtained sparse outlier, which in fact contains substantial rich information about the motions of object flows. In addition, the extra dense noise caused by camera lens or environment illumination changes make the exact decomposition assumption $X = L + S$ cannot be satisfied in practices. Hence several recent approaches [271][121] aim to obtain the approximated RPCA decomposition $X = L + S + G$, where G is the dense noise. In order to overcome the shortcomings of existing RPCA approaches, GoDec in Chapter 6 and GreBsmo in Chapter 7 are proposed as efficient and robust algorithms for low-rank and sparse matrix decomposition.

A significant open problem left by the mentioned RPCA approaches is how to further analyze the rich structure of the sparse part. In many practical applications, the sparse part has structures that can contribute more useful information than the low-rank part. An example in point is that the sparse part of video se-

quence data is comprised of the motions of multiple object flows. In this chapter, we consider decomposing the sparse part as the sum of several shifted low-rank matrices, each of which corresponds to objects sharing one motion trajectory. This can be viewed as a novel structured sparsity. It is also worthwhile to point out that although the main focus of this chapter is object flow tracking and motion segmentation, the SST framework allows other forms of nonlinear transformation and thus can be directly applied to other problems.

We develop an efficient unsupervised framework to detect, track and segment multiple motions in complex scenes via solving a matrix factorization model. This framework invokes a sequence of matrix decompositions as subroutines, which can be summarized in two steps, i.e., background modeling and flow tracking. For the first step, we proposes "semi-soft GoDec" which replaces the cardinality constraint in GoDec with an ℓ_1 penalty. This small change greatly shortens the computational time and facilitates the parameter tuning. For the second step, as the main contribution of this chapter, we present a novel insight that each flow can be depicted by a low rank matrix after certain geometric transformation sequence to video frames (which are stored as rows of a matrix), and develop a matrix factorization method to recover both the low-rank matrices and the transformation sequences. If we treat the sparse outliers attained by semi-soft GoDec as the new data matrix X, our flow tracking approach "shifted subspaces tracking (SST)" decomposes X as $X = \sum_{i=1}^{k} L(i) \circ \tau(i) + S + G$, where $L(i)$ stands for a low-rank matrix and $\tau(i)$ stands for a transformation sequence, both correspond to the motion of the i^{th} object flow. SST reduces the matrix factorization to a sequence of alternating optimizations in a similar manner with semi-soft GoDec, and efficiently solves them by taking advantages of the between-frame affinity and the motion sparsity of $L(i)$. Besides, BRP [263] is invoked to speed up the update of $L(i)$. The low-rank patterns, together with their transformation sequences, reveals unexplored structure of the sparse part and rich information of segmented motion in complex scenes.

9.3.2 SST model

We consider the problem of motion segmentation from the raw video data. Given a data matrix $X \in \mathbb{R}^{n\times p}$ that stores a video sequence of n frames, each of which has $w \times h = p$ pixels and reshaped as a row vector in X, the goal of SST framework is to separate the motions of different object flows, recover both their low-rank patterns and geometric transformation sequences. This task is decomposed as two steps, background modeling that separates all the moving objects from the static background, and flow tracking that recovers the information of each motion. In this problem, $\cdot_i$ stands for the i^{th} entry of a vector or the i^{th} row of a matrix, while $\cdot_{i,j}$ signifies the entry at the i^{th} row and the j^{th} column of a matrix.

The first step can be accomplished by either GoDec (Chapter 6) or GreBsmo (Chapter 7). After obtaining the sparse outliers S storing multiple motions, SST treats the sparse matrix S as the new data matrix X, and decomposes it as $X = \sum_{i=1}^{k} \tilde{L}(i) + S + G$, wherein $\tilde{L}(i)$ denotes the i^{th} motion, S stands for the sparse outliers and G stands for the Gaussian noise.

The motion segmentation in SST is based on an observation to the implicit structures of the sparse matrix $\tilde{L}(i)$. If the trajectory of the object flow $\tilde{L}(i)$ is known and each frame (row) in $\tilde{L}(i)$ is shifted to the position of a reference frame, due to the limited number of poses for the same object flow in different frames, it is reasonable to assume that the rows of the shifted $\tilde{L}(i)$ exist in a subspace. In other words, $\tilde{L}(i)$ after inverse geometric transformation is low-rank. Hence the sparse motion matrix $\tilde{L}(i)$ has the following structured representation

$$\tilde{L}(i) = \begin{bmatrix} L(i)_1 \circ \tau(i)_1 \\ \vdots \\ L(i)_n \circ \tau(i)_n \end{bmatrix} = L(i) \circ \tau(i). \tag{9.1}$$

The invertible transformation $\tau(i)_j : \mathbb{R}^2 \to \mathbb{R}^2$ denotes the 2-D geometric transformation (to the reference frame) associated with the i^{th} motion in the j^{th} frame, which is represented by $L(i)_j$. To be specific, the j^{th} row in $\tilde{L}(i)$ is $L(i)_j$ after certain permutation of its entries. The permutation results from applying the

nonlinear transformation $\tau(i)_j$ to each nonzero pixel in $L(i)_j$ such that,

$$\tau(i)_j(x, y) = (u, v), \tag{9.2}$$

where $\tau(i)_j$ could be one of the five geometric transformations [189], i.e., translation, Euclidean, similarity, affine and homography, which are able to be represented by 2, 3, 4, 6 and 9 free parameters, respectively. For example, affine transformation is defined as

$$\begin{bmatrix} u \\ v \end{bmatrix} = \begin{bmatrix} \rho\cos\theta & \rho\sin\theta \\ -\rho\sin\theta & \rho\cos\theta \end{bmatrix} \begin{bmatrix} x \\ y \end{bmatrix} + \begin{bmatrix} t_x \\ t_y \end{bmatrix}, \tag{9.3}$$

wherein θ is the rotation angle, t_x and t_y are the two translations and ρ is the scaling ratio. It is worth to point out that $\tau(i)_j$ can be any other transformation beyond the geometric group. So SST can be applied to sparse structure in other applications if parametric form of $\tau(i)_j$ is known. We define the nonlinear operator $\circ$ as

$$\begin{aligned} \tilde{L}(i)_{j,u+(v-1)h} &= (L(i)_j \circ \tau(i)_j)_{u+(v-1)h} \\ &= L(i)_{j,x+(y-1)h}. \end{aligned} \tag{9.4}$$

Therefore, the flow tracking in SST aims at decomposing the sparse matrix X (S obtained in the background modeling) as

$$\begin{aligned} &X = \sum_{i=1}^{k} L(i) \circ \tau(i) + S + G, \\ &\text{rank}\,(L(i)) \leq r_i, \text{card}(S) \leq s. \end{aligned} \tag{9.5}$$

In SST, we iteratively invoke k times of the following matrix decomposition to greedily construct the decomposition in (9.5):

$$X = L \circ \tau + S + G, \text{rank}\,(L) \leq r, \text{card}(S) \leq s. \tag{9.6}$$

In each time of the matrix decomposition above, the data matrix X is S obtained by former decomposition. In order to save the computation and facilitate the

parameter tuning, we cast the decomposition (9.6) into an optimization similar to (6.2),

$$\begin{aligned} \min_{L,\tau,S} \quad & \|X - L \circ \tau - S\|_F^2 + \lambda \|S\|_1 \\ s.t. \quad & \operatorname{rank}(L) \leq r, \end{aligned} \tag{9.7}$$

To summarize, the complete procedures of SST based motion segmentation from raw pixels of video frames are given in Algorithm 8.

Algorithm 8: SST Motion Segmentation from Raw Pixels

Input: X, r, λ, $r_i, \lambda_i (i = 1, \cdots, k)$
Output: L, $L(i), \tau(i) (i = 1, \cdots, k)$, S
$(L^\star, S^\star) = \arg \min_{\operatorname{rank}(L) \leq r, S} \|X - L - S\|_F^2 + \lambda \|S\|_1$;
$X := S^\star, L := L^\star$;
for $i \leftarrow 1$ **to** k **do**
  $(L^\star, \tau^\star, S^\star) =$;
  $\arg \min_{\operatorname{rank}(L) \leq r_i, \tau, S} \|X - L \circ \tau - S\|_F^2 + \lambda_i \|S\|_1$;
  $X := S^\star, L(i) := L^\star, \tau(i) = \tau^\star$;
end
$S := X$;

9.3.3 SST Algorithm

SST algorithm for motion segmentation solves a sequence of optimization problem of type (9.7). Thus we firstly apply alternating minimization to (9.7). This results in iterative update of the solutions to the following three subproblems,

$$\begin{cases} \tau^t = \arg\min_{\tau} \|X - L^{t-1} \circ \tau - S^{t-1}\|_F^2; \\ L^t = \arg \min_{\operatorname{rank}(L) \leq r} \|X - L \circ \tau^t - S^{t-1}\|_F^2; \\ S^t = \arg\min_{S} \|X - L^t \circ \tau^t - S\|_F^2 + \lambda \|S\|_1. \end{cases} \tag{9.8}$$

9.3.3.1 Initialization

Unfortunately, alternating solving the 3 sub-problems might not guarantee to produce a global solution or even a stable one, unless an appropriate initialization

is adopted. In particular, in the case when the solutions to $L \circ \tau$ and S in (9.7) are unique, the pair (L, τ) may not be unique. This is because that we can choose arbitrary frame as the reference frame and transform the object flow in all the other frames of $L \circ \tau$ to their positions in the reference frame, while the low-rank $L \circ \tau$ keeps the same. To avoid such trivial multiple solutions of τ, we pre-define a template frame s and do not update its transformation τ_s during the tracking (w.l.o.g., we fix all the parameters of τ_s to be zeros). Then the object flow in each other frame of $L \circ \tau$ are transformed to the position of the object flow in frame s via the inverse transform of τ, and thus the uniqueness of L and τ can be guaranteed. In this chapter, we choose the template frame s as the one with the largest cardinality, which implies that almost all the objects are included in the frame,

$$s = \arg\max_i \operatorname{card}(X_i). \tag{9.9}$$

We then set the rows of the low-rank pattern L as the duplicates of X_s, and initialize both the entries of the sparse outlier S as well as the parameters of τ to be zeros,

$$L = [X_s; \cdots; X_s], S = \mathbf{0}, \tau = \overrightarrow{0}. \tag{9.10}$$

We now start to solve the three subproblems in (9.8).

9.3.3.2 Update of τ

The first subproblem aims at solving the following series of nonlinear equations of τ_j,

$$L_j^{t-1} \circ \tau_j = X_j - S_j^{t-1}, j = 1, \cdots, n. \tag{9.11}$$

Albeit directly solving the above equation is difficult due to its strong nonlinearity, we can approximate the geometric transformation $L_j^{t-1} \circ \tau_j$ by using piece-wise linear transformations, where each piece corresponds to a small change of τ_j defined by $\Delta\tau_j$. Thus the solution of (9.11) can be approximated by accumulating a series of $\Delta\tau_j$. This can be viewed as an inner loop included in the update of τ. Thus we have linear approximation

$$L_j^{t-1} \circ (\tau_j + \Delta\tau_j) \approx L_j^{t-1} \circ \tau_j + \Delta\tau_j J_j, \tag{9.12}$$

where J_j is the Jacobian of $L_j^{t-1} \circ \tau_j$ with respect to the transformation parameters in τ_j. Therefore, by substituting (9.12) into (9.11), $\Delta\tau_j$ in each linear piece can be solved as

$$\Delta\tau_j = \left(X_j - S_j^{t-1} - L_j^{t-1} \circ \tau_j\right) (J_j)^{\dagger}. \tag{9.13}$$

The update of τ_j starts from some initial τ_j, and iteratively solves the overdetermined linear equation (9.13) with update $\tau_j := \tau_j + \Delta\tau_j$ until the difference between the left hand side and the right hand side of (9.11) is sufficiently small. It is critical to emphasize that a well selected initial value of τ_j can significantly save computational time. Based on the between-frame affinity, we initialize τ_j by the transformation of its adjacent frame that is closer to the template frame s,

$$\tau_j := \begin{cases} \tau_{j+1}, & j < s; \\ \tau_{j-1}, & j > s. \end{cases} \tag{9.14}$$

Another important support set constraint, $\text{supp}(L \circ \tau) \subseteq \text{supp}(X)$, needs to be considered in calculating $L_j^{t-1} \circ \tau_j$ during the update of τ. This constraint ensures that the object flows or segmented motions obtained by SST always belong to the sparse part achieved from the background modeling, and thus rules out the noise in background. Hence, suppose the complement set of $\text{supp}(X_j)$ to be $\text{supp}_c(X_j)$, each calculation of $L_j^{t-1} \circ \tau_j$ follows a screening such that,

$$\left(L_j^{t-1} \circ \tau_j\right)_{\text{supp}_c(X_j)} = \overrightarrow{0}. \tag{9.15}$$

9.3.3.3 Update of L

The second subproblem has the following global solution that can be updated by BRP based low-rank approximation (5.1) and its power scheme modification,

$$L^t = \sum_{i=1}^{r} \lambda_i U_i V_i^T, \text{svd}\left(\left(X - S^{t-1}\right) \circ \tau^{-1}\right) = U \Lambda V^T, \tag{9.16}$$

wherein τ^{-1} denotes the inverse transformation towards τ. The SVDs can be accelerated by BRP based low-rank approximation (6.2). Another acceleration trick is based on the fact that most columns of $(X - S^{t-1}) \circ \tau^{-1}$ are nearly all-

zeros. This is because the object flow or motion after transformation occupies a very small area of the whole frame. Therefore, The update of L^t can be reduced to low-rank approximation of a submatrix of $(X - S^{t-1}) \circ \tau^{-1}$ that only includes dense columns. Since the number of dense columns is far less than p, the update of L^t can become much faster.

Algorithm 9: Shifted Subspace Tracking (SST)

Input: X, $r_i, \lambda_i (i = 1, \cdots, n)$, k
Output: $L_i (i = 1, \cdots, n)$, S
for $i \leftarrow 1$ **to** k **do**
  Initialize: $s = \arg\max_i \operatorname{card}(X_i)$;
  $L = [X_s; \cdots; X_s]$, $S = \mathbf{0}$, $\tau = \overrightarrow{0}$;
  while *not converge* **do**
    for $j \leftarrow s - 1$ **to** 1 **do**
      $\tau_j := \tau_{j+1}$;
      while *not converge* **do**
        $\tilde{L}_j^{t-1} = L_j^{t-1} \circ \tau_j, \tilde{L}_{j,\mathrm{supp}_c(X_j)}^{t-1} = \overrightarrow{0}$;
        $\tau_j := \tau_j + \left(X_j - S_j^{t-1} - \tilde{L}_j^{t-1}\right)(J_j)^{\dagger}$;
      end
    end
    for $j \leftarrow s + 1$ **to** n **do**
      $\tau_j := \tau_{j-1}$;
      while *not converge* **do**
        $\tilde{L}_j^{t-1} = L_j^{t-1} \circ \tau_j, \tilde{L}_{j,\mathrm{supp}_c(X_j)}^{t-1} = \overrightarrow{0}$;
        $\tau_j := \tau_j + \left(X_j - S_j^{t-1} - \tilde{L}_j^{t-1}\right)(J_j)^{\dagger}$;
      end
    end
    $\tau^t = \tau$;
    $L^t = \mathrm{BRP}\left((X - S^{t-1}) \circ \tau^{-1}\right)$;
    $S^t = \mathcal{P}_\lambda\left(X - L^t \circ \tau^t\right), S_{j,\mathrm{supp}_c(X_j)}^t = \overrightarrow{0}$;
  end
  $X := S^t, L(i) := L^t, \tau(i) = \tau^t$;
end

9.3.3.4 Update of S

The third subproblem has a global solution that can be obtained via soft-thresholding $\mathcal{P}_\lambda(\cdot)$ similar to the update of S in GreBsmo,

$$S^t = \mathcal{P}_\lambda \left(X - L^t \circ \tau^t \right). \tag{9.17}$$

Figure 9.1: Background modeling and object flow tracking results of a 50-frame surveillance video sequence from Hall dataset with resolution 144×176.

A support set constraint $\text{supp}(S) \subseteq \text{supp}(X)$ should be considered in the

update of S as well. Hence the above update follows a postprocessing,

$$S^t_{j,\text{supp}_c(X_j)} = \overrightarrow{0}, j = 1, \cdots, n. \tag{9.18}$$

Note the transformation computation $\circ$ in the update can be accelerated by leveraging the sparsity of the motions. Specifically, the sparsity allows SST to only compute the transformed positions of the nonzero pixels. We summarize the SST algorithm in Algorithm 9.

9.3.4 Motion Segmentation Experiments of SST

We evaluate SST by using it to track object flows in four surveillance video sequences from the same dataset. In these experiments, the type of geometric transformation τ is simply selected as translation. The detection, tracking and segmentation results as well as associated time costs are shown in Figure **??**.

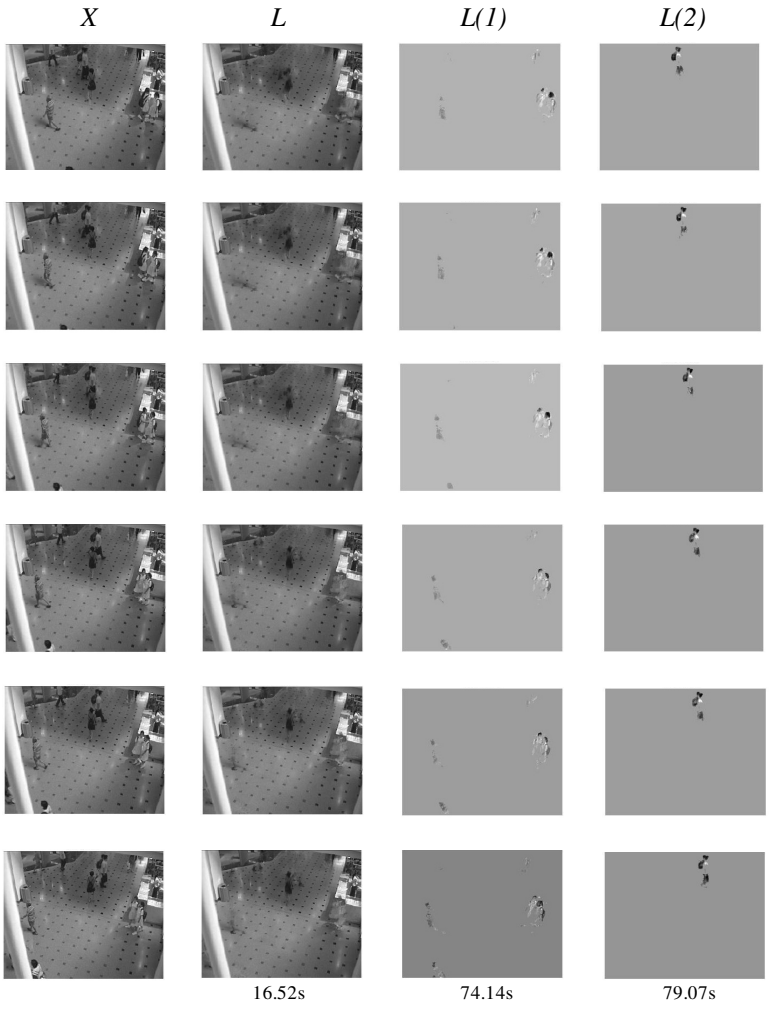

Figure 9.2: Background modeling and object flow tracking results of a 50-frame surveillance video sequence from Shoppingmall dataset with resolution 256×320.

The results show SST can successfully recover both the low-rank patterns and the associated geometric transformations for motions of multiple object flows from the sparse component achieved by GoDec. The detection, tracking and segmentation are seamlessly unified in a matrix factorization framework and achieved with high accuracy. Moreover, it also verifies that SST performs significantly robust on complicated motions in complex scenes. This is attributed to their distinguishing shifted low-rank patterns, because different object flows can hardly share a subspace after the same geometric transformation. Since SST show stable

and appealing performance in motion detection, tracking and segmentation for either crowd or individual, it provides a more semantic and intelligent analysis to the video content than existing methods.

9.4 Multi-label Subspace Ensemble for Multi-label Learning

A challenging problem of multi-label learning is that both the label space and the model complexity will grow rapidly with the increase in the number of labels, and thus makes the available training samples insufficient for training a proper model. In this chapter, we eliminate this problem by learning a mapping of each label in the feature space as a robust subspace, and formulating the prediction as finding the group sparse representation of a given instance on the subspace ensemble. We term this approach as "multi-label subspace ensemble (MSE)". In the training stage, the data matrix is decomposed as the sum of several low-rank matrices and a sparse residual via a randomized optimization, where each low-rank part defines a subspace mapped by a label. In the prediction stage, the group sparse representation on the subspace ensemble is estimated by group *lasso*. Experiments on several benchmark datasets demonstrate the appealing performance of MSE.

9.4.1 The Problem of Multi-label Learning

Multi-label learning [220][187][122] (ML) predicts multiple labels that characterize an instance from a set of possible labels. Conventional multi-label learning algorithms aim to find a mapping from the feature space $\mathcal{X} \subseteq \mathbb{R}^p$ to the label space $\mathcal{Y} \subseteq \{0,1\}^k$, wherein k is the number of labels and $y_i = 1, y \in \mathcal{Y}$ means the sample belongs to label i. Binary relevance (BR) [216] and label powerset (LP) [216] are two natural approaches. BR relaxes ML to k independent binary classifications on the k labels respectively, while LP frames ML as a multi-class classification problem, where each class denotes a unique k-dimensional label vector. Both BR and LP do not duly explore the characteristics of ML, because BR

ignores the label correlations, and LP makes the training samples of each class far less than the prerequisite.

A central problem limiting ML is that the label space $\mathcal{Y}$ will exponentially grow with the increase in the number of labels, i.e., k labels lead to a search in a label space $\mathcal{Y}$ of size 2^k in the prediction. BR independently predicts each dimension of the k-dimensional label vector y in isolation and thus does not encounter this problem with the price of ignoring the label correlations. LP attempts to distinguish each element in $\mathcal{Y}$ from the other $2^k - 1$ ones. Thus the size of the training set, which is large enough for binary classification, will be insufficient for multi-label prediction. This problem also leads to a rapid growth of the model complexity, which increases the training costs. By viewing the problem from the perspective of probabilistic approaches, the exponential growth of $\mathcal{Y}$ drastically enlarges the parameter space for modeling $P(y|x), x \in \mathcal{X}$, which makes ML intractable in computation. Another important problem is that, for a given training set $\{X, Y\}$, the instances in Y often scatters sparsely in the ambient space $\mathcal{Y}$. It is therefore difficult to study the structure of $\mathcal{Y}$ and reduce its dimensionality.

Most recent multi-label learning approaches [252][251] investigate the label correlations (or dependencies) to build a structured classification model. They partially solve the first problem by reducing the size of the search space $\mathcal{Y}$. For example, the random k-labelsets (RAkEL) method [219] randomly selects an ensemble of subsets from the original labelsets (the set of labels one instance belongs to), and then LP is applied to each subset. The final prediction is obtained by ranking and thresholding of the results on the subsets. Hierarchical binary relevance (HBR) [20] builds a general-to-specific tree structure of labels, where a sample with a label must be associated with its parent labels. A binary classifier is trained on each non-root label. Hierarchy of multi-label classifiers (HOMER) [217] recursively partitions the labels into several subsets and builds a tree-shaped hierarchy. A binary classifier is trained on each non-root label subset. The classifier chain (CC) [193] adopts a greedy method to predict unknown label from features and predicted labels by using a binary classifier.

However, the size of the search space in these approaches is much larger than $\mathcal{O}(k)$, and their model complexities are too high to be applied in practice. Also,

the instances in Y are always insufficient to generate a reliable estimation of the label space structure, and thus weaken the effectiveness of the structured classification models used in these approaches.

Some existing learning methods except binary classification are reformulated and then can be extended to multi-label prediction problem. For example, the C&W procedure [30] separates multi-label prediction into two stages, i.e., BR and correction of the BR results by using the label dependence. Regularized multi-task learning [46] and shared-subspace learning [134] formulate the problem as regularized regression or classification problem. Multi-label dimensionality reduction via dependence maximization (MDDM) [253] maximizes the dependence between feature space and label space, and provides a data preprocessing for other multi-label learning methods. A linear dimensionality reduction method for multi-label data is proposed in [133]. In [122], multi-label prediction is formulated as a sparse signal recovery problem.

However, these methods cannot provide an explicit modeling of the label correlations (or dependence) and thus their performance improvements due to exploring label structure are limited. Moreover, they bring extra time costs to the training process, so the efficiency is weakened.

In this chapter, we consider the ML problem in a novel manner: we study the mapping of each label as a feature subspace. In other words, we assume each instance x exists in the ensemble of subspaces defined by the labels that x belongs to. However, it is not always guaranteed that each instance can be completely explained by the labels we consider, so a method should be developed to separate the parts that can be explained by the considered labels and the part that cannot. The label correlations are naturally preserved in the subspace ensemble. Given a new instance, its labels are predicted by estimating its group sparse representation in the subspace ensemble, where the nonzero entries are associated with the predicted labels. There are only k subspaces that are demanded, so the model complexity is small. The prediction is accomplished by searching in the subspace ensemble, and thus avoids the estimation of the label space structure.

We therefore develop "multi-label subspace ensemble (MSE)" to solve the above problem. In the training stage, we develop a randomized decomposition of the training data X, where X is factorized to the sum of k low-rank parts and

a sparse residual. Each of the low-rank part defines the subspace mapped by a particular label, while the sparse residual stores the part that cannot be explained by the considered labels. The decomposition is fast due to an application of the bilateral random projection (BRP) based low-rank approximation [265]. Its convergence to local optimum is proved. In the prediction stage of MSE, group *lasso* estimates the group sparse representation of a given instance in the subspace ensemble, and the nonzero entries indicates the predicted labels. The experiments on several benchmark datasets for ML imply the competitive effectiveness and efficiency of MSE.

The rest of the chapter is organized as follows. Section 9.2.2 introduces the MSE model, which explains the mapping of the label in the feature space and includes the assumption of MSE to the multi-label data. Section 9.2.3 presents the training algorithm of MSE via randomized matrix decomposition, which produces the ensemble of the subspaces. Section 9.2.4 presents the prediction algorithm in MSE by exploring the group sparse representation of a multi-label sample on the subspace ensemble. Section 9.2.5 shows the experimental results of MSE on 13 benchmark datasets. Section 9.2.6 gives a discussion and concludes the chapter.

9.4.2 MSE model

Given a sample $x \in \mathbb{R}^p$ and its label vector $y \in \{0,1\}^k$, we assume that x can be decomposed as the sum of several components l^i and a sparse residual s

$$x = \sum_{i:y_i=1} l^i + s. \tag{9.19}$$

The component l^i is caused by the label i that x belongs to. Thus l^i can be explained as the mapping from the label i in x to the feature space. The residual s is the component that all the labels in y cannot explain. The model in (9.19) reveals the general relationship between the feature space and the label space.

For all the samples with label i, we assume their components explained by label i lie in a linear subspace $C^i \in \mathbb{R}^{r^i \times p}$, i.e., $l^i = \beta_{G_i} C^i$, wherein β_{G_i} is the representation coefficients corresponding to C^i. Thus the model (9.19) can be

equivalently written as

$$\begin{aligned} x &= \sum_{i=1}^{k} \beta_{G_i} C^i + s, \\ &\forall i \in \{i : y_i = 0\}, \beta_{G_i} = \mathbf{0}. \end{aligned} \tag{9.20}$$

If we build the subspace ensemble $C = [C^1; \ldots; C^k]$ characterized by the k labels as a dictionary for x, the corresponding representation coefficient vector for x is $\beta = [\beta_{G_1}, \ldots, \beta_{G_k}]$. The coefficients β_{G_i} corresponding to the labels that x does not belong to are zeros, so β is group sparse, wherein the groups are $G_i, i = 1, \ldots, k$.

In the training stage of MSE, we learn the subspace ensemble $C^i, i = 1, \ldots, k$ from the training data X via a randomized decomposition of X, in which the components explained by label i from all the samples consists a low-rank matrix $L^i_{\Omega_i}$, wherein Ω_i is the index set of training samples with label i. Thus the row space of $L^i_{\Omega_i}$ is the subspace C^i. In the prediction stage of MSE, given a new instance x, we use group *lasso* to find the group sparse representation β on the subspace ensemble C, and then a simple thresholding is used to test which groups that β concentrates on. The labels that these groups correspond to are the predicted labels for the instance x.

In the training stage of MSE, the label correlations is naturally preserved in the subspace ensemble C, because all the subspaces are jointly learned. Specifically, if two labels i and j simultaneously appear for many times in the training samples, then Ω_i and Ω_j will have many shared elements. Thus the row spaces of $L^i_{\Omega_i}$ and $L^i_{\Omega_j}$, i.e., C^i and C^j, will be close to each other. In the prediction stage, both discriminative and structured information encoded in the subspace ensemble are considered via group *lasso*. Since only k subspaces are learned in the training stage, MSE explores label correlations without increasing the model complexity.

9.4.3 MSE Algorithm

9.4.3.1 MSE training: randomized decomposition

The training stage of MSE approximately decomposes the training data matrix $X \in \mathbb{R}^{n\times p}$ into $X = \sum_{i=1}^{k} L^i + S$. For the matrix L^i, the rows corresponding to the samples with label i are nonzero, while the other rows are all-zero vectors. The nonzero rows denote the components explained by label i in the feature space. We use Ω_i to denote the index set of samples with label i in the matrix X and L^i, and then the matrix composed of the nonzero rows in L^i is represented by $L^i_{\Omega_i}$. In the decomposition, the rank of $L^i_{\Omega_i}$ is upper bounded, which indicates that all the components explained by label i nearly lies in a linear subspace. The matrix S is the residual of the samples that cannot be explained by the given labels. In the decomposition, the cardinality of S is upper bounded, which makes S sparse.

If the label matrix of X is $Y \in \{0,1\}^{n\times k}$, the rank of $L^i_{\Omega_i}$ is upper bounded by r^i and the cardinality of S is upper bounded by K, the decomposition can be written as solving the following constrained minimization problem:

$$\begin{array}{ll} \min\limits_{L^i,S} & \left\| X - \sum_{i=1}^{k} L^i - S \right\|_F^2 \\ s.t. & \text{rank}\left(L^i_{\Omega_i}\right) \leq r^i, L^i_{\overline{\Omega}_i} = \mathbf{0}, \forall i = 1,\ldots,k \\ & \text{card}\left(S\right) \leq K. \end{array} \tag{9.21}$$

Therefore, each training sample in X is decomposed as the sum of several components, which respectively correspond to multiple labels that the sample belongs to. MSE separates these components from the original sample by building the mapping from the labels to the feature space. For label i, we obtain its mapping in the feature space as the row space of $L^i_{\Omega_i}$.

Although the rank constraint to $L^i_{\Omega_i}$ and cardinality constraint to S are not convex, the optimization in (9.21) can be solved by alternating minimization that decomposes it as the following $k+1$ subproblems, each of which has the global

solution:

$$\begin{cases} L^i_{\Omega_i} = \arg \min\limits_{\text{rank}\left(L^i_{\Omega_i}\right)\leq r^i} \left\| X - \sum\limits_{j=1, j\neq i}^{k} L^j - S - L^i \right\|_F^2, \\ \qquad \forall i = 1, \ldots, k. \\ S = \arg \min\limits_{\text{card}(S)\leq K} \left\| X - \sum\limits_{j=1}^{k} L^j - S \right\|_F^2. \end{cases} \tag{9.22}$$

The solutions of $L^i_{\Omega_i}$ and S in the above subproblems can be obtained via hard thresholding of singular values and the matrix entries, respectively. Note that both SVD and matrix entry-wise hard thresholding have global solutions. In particular, $L^i_{\Omega_i}$ is built from the first r^i largest singular values and the corresponding singular vectors of $\left(X - \sum_{j=1, j\neq i}^{k} L^j - S\right)_{\Omega_i}$, while S is built from the K entries with the largest absolute value in $X - \sum_{j=1}^{k} L^j$, i.e.,

$$\begin{cases} L^i_{\Omega_i} = \sum\limits_{q=1}^{r^i} \lambda_q U_q V_q^T, i = 1, \ldots, k, \\ \text{svd}\left[\left(X - \sum_{j=1, j\neq i}^{k} L^j - S\right)_{\Omega_i}\right] = U \Lambda V^T; \\ S = \mathcal{P}_\Phi \left(X - \sum\limits_{j=1}^{k} L^j\right), \Phi : \left| \left(X - \sum\limits_{j=1}^{k} L^j\right)_{r,s\in\Phi} \right| \neq 0 \\ \text{and } \geq \left| \left(X - \sum\limits_{j=1}^{k} L^j\right)_{r,s\in\overline{\Phi}} \right|, |\Phi| \leq K. \end{cases} \tag{9.23}$$

The projection $S = \mathcal{P}_\Phi(R)$ represents that the matrix S has the same entries as R on the index set Φ, while the other entries are all zeros.

The decomposition is then obtained by iteratively solving these $k+1$ subproblems in (9.22) according to (9.23). In this problem, we initialize $L^i_{\Omega_i}$ and S as

$$\begin{cases} L^i_{\Omega_i} := Z_{\Omega_i}, i = 1, \ldots, k, \\ Z = D^{-1} X, D = \text{diag}\left(Y \mathbf{1}\right); \\ S := \mathbf{0}. \end{cases} \tag{9.24}$$

In each subproblem, only one variable is optimized with the other variables fixed.

Similar to GoDec in Chapter 6, BRP based acceleration strategy in Chapter 5 is applied to the above model and produces the practical training algorithm in Algorithm 10.

In the training, the label correlations is naturally preserved in the subspace ensemble, because all the subspaces are jointly learned. Since only k subspaces are learned in the training stage, MSE explores label correlations without increasing the model complexity.

9.4.3.2 MSE prediction: group sparsity

Algorithm 10: MSE Training

Input: X, Ω_i, $r^i, i = 1, \ldots, k$, K, ϵ

Output: $C^i, i = 1, \ldots, k$

Initialize L^i and S according to (9.24), $t := 0$;

while $\left\| X - \sum_{j=1}^{k} L^j - S \right\|_F^2 > \epsilon$ **do**

  $t := t + 1$;

  for $i \leftarrow 1$ **to** k **do**

    $N := \left(X - \sum_{j=1, j \neq i}^{k} L^j - S \right)_{\Omega_i}$;

    Generate standard Gaussian matrix $A_1 \in \mathbb{R}^{p \times r^i}$;

    $Y_1 := NA_1$, $A_2 := Y_1$;

    $Y_2 := N^T Y_1$, $Y_1 := NY_2$;

    $L^i_{\Omega_i} := Y_1 \left(A_2^T Y_1 \right)^{-1} Y_2^T$, $L^i_{\overline{\Omega}_i} := \mathbf{0}$;

  end

  $N := \left| X - \sum_{j=1}^{k} L^j \right|$;

  $S := \mathcal{P}_{\Phi}(N)$, Φ is the index set of the first K largest entries of $|N|$;

end

QR decomposition $\left(L^i_{\Omega_i} \right)^T = Q^i R^i$ for $i = 1, \ldots, k$, $C^i := (Q^i)^T$;

In the prediction stage of MSE, we use group *lasso* [243][161] to estimate the group sparse representation $\beta \in \mathbb{R}^{\sum r^i}$ of a test sample $x \in \mathbb{R}^p$ on the subspace ensemble $C = [C^1; \ldots; C^k]$, wherein the k groups are defined as index sets of the coefficients corresponding to $C^1, \ldots, C^k$. Since group *lasso* selects nonzero coefficients group-wisely, nonzero coefficients in the group sparse representation

will concentrate on the groups corresponding to the labels that the sample belongs to.

According to the above analysis, we solve the following group *lasso* problem in the prediction stage of MSE

$$\min_{\beta} \frac{1}{2} \|x - \beta C\|_F^2 + \lambda \sum_{i=1}^{k} \|\beta_{G_i}\|_2 , \tag{9.25}$$

where the index set G_i includes all the integers between $1 + \sum_{j=1}^{i-1} r^j$ and $\sum_{j=1}^{i} r^j$ (including these two).

To obtain the final prediction of the label vector $y \in \{0,1\}^k$ for a test sample x, we use a simple thresholding of the magnitude sum of coefficients in each group to test which groups that the sparse coefficients in β concentrate on

$$y_\Psi = \mathbf{1}, y_{\overline{\Psi}} = \mathbf{0}, \Psi = \{i : \|\beta_{G_i}\|_1 \geq \delta\} . \tag{9.26}$$

Although y can also be obtained via selecting the groups with nonzero coefficients when λ in (9.25) is chosen properly, we set the threshold δ as a small positive value to guarantee the robustness to λ.

Algorithm 11 below summarizes the prediction stage of MSE.

Algorithm 11: MSE Prediction

Input: x, $C^i, i = 1, \ldots, k$, λ, δ
Output: y
Solve group *lasso* in (9.25) by using an existing group *lasso* algorithm [161];
Predict y via thresholding in (9.26);

9.4.4 Multi-label Prediction Experiments of MSE

We evaluate MSE on 13 benchmark datasets from different domains and of different scales, including Corel5k (image), Scene (image), Mediamill (video), Enron (text), Genbase (genomics), Medical (text), Emotions (music), Slashdot (text) and 5 sub datasets selected in Yahoo dataset (web data). These datasets were

obtained from Mulan's website [1] and MEKA's website [2]. They were collected from different practical problems. Table 9.1 shows the number of samples n (training samples+test samples), number of features p, number of labels k, and the average cardinality of all label vectors $Card$ of different datasets.

Table 9.1: Information of datasets that are used in experiments of MSE. In the table, n (training samples+test samples) is the number of samples, p is the number of features, k is the number of labels, "Card" is the average cardinality of all label vectors.

Datasets	n	p	k	Card
Corel5k	$4500 + 500$	499	374	3.522
Mediamill	$30993 + 12914$	120	101	4.376
Enron	$1123 + 579$	1001	53	3.378
Genbase	$463 + 199$	1186	27	1.252
Medical	$333 + 645$	1449	45	1.245
Emotions	$391 + 202$	72	6	1.869
Scene	$1211 + 1196$	294	6	1.074
Slashdot	$2338 + 1444$	1079	22	1.181
Arts	$2000 + 3000$	462	26	1.636
Business	$2000 + 3000$	438	30	1.587
Education	$2000 + 3000$	550	33	1.461
Recreation	$2000 + 3000$	606	22	1.423
Science	$2000 + 3000$	743	40	1.451

We compare MSE with BR [216], ML-KNN [246] and MDDM [253] on four evaluation metrics for evaluating the effectiveness, as well as the CPU seconds for evaluating the efficiency. In multi-label prediction, four metrics, which are precision, recall, F1 score and accuracy, are used to measure the prediction performance. The detailed definitions of these metrics are given in Section 7.1.1 of [218]. A fair evaluation of prediction performance should include integrative

[1] `http://mulan.sourceforge.net/datasets.html`
[2] `http://meka.sourceforge.net/`

consideration of all the four metrics, whose importances can be roughly given by $F1, Acc > \{Prec, Rec\}$.

Table 9.2: Prediction performances (%) and CPU seconds of BR [216], ML-KNN [246], MDDM [253] and MSE on Yahoo. Prec-precision, Rec-recall, F1-F1 score, Acc-accuracy

	Methods	Prec	Rec	F1	Acc	CPU sec.
Arts	BR	76	25	26	24	46.8
	ML-knn	62	7	25	6	77.6
	MDDM	68	6	21	5	37.4
	MSE	35	40	31	28	11.7
Education	BR	69	27	28	26	50.1
	ML-knn	58	6	31	5	99.8
	MDDM	59	5	26	5	45.2
	MSE	41	35	32	29	12.6
Recreation	BR	84	23	23	22	53.2
	ML-knn	70	9	23	8	112
	MDDM	66	7	18	6	41.9
	MSE	41	49	36	30	19.1
Science	BR	79	19	19	19	84.9
	ML-knn	59	4	20	4	139
	MDDM	66	4	19	4	53.0
	MSE	31	39	29	26	20.1
Business	BR	87	74	76	71	28.9
	ML-knn	68	9	70	8	93.2
	MDDM	66	7	69	7	42.7
	MSE	84	82	78	78	13.5

We show the prediction performance and time cost in CPU seconds of BR, ML-KNN, MDDM and MSE in Table 9.3 and Table 9.2. In BR, we use the MatLab interface of LIBSVM 3.0 [1] to train the classic linear SVM classifiers for each label. The parameter $C \in \{10^{-3}, 10^{-2}, 0.1, 1, 10, 10^2, 10^3\}$ with the best performance on the training set was used. In ML-KNN, the number of neighbors was 30 for all the datasets.

[1] `http://www.csie.ntu.edu.tw/čjlin/libsvm/`

Table 9.3: Prediction performances (%) and CPU seconds of BR [216], ML-KNN [246], MDDM [253] and MSE on 8 datasets. Prec-precision, Rec-recall, F1-F1 score, Acc-accuracy

	Methods	Prec	Rec	F1	Acc	CPU sec.
Mediamill	BR	69	35	43	33	120141
	ML-knn	41	6	54	5	5713
	MDDM	36	5	53	4	48237
	MSE	58	78	53	37	1155
Enron	BR	51	28	35	24	77.1
	ML-knn	51	7	46	5	527
	MDDM	50	8	49	7	29
	MSE	44	50	40	28	271
Medical	BR	2	26	5	2	4.88
	ML-knn	75	7	48	6	22.8
	MDDM	74	3	30	2	32.3
	MSE	36	90	45	26	7.5
Slashdot	BR	11	22	14	10	140
	ML-knn	71	10	31	8	708
	MDDM	39	1	4	1	114
	MSE	38	61	37	27	175
Scene	BR	55	67	66	63	4.19
	ML-knn	78	62	69	54	14.3
	MDDM	75	64	69	53	7.59
	MSE	61	85	70	68	3.62
Emotions	BR	55	53	51	42	0.68
	ML-knn	68	28	41	22	0.66
	MDDM	54	28	41	22	0.66
	MSE	40	100	52	37	0.01
Genbase	BR	5	39	9	5	1.99
	ML-knn	100	50	92	50	9.38
	MDDM	98	51	92	51	6.09
	MSE	83	96	86	70	8.62
Corel5k	BR	2	20	4	2	2240
	ML-knn	62	1	3	0.9	2106
	MDDM	62	1	7	1	458
	MSE	9	11	8	5	1054

In MDDM, the regularization parameter for uncorrelated subspace dimensionality reduction was selected as 0.12 and the dimension of the subspace was set as 20% of the dimension of the original data.

In MSE, we selected r^i as an integer in $[1, 6]$, $K \in [10^{-6}, 10^{-3}]$, $\lambda \in [0.2, 0.45]$ and $\delta \in [10^{-4}, 10^{-2}]$. We roughly selected 4 groups of parameters in the ranges for each dataset and chose the one with the best performance on the training data. Group *lasso* in MSE is solved by SLEP [161] in our experiments.

The experimental results show that MSE is competitive on both speed and prediction performance, because it explores label correlations and structure without increasing the problem size. In addition, the bilateral random projections further accelerate the computation. In particular, its training time increases much more slowly than other methods, so it is more efficient when applied to large scale datasets such as Mediamill, Arts and Education. MDDM is faster than MSE on a few datasets because MDDM invokes ML-knn on the data after dimension reduction, while MSE is directly applicable to the original high dimensional data.

In the comparison of performance via the four metrics, the F1 score and accuracy of MSE outperform those of other methods on most datasets. Moreover, MSE has smaller gaps between precision and recall on different tasks than other methods, and this implies it is robust to the imbalance between positive and negative samples. Note in multi-label prediction, only large values of all four metrics are sufficient to indicate the success of the prediction, while the combination of some large valued metrics and some small valued ones are always caused by the imbalance of the samples. Therefore, MSE provides better prediction performance than other methods on most datasets.

9.5 Linear Functional GoDec for Learning Recommendation System

Although low-rank matrix completion provides an effective and simple mathematical model predicting a user's rating to an item from her/his ratings to other items and the ratings of other users by exploring the user relationships. However, a primary problem of this model is that adding a new item or a new user to the

model requires an new optimization of the whole low-rank rating matrix, which is not practical for its expensive time cost. Moreover, although the attributes of users are always missing in real recommendation systems, features of the items have been proved to be helpful side information that is much easier to obtain. But previous matrix completion methods and GoDec cannot leverage this information in their models. Furthermore, robust rating prediction should allow advertising effects in known ratings.

9.5.1 LinGoDec Model and Algorithm

In this part, we propose a variant of GoDec called "linear functional GoDec (LinGoDec)" that learns a scoring function of item features for each user by replacing L with WZ^T WZ^T, where W represents the linear functions and the rows of Z are items represented by features. LinGoDec aims at solving the following optimization,

$$\begin{aligned} &\min_{W,S} \|X - WZ^T - S\|_F^2 + \lambda \|\text{vec}(S)\|_1 \\ &\text{s.t.} \quad rank(W) \leq r. \end{aligned} \tag{9.27}$$

We constrain W to be low-rank so that the functions of different users share the same small set of basis functions. In addition, we apply ℓ_1 regularization to the entries of S so that the advertising effects in training ratings can be captured and ruled out from the learning of W. By applying alternating minimization to (9.27), we have

$$\begin{cases} W_k = \arg\min_W \left\|X - WZ^T\right\|_F^2 \quad s.t. \ \ \text{rank}(W) \leq r, \\ S_k = \mathcal{S}_\lambda \left(X - W_k Z^T\right), \end{cases} \tag{9.28}$$

The update of W_k in above procedures equals to solve a least squares rank minimization, which has been discovered owning closed-form solution that can be obtained by truncated SVD when X is singular (the most common case in our problem). By applying bilateral random projection based acceleration to the truncated SVD, we immediately achieve the final fast algorithm for LinGoDec. LinGoDec has a similar model as rank-regularized multi-task learning, but the

major difference is that the sparse matrix in LinGoDec is a component of the data matrix rather than the linear functions W.

9.5.2 Empirical Study of LinGoDec

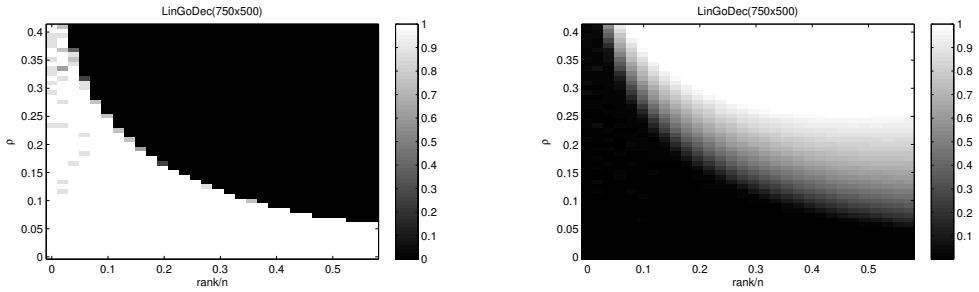

Figure 9.3: Phase diagram (left) and corresponding CPU seconds (right) for LinGoDec on 750×750 matrices. Low-rank weight matrix W is of size 750×500, and is generated by $W = UV$, where entries of U and V are sampled from $\mathcal{N}(0, 1/750)$ and $\mathcal{N}(0, 1/750)$, respectively. Features of items in Z is sampled from $\mathcal{N}(0, 1/750)$. Entries of sparse anomaly S are sampled as 1 or -1 with probability $\rho/2$ and 0 with probability $1 - \rho$. Noise G has entries sampled from $\mathcal{N}(0, 10^{-3})$. On the 50×30 grid of sparsity-rank/n plane, 10 trials are performed for each (ρ, r) pair. W is said to be successfully recovered if its rel. err.$\leq 10^{-2}$. The phase diagram shows the successful recovery rate for each (ρ, r) pair.

Since most public available dataset for recommendation system rarely fulfill our demands for the training data in LinGoDec, we justify LinGoDec on synthetic data. Specifically, the rating matrix X is generated by $WZ^T + S + G$. The weight matrix of linear functions W is generated as the product of two Gaussian matrices. Entries in both the item feature matrix Z and noise matrix G are generated by i.i.d. Gaussian distribution. The sparse part has a Bernoulli model generated support set on which ± 1 values are randomly assigned.

We show the phase diagram and the corresponding time cost in Figure 9.5.2. It could be seem that LinGoDec has a slightly larger region (the white region) for successful recovery than both GreBsmo and robust PCA [37]. This is because side-information, i.e., the features of items, is utilized in LinGoDec. Moreover,

the time cost of LinGoDec is still small due to the closed-form update of W and BRP based acceleration. Therefore, LinGoDec is capable to achieve the scoring functions of users, which cannot be learned by previous matrix completion based methods, and is effective to rule out the advertising effects in user ratings. Its fast speed makes it very efficient when applied to practical systems.

Chapter 10

Compressed Labeling on Distilled Labelsets

Directly applying single-label classification methods to the multi-label learning problems substantially limits both the performance and speed due to the imbalance, dependence and high dimensionality of the given label matrix. Existing methods either ignore these three problems or reduce one with the price of aggravating another. In this chapter, we propose a $\{0, 1\}$ label matrix compression and recovery method termed "compressed labeling (CL)" to simultaneously solve or at least reduce these three problems. CL first compresses the original label matrix to improve balance and independence by preserving the signs of its Gaussian random projections. Afterward, we directly train popular binary classifiers (e.g., support vector machines) for each new label. A fast recovery algorithm is developed to recover the original labels from the predicted new labels. In the recovery algorithm, "labelset distilling method" is designed to extract distilled labelsets (DLs), i.e., the frequently appeared label subsets from the original labels via recursive clustering and subtraction. Given a DL and an original label vector, we discover that the signs of their random projections have an explicit joint distribution that can be quickly computed from a geometric inference. Based on this observation, the original label vector is exactly determined after performing a series of Kullback-Leibler divergence based hypothesis tests on the distribution about the new labels. CL significantly improves the balance of the training sam-

ples and reduces the dependence between different labels. Moreover, it accelerates the learning process by training less binary classifies for compressed labels, and makes use of label dependence via DLs based tests. Theoretically, we prove the recovery bound of CL which verifies the effectiveness of CL for label compression and multi-label classification performance improvement brought by label correlations preserved in DLs. We show the effectiveness, efficiency and robustness of CL via 5 groups of experiments on 21 datasets from text classification, image annotation, scene classification, music categorization, genomics and web page classification.

10.1 Introduction

The past years have witnessed the significant contributions of multi-label learning for various practical applications, such as text classification, image annotation, scene classification, music categorization, genomics and web page classification, where each sample simultaneously belongs to several classes out of a great amount of possible candidates. Recently, learning from data with multiple labels attracts growing attention from related fields and is developed rapidly. Its importance and necessity have been well appreciated by plenty of specific utilizations.

In contrast to single-label binary classification, multi-label learning predicts a $\{0,1\}$ label matrix $Y \in \{0,1\}^{n\times k}$ (n is the number of samples and k is the number of labels) rather than a $\{0,1\}$ label vector $y \in \{0,1\}^n$. At an early stage, binary relevance (BR) [216] and label powerset (LP) [216] were developed to transform a multi-label learning problem to several binary classification tasks. Specifically, BR associates each label with an individual class, i.e., assigns samples with the same label to the same class; and LP treats each unique set of labels as a class, in which samples share the same label vector.

Although BP/LP and their variants can directly transform a multi-label learning problem into multiple binary classification tasks, it has been widely acknowledged that these transformations share the following three problems: sample imbalance, label dependence and label high-dimensionality. Because these transformations do not consider the differences between the information embedded in several independent single labels and multiple labels. These problems may ruin

the binary classifier training and even end up with trivial solutions to the label prediction, e.g., assigning the same label to all samples from different classes.

10.1.1 Three problems

The problem of sample imbalance. It usually occurs in multi-label learning and multi-class learning, when more than two classes are considered and the one-versus-all rule is adopted. In such a case, the conventional binary classification methods tend to overwhelm the class with more samples. This sample imbalance between these two classes will seriously weaken the classification performance and even make the learning task fail by assigning the same label to all samples from different classes.

Specifically, BR directly uses the original label matrix as the class indicator matrix and trains a classifier for each label. It allows the overlapping of classes and treats them as independent ones. Thus BR has the imbalance problem when 0 and 1 in columns of the label matrix are imbalanced. LP treats each unique labelset as the sign of an independent class and transfers a multi-label prediction problem to a larger size multi-class classification problem. In contrast to BR, LP tremendously increases the number of classes and decreases samples in each class, so it aggravates the sample imbalance problem.

The problem of label dependence. This is a characteristic problem of multi-label learning and tells the difference between multi-label learning and multi-class learning, because it is admitted a simultaneous appearance of different labels on one sample. The dependence or correlation between different labels can then be studied by using statistics of their distributions, e.g., χ^2 test and Pearson's correlation coefficient. This dependence between labels ends up with the poor performance of multi-label learning when it is directly decomposed into several binary classification tasks. That is because the direct decomposition assumes different labels are independent of each other and ignores the label dependence. In contrast to binary classification, samples sharing one identical label in multi-label learning may quite differ in concepts and have large pairwise distances in the feature space, because their other labels can be different. There are two types of label dependence [64], the conditional dependence and uncon-

ditional dependence. The former captures the label dependence given a specific sample, while the latter considers the global label dependence in the label space. Exploiting label dependence becomes a popular motivation in recently developed multi-label learning methods [193][20][217]. Both empirical and theoretical studies have proved that it helps improving the learning performance by considering the label dependence.

BR simply ignores the dependence between labels and independently trains classifiers for given labels. Thus it performs unsatisfying when the labels are highly correlated to each other. LP treats the unique different labelsets as independent classes and trains one binary classifier for each of them. It takes label dependence into consideration, but it neglects the correlation or shared labels between different labelsets and deteriorates the data imbalance as the cost.

The problem of label high dimensionality. In practical problems, e.g., text classification [100][162] and image annotation, data usually have hundreds or even thousands of labels, which leads to a $\{0, 1\}$ label matrix with sparse entries and high dimensionality. The high dimensionality of the label matrix makes the multi-label learning task very challenging. In particular, the increasing of label dimensionality enlarges the sample imbalance in each binary classification and increases the number of classifiers to be trained.

BR directly adopts the original label matrix, while LP increases the number of labels to K much larger than that of the original labels. Most existing multi-label learning methods transfer the original problem to m binary classifications, wherein m is a number between k and K.

10.1.2 Previous works

Both empirical and theoretical studies of the existing multi-label algorithms suggest that the learning performance is determined by specific properties of a multi-label dataset, e.g., label dependence, label structure and dependence between samples and the corresponding labels. Nevertheless, most existing methods either partially tackle some of the three problems and ignore the others, or eliminate one with the price of exacerbating the other two. In this chapter, we categorize these methods into five groups. It is impossible to exhaustively summarize all the

published methods. Thus, domain experts will easily note the missing references. We hope that the cited reviews [218][216] cited here will point to the missing references.

Label Transformation. Methods belonging to this group transform the given labels into new ones, and then decompose the original multi-label prediction problem into a series of binary classification tasks according to the new labels. This group of methods can embed the label dependence information into the transformed labels, and can exploit the label structure to decrease the number of new labels.

Some methods, e.g., BR and LP, in this group treat new labels independently, so the label transformation and the classifier training are independent. After obtaining the new labels, one binary classifier is trained for each new label independently without considering its relationship to others. The pruned problem transformation (PPT) [192] modifies LP via replacing the rare labelsets with their more frequent subsets, and thus both the sample imbalance problem and label high dimensionality problem are alleviated. The random k-labelsets (RAkEL) method [219] randomly selects an ensemble of subset from the original labelsets, and then LP is applied to each subset. The sequential prediction is accomplished by ranking and thresholding on the results of the ensemble of LP classifiers. It is a modification of LP with the motivation of utilizing the label dependence. Ranking by pairwise comparison (RPC) [125] adopts one-versus-one rule by training a binary classifier for each pair of labels and ranking the classification results for prediction. RPC alleviates the sample imbalance problem in the training of each classifier, but increases the number of labels to $k(k-1)/2$ (k is the number of original labels). Related methods includes multi-label pairwise perceptron (MLPP) [175] and calibrated label ranking (CLR) [91].

Other methods in this group establish a structure of labels. The classifier training and label prediction are then implemented on the obtained structure. These methods take the label dependence into consideration and reduces the sample imbalance problem. Two representatives are hierarchical binary relevance (HBR) [20] and hierarchy of multi-label classifiers (HOMER) [217]. HBR builds a general-to-specific tree structure of labels, where a sample with a label must be associated with its parent labels. For each non-root label on the hierarchical

structure, a binary classifier is designed by using a subset of samples whose labels include the parent labels of the current one. HOMER recursively partitions the labels into several subsets and builds a tree hierarchy. In the hierarchy, each node is composed of several labels that are separated into a number of subsets in its child nodes. HBR method is then applied to each node for obtaining multi-label classifiers to separate the child nodes.

Regularized classifications. This group of methods formulates the problem as a series of classifications with regularization. Stacking method [49] and "Curds and Whey (C&W)" procedure [30] separate the classification and regularization as two isolated stages. They train a classifier on each label as BR, and then correct the prediction of each label by using the predictions of the others. These two methods impose a regularization to the conventional classification results, wherein the predictions of the other labels perform as a bias to decrease the variance of current label prediction. The regularization item always aims to exploit label dependence. Another kind of regularization directly solve regularized classification problems and jointly learn all the binary classifiers that share a parameter space. Two examples are regularized multi-task learning [81] and shared-subspace learning [134]. If a linear classifier w_i is trained on each label, the former method assumes $w_i = v_i + w_0$, while the latter method assumes $w_i = v_i + u_i\Theta$. Both the w_0 and Θ are shared parameters that store the label dependence information.

Multi-label Learning Problem reformulation. This group of methods formulates the multi-label learning problem as other supervised learning problems [140] rather than classification and ranking, which are two methods usually extended from single-label learning. Multi-label dimensionality reduction via dependence maximization (MDDM) [253] tackles the "curse of dimensionality" in multi-label data and formulates the problem as a discriminative dimension reduction [269][19]. It maximizes the dependence between feature space and label space via maximizing the empirical estimate of Hilbert-Schmidt Independence Criterion (HSIC) [101]. Graphical models such as conditional random fields (CRF) [95] are natural solutions to estimate the joint distribution of samples and labels in multi-label learning. They provide a probabilistic formulation of the problem. The classifier chain (CC) [193] adopts a greedy way to concatenate the binary

classifiers for all the labels and makes use of conditional label dependence. It trains a classifier for each label at a time by using given samples and the previously predicted labels as the input. Thus the prediction of each label is related to the previously predicted ones. It has an ensemble variant (ECC) [193] and a probabilistic variant (PCC) [63]. The former alleviates the influence of label order, while the latter tackles the Bayes-optimal solution of CC.

Linear Regression. This group of methods adopts linear regression model to solve multi-label learning problems. Although linear model for classification is criticized due to its underlying assumption [109], a number of popular techniques can be used to solve the aforementioned three problems in this scenario. In particular, the linear model is $Y = XW$, where Y is the label matrix and the columns of W are the corresponding model coefficient vectors. In [132], the optimization of W is formulated as a matrix completion problem when W is assumed to be low-rank. The given samples and the corresponding label vectors comprise the random measurement matrix ensemble $X_i^T Y_i (i = 1, ..., n)$. The low rank assumption of W embeds the label dependence in the learning process. A multi-task method proposed in [46] assumes that W is the sum of a low-rank component and a sparse component. Different from imposing an extra assumption on W in the linear model, another observation is that the $\{0, 1\}$ label matrix Y is sparse and thus compressible. In [122], Y is compressed via random projections $Y' = YA$, and then a new regression model $Y' = XW'$ is obtained, the original Y can be recovered via compressed sensing algorithms. This is a successful application of compressed sensing (CS) [71] to multi-label learning and inherits the theoretical merit of CS, i.e., only $\mathcal{O}(\log(k))$ models needs to be trained for data with k labels. This method reduces the model complexity caused by the large number of labels.

Algorithm extension. This group of methods extends or modifies the existing supervised learning algorithms to the multi-label learning scenario. C4.5 is a popular decision tree algorithm and is extended to multi-label learning in [51] via modifying the entropy criterion. AdaBoost has been extended to multi-label data ranking and Hamming loss minimization in AdaBoost.MR and AdaBoost.MH [198], respectively. The multi-class multi-label perceptron (MMP) [56], back-propagation for multi-label Learning (BP-MLL) [247], and RBF neural networks

for multi-instance multi-label learning (MIMLRBF) [248] are extensions of neural networks algorithms in multi-label learning. Multi-label k-nearest neighbor (ML-knn) [246] is an extension of knn. It obtains the label prior distribution from the k nearest neighbors and applies "maximizes a posteriori (MAP)" to the label prediction. It partially solves the imbalance problem.

Multi-label learning methods can also be roughly distinguished into learning reduction methods and fully-specified learning methods. In particular, the "label transformation" and "algorithm extension" in our results can be attributed to "learning reduction methods" because they transform the multi-label learning problem into other different subproblems. "Regularized classification", "multi-label learning problem reformulation", "linear regression" and some methods of "algorithm extension" in our results can be attributed to "specified learning methods" because they formulate multi-label learning as specific problems. From some perspective, compressed labeling (CL) proposed in this chapter can be viewed as a learning reduction method.

10.1.3 The proposed method

In this chapter, we propose a $\{0, 1\}$ label matrix compression and recovery scheme termed "compressed labeling (CL)" for multi-label learning. It simultaneously solves or at least substantially alleviates the aforementioned three problems via random coding of the label matrix and fast corresponding decoding (recovery). CL is a general scheme for embedding existing single-label learning methods in a multi-label learning setting. The label compression in CL leads to a shrinkage of the problem size, and thus it is efficient in large-scale problems [174].

We summarize the CL scheme including its training stage and prediction stage in Figure 10.1.

In the training stage, CL first compresses the given $\{0, 1\}$ label matrix Y into a sign matrix Z of its random projections on Gaussian random matrix A. Due to the properties of random projections, the new labels in Z are independent of each other and the sample imbalance problem for each class is substantially alleviated. Afterward, one binary classifier, e.g., SVM, is trained for each new label independently on the training set $\{X, Z\}$.

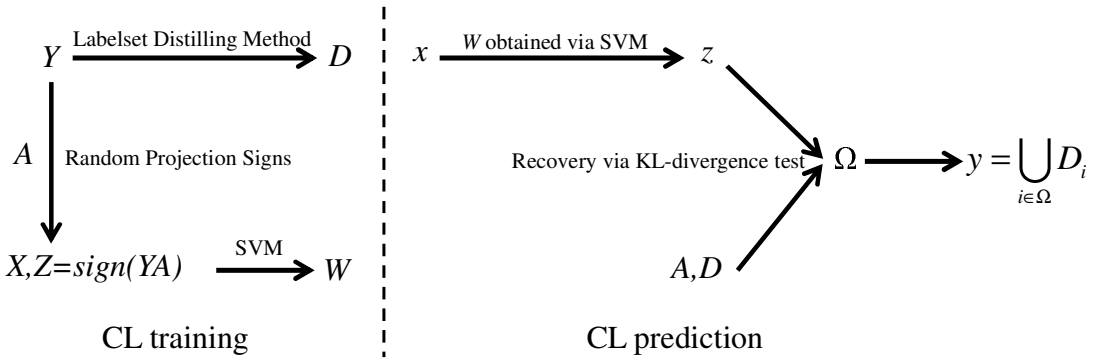

Figure 10.1: Compressed labeling on distilled labelsets. In the training stage, CL first compresses the original label matrix Y into Z, which is the sign matrix of random projections of Y on Gaussian random matrix A. Then binary classifiers (such as SVM) corresponding to the training set $\{X, Z\}$ are independently learned and stored in W. Meanwhile, the frequently appeared label subsets in Y are extracted by labelset distilling method (LDM) and stored in the distilled labelsets (DLs) D. In the prediction stage, CL first predicts the new labels z of a given sample x via the binary classifiers W. Given A and D, the DLs appearing in z are identified by a KL-divergence test based recovery algorithm and indexed by Ω. The final prediction y is the union of all the appeared DLs.

In the prediction stage, CL first predicts the new labels z of a given sample x via the binary classifiers W obtained in the training stage, and then a fast recovery algorithm is developed to recover the original labels y from the predicted new labels z. In the recovery algorithm, we predict the original label vector of the given sample via performing a series of KL divergence based hypothesis tests on the "distilled labelsets (DLs)".

DLs are the frequently appeared label subsets extracted from the original labels Y via a "label distilling method (LDM)". LDM performs a recursive clustering and subtraction on the label vectors, i.e., the rows of Y. Each distilled labelset (DL) is the intersection of the label vectors in each cluster. LDM takes the label dependence into consideration and this critical information guarantees the success of the recovery algorithm.

Given a DL and an original label vector, we discover that the signs of their random projections on a Gaussian ensemble follow an explicit joint distribution that can be quickly computed from a geometric inference. The corresponding em-

pirical joint distribution can also be quickly obtained from the new label matrix. A KL divergence based comparison between the explicit joint distribution and the empirical one indicates whether a given DL is a subset of the original label vector. Since this test includes only comparison and thresholding, the recovery algorithm is fast with linear time complexity.

We theoretically prove the recovery bound of CL by investigating the upper bounds for the probabilities of 2 types of recovery failures in CL. The probabilistic bound for recovery failures exponentially shrinks with the increasing of measurements, i.e., the dimensionality of the compressed label matrix, and with the increasing of the cardinalities of the distilled labelsets. This result soundly shows the effectiveness of label compression and the prediction improvement brought by DL.

We evaluate CL on both large-scale datasets and small-scale ones including text classification, music categorization, image annotation, scene classification, genomics and web data mining. The experiments are divided into 5 groups on 21 datasets. The first group tests the label matrix compression and recovery. The recovery accuracy, sample balance and mutual independence of new labels are evaluated. The other 4 groups test CL in multi-label prediction problems, and five different evaluation metrics are used to measure its prediction performance. Thorough comparisons between CL and BR, 3 popular multi-label learning methods, 2 SVM algorithms dealing with imbalance datasets are provided respectively. The trade-off between label compression and prediction improvement is empirically studied and analyzed. The experimental results show the effectiveness, efficiency and robustness of CL in multi-label learning.

The rest of the chapter is organized as follows. Section 10.2 presents the label matrix compression and classification in CL. Section 10.3 presents LDM and the KL divergence test based recovery algorithm of CL. Section 10.4 studies the relationship between compressed labeling and compressed sensing, and the contribution of CL to multi-label learning. Section 10.5 shows the experimental results. Section 10.6 concludes the chapter.

10.2 Compressed labeling (CL) via random projections

This section presents the label matrix compression and the subsequent classification in CL, which comprises the training stage of CL. The label matrix compression is based on the random projections of the given $\{0, 1\}$ label matrix on a Gaussian ensemble. We show that the pseudorandomness of the new label matrix results in 1) an improved balance of samples for the binary classification on each label, 2) the mutual independence among the new labels and 3) the information of original label matrix is properly preserved in a low-dimensional subspace. These improvements solve or at least reduce the three aforementioned problems in Section 10.1.1 and thus benefit simultaneously the subsequent classification.

10.2.1 Random projection signs of label matrix

Random projection [226] is a simple yet powerful technique that has been widely used in fast approximation algorithms and data recovery. It is introduced as an efficient pairwise distance calculation and approximation method according to Johnson-Lindenstrauss (JL) Lemma [135] and its variants in particular metric spaces, i.e., ℓ_2 (Euclidian) space [226], $\ell_p(0 < p < 2)$ space (stable random projection) [155], $\ell_p(p > 2)$ (high-order) space [154] and smooth manifold [52]. This property has been broadly used in fast nearest neighbor search [127], low distortion embedding [57] and hashing [58]. Compressed sensing [40] proves that an exact reconstruction of a sparse signal from a few of its random projections is possible, when the random projection ensemble satisfies restricted isometry property (RIP). Random projection also attracts attentions in matrix low-rank approximation, because the column space of a matrix's low-rank random projection is proved as a sufficiently close approximation of its principle subspace [108].

In CL, the random projection offers a different function. Since CL formulates the multi-label prediction as a classification problem rather than a linear regression problem, the label matrix after compression has to be a binary matrix rather than a real-valued one. Thus the direct utilization of random projection is im-

proper. We consider the signs of the random projections in CL, so the compressed label matrix Z is:

$$Z = \text{sign}\left(YA\right), \tag{10.1}$$

where $Z \in \{-1, 1\}^{n\times m}$ is the compressed label matrix, $Y \in \{0, 1\}^{n\times k}$ is the original label matrix, $A \in \mathbb{R}^{k\times m}$ is a random matrix whose columns are randomly sampled from an ensemble, and sign$(\cdot)$ is an element-wise sign operator. In CL, we adopt the i.i.d. standard Gaussian ensemble, i.e., entries of A are independent standard normal variables.

Although it seems that the hard thresholding of random projections in (10.1) discards partial useful information for recovery at the first glance, the information is fully retained in DLs and the $\{0, 1\}$ binary prior of the label matrix, which play critical roles in the CL recovery algorithm.

This simple label compression method provides an effective cure for the aforementioned three problems, i.e., sample imbalance, label dependence and label high dimensionality. It is self-evident that the last problem is alleviated on the CL labels, because the number of random projections m can be much less than k in CL. Therefore, the number of binary classifiers in the training stage can be substantially reduced from k to m, which significantly reduces the computational complexity.

Below, we theoretically show that CL improves the sample balance and ensures the label independence. Empirical studies of these two properties on several datasets are presented in the experimental section.

10.2.2 Improved sample balance of CL labels

Given a set of positive samples labeled by 1 and negative ones labeled by 0 for each label in multi-label learning, without loss of generality, we can define the degree-of-balance of a given label matrix $Y \in \{0, 1\}^{n\times k}$ as the average proportion of positive samples on all the different labels, i.e.,

$$balance = \frac{1}{k}\sum_{i=1}^{k}\frac{np_i}{n}, \tag{10.2}$$

where np_i is the number of positive samples on the i^{th} label. The degree-of-balance close to 0.5 yields a balanced sample set for training.

In CL labels, the degree-of-balance on each label has essential connection with the overlaps between unique labelsets of the original labels. Before investigating this phenomenon, we first give an important theorem that will be used several times in this chapter. The main significance of this theorem is that it bridges the inner distribution of the compressed label vector z (an arbitrary row of Z) with the corresponding original label vector x via a one-to-one bijection. This bijection provides a direct and efficient estimate to x from the statistics of z, which will be presented in Section 10.3.2 and 10.3.3.

Theorem 18. *(**Random projection signs of two binary vectors**) Given two nonzero binary vectors $x, y \in \{0,1\}^k$, the signs of their projections on a random vector α satisfies the following distribution, if the entries of α are independent standard Gaussian variable with unit variance and zero mean.*

$$\Pr(\text{sign}(\langle x, \alpha\rangle) \cdot \text{sign}(\langle y, \alpha\rangle) = -1) = \frac{1}{\pi} \arccos\left(\frac{\text{card}(x \cap y)}{\sqrt{\text{card}(x)}\sqrt{\text{card}(y)}}\right), \quad (10.3)$$

where $\text{card}(\cdot)$ *refers to the cardinality of a given vector.*

Proof. The proof follows a geometric inference on a sphere in a high dimensional space. Since the entries of α are i.i.d. standard Gaussian variables, α is a vector which is drawn uniformly from a hypersphere S_k in the k dimensional space. Figure 10.2 shows the random projections of x, y on two Gaussian random vectors α and β.

In Figure 10.2, we uses "+" and "−" to indicate the signs of random projections. Two hyperplanes $W1$ and $W2$ are perpendicular to x and y, respectively. Figure 10.2 verifies that if x and y are projected onto a random vector β in the two shaded regions, which determined by $W1$ and $W2$, the random projections will have opposite signs. When x and y are projected onto a random vector α in the unshaded regions, their random projection signs will be identical. Since the

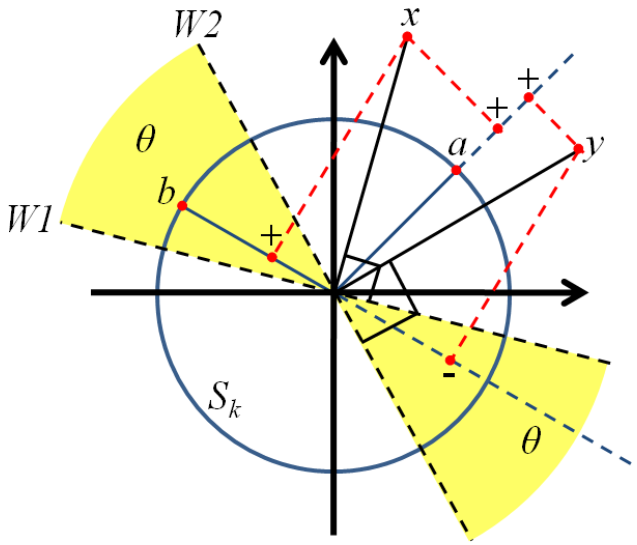

Figure 10.2: Random projections of x, y on two random vectors α and β, which are drawn uniformly from a k-dimensional hypersphere. The signs of random projections are marked as "+" for positive and "−" for negative in the figure. The hyperplanes $W1$ and $W2$ are perpendicular to x and y, respectively.

dihedral angle θ is equal to the angle between vectors x and y, i.e.,

$$\theta = \arccos\left(\frac{\langle x, y\rangle}{\|x\|_2\|y\|_2}\right) = \arccos\left(\frac{\text{card}\,(x \cap y)}{\sqrt{\text{card}(x)}\sqrt{\text{card}(y)}}\right), \tag{10.4}$$

and the Gaussian random vector α is drawn uniformly from the hypersphere, the probability that x and y have different random projection signs is proportional to the area of the shaded regions. Therefore, we have

$$\Pr(\text{sign}\,(\langle x, \alpha\rangle) \cdot \text{sign}\,(\langle y, \alpha\rangle) = -1) = \frac{2\theta}{2\pi}, \tag{10.5}$$

which completes the proof of Theorem 18.

□

Theorem 18 can be seem as an extension of Lemma 3.2 in [98]. Now we

analyze the degree-of-balance of CL labels based on Theorem 18. Without loss of generality, we assume the original label matrix Y has the following label powerset L, which consists of the unique rows of Y:

$$L = [L_1; L_2; \cdots; L_K], L_i \in \{0,1\}^k. \tag{10.6}$$

We call each unique row of Y as a "labelset". The appearance times of labelset L_i in all the rows of Y is represented as n_i, so we have $n = \sum_{i=1}^{K} n_i$. Given a vector α whose entries are randomly drawn from independent standard Gaussian distributions, if the random projection sign of L_i on α, i.e., $z_{l:Y_l=L_i} = \text{sign}(\langle L_i, \alpha \rangle)$ in the corresponding column of Z is known, the probabilities that random projection signs of the other label vectors $L_{j:j\neq i}$ in L are $-z_l$ can be obtained by using (10.3). In particular, we calculate the expected number of label vectors (rows) in Y whose random projection signs on an arbitrary row α of A are opposite to z_l, namely, the expected number of $-z_l$ in arbitrary column z of CL label matrix Z:

$$\begin{aligned}
\mathbb{E}_{z_l}(|p : z_p = -z_l|) &= \sum_{j=1,j\neq i}^{K} n_j \Pr(z_l \cdot \text{sign}(\langle L_j, \alpha \rangle) = -1) \\
&= \sum_{j=1,j\neq i}^{K} n_j \Pr(\text{sign}(\langle L_i, \alpha \rangle) \cdot \text{sign}(\langle L_j, \alpha \rangle) = -1) \\
&= \sum_{j=1,j\neq i}^{K} \frac{n_j}{\pi} \arccos\left(\sqrt{\frac{\text{card}(L_i \cap L_j)}{\text{card}(L_i)}} \cdot \sqrt{\frac{\text{card}(L_i \cap L_j)}{\text{card}(L_j)}} \right),
\end{aligned} \tag{10.7}$$

where $|\cdot|$ denotes the number of given variable, and the last step is due to Theorem 18.

The conditional expectation in (10.7) computes the expected number of samples with CL label $-z_l$ in an arbitrary column of Z given the CL label of L_i, i.e., z_l. Without loss of generality, we assume $z_l = -1$. Thus the expected degree-of-balance $\mathbb{E}_{z_l=-1}(balance)$ given $z_l = -1$ can be calculated by using (10.7). Since

the distribution of L_i in rows of Y is given by

$$\Pr(L_i) = n_i/n, \tag{10.8}$$

the unconditioned expected degree-of-balance $\mathbb{E}(balance)$ can be computed over the whole label powerset L, i.e.,

$$\begin{aligned}\mathbb{E}(balance) &= \sum_{i=1}^{K} \Pr(L_i)\mathbb{E}_{z_l=-1}(balance) = \sum_{i=1}^{K} \Pr(L_i) \cdot \frac{np_i}{n} \\ &= \sum_{i=1}^{K} \frac{n_i}{n} \cdot \frac{1}{n}\mathbb{E}_{z_l}\left(|p : z_p = -z_l|\right) \\ &= \sum_{i=1}^{K} \sum_{j=1, j\neq i}^{K} \frac{n_i n_j}{n^2\pi} \arccos\left(\sqrt{\frac{\operatorname{card}\left(L_i \cap L_j\right)}{\operatorname{card}\left(L_i\right)}} \cdot \sqrt{\frac{\operatorname{card}\left(L_i \cap L_j\right)}{\operatorname{card}\left(L_j\right)}}\right).\end{aligned} \tag{10.9}$$

In multi-label learning, a degree-of-balance close to 0.5 is preferred, because the numbers of positive samples and negative ones will be equal to each other in the training set. To see how CL improves the sample balance, we first study a special case of multi-label learning: multi-class learning. In multi-class learning, each labelset in L includes only one "1" in its entries. There is no overlap between any two different labelsets in multi-class learning, which is a special case of multi-label learning. When the number of labelsets K in L increases, the label matrix Y seriously confronts the problem of sample imbalance. That is because each class has only a few positive samples and a large amount of negative ones. In such case, we have:

$$\sum_{i=1}^{K} \sum_{j=1, j\neq i}^{K} \frac{n_i n_j}{n^2} \to 1, \tag{10.10}$$

$$\sqrt{\frac{\operatorname{card}\left(L_i \cap L_j\right)}{\operatorname{card}\left(L_i\right)}} = 0, (i \neq j). \tag{10.11}$$

Equation (10.10) is a result of a large K, while (10.11) is due to the orthogonality of the unique labelsets. Thus the expected degree-of-balance of CL labels in multi-

class learning problem can be calculated by substituting the above equations into (10.9):

$$\mathbb{E}_{multi-class}(balance) = \frac{1}{2}\sum_{i=1}^{K}\sum_{j=1,j\neq i}^{K}\frac{n_i n_j}{n^2} \rightarrow 0.5. \tag{10.12}$$

Therefore, after label compression in CL, the problem of sample imbalance in multi-class learning is substantially alleviated.

The situation in general multi-label learning is similar but different because of the existence of labelset overlapping. Without loss of generality, we consider a pair of labelsets L_i and L_j. Referring to the situation of multi-class learning, small values of

$$\sqrt{\frac{\text{card}\,(L_i \cap L_j)}{\text{card}\,(L_i)}} \quad and \quad \sqrt{\frac{\text{card}\,(L_i \cap L_j)}{\text{card}\,(L_j)}} \tag{10.13}$$

are preferred in multi-label learning, because the arccosine of their multiplication will be close to $\pi/2$. Thus the expected degree-of-balance in (10.9) will approach to 0.5. This corresponds to a small overlap $\text{card}\,(L_i \cap L_j)$ or large cardinalities $\text{card}\,(L_i)$ and $\text{card}\,(L_j)$, which are exactly in accordance with the multi-label data. That is because in multi-label data, a labelset L_i with large overlap to the other ones usually has very large cardinality, while a labelset L_i with small cardinality often has ultra-small overlap to the other ones. In both of these two cases, the two values in (10.13) will be kept close to 0.

Even when L_i and L_j share a large overlap and both have small cardinality, the degree-of-balance of CL labels will not deviate far away from 0.5. That is because this case causes small n_i and n_j, which results in a small weight $n_i n_j/n^2\pi$ in (10.9) to eliminate the influence of small arccosine in (10.9). Therefore, $\mathbb{E}(balance)$ approaches to 0.5.

We place an empirical study of the sample balance of CL labels on various multi-label datasets in the experimental section. The result demonstrates that the sample degree-of-balance after label compression in CL is significantly improved and is close to the ideal value 0.5.

10.2.3 Mutual independence of CL labels

The mutual independence between CL labels after label compression is a natural result of random projection sign. To see this, we have the following theorem:

Theorem 19. *(**Label independence**) Given a binary and nonzero vector $x \in \{0,1\}^k$, its CL labels obtained via random projection signs $z = \text{sign}(xA)$ are independent random variables, if A is a standard Gaussian matrix with entries drawn from independent standard Gaussian distribution.*

Proof. Consider two random variables $y_i = xA_i$ and $y_j = xA_j$, if A_i and A_j are both composed by independent standard Gaussian variables, y_i and y_j are weighted sum of independent standard Gaussian variables. Thus the two variables y_i and y_j are independent Gaussian variables. Since $z_i = \text{sign}(y_i)$ and $z_j = \text{sign}(y_j)$, the random variables z_i and z_j are independent. Therefore, the CL labels in z are mutual independent random variables. Two similar analyses can be found in [108][226]. This completes the proof. □

The mutual independence between CL labels after compression is not equal to a sheer discard of the label dependence information in the original labels. Actually, we extract and store the label dependence information in the distilled labelsets (DLs) that consists of the most frequent label subsets in the original label matrix, and use it in the recovery algorithm. We transform the original labels into independent ones before the training stage, because the conventional binary classification methods cannot use the label dependence, and will be even harmed by the label dependence under some circumstances. Therefore, CL isolates the application of label dependence from the training stage in its scheme in order to apply binary classification methods without loss of useful information.

The label compression preserves the distribution and the pairwise distance of the original label vectors in the low-dimensional space, though it reduces the number of labels. The pairwise-distance preservation of random projections has been proved in various scenarios [226]. Its variant, the random projection signs, has also been proved as a pairwise-distance preservation method in terms of the cosine distance [190][98]. Therefore, the classifications on the compressed labels will not be more difficult than on the original labels.

10.2.4 Classification via support vector machines

After obtaining the CL label matrix Z via random projections of the original label matrix Y, we train one binary classifier for each new label on the CL label matrix Z. There are a large number of binary classification methods with appealing performance and fast speed. Most of them can be directly applied to the training stage of CL.

Among the existing binary classification methods, support vector machine (SVM) [224] has been widely used because of its appealing properties in statistical machine learning theory, optimality in classification hyperplane design, robustness to different types of data, extensionality to nonlinear kernel space, and many existing fast solvers. In this chapter, we adopt SVM as the binary classification method used in the training stage of CL.

10.3 Recovery algorithm on distilled labelsets (DLs)

In this section, we introduce the label distilling method (LDM) and the label recovery algorithm in CL, which comprise the prediction stage of CL. In recovery, given the CL label vector z predicted by the m binary classifiers from a sample x and a $\{0,1\}$ binary dictionary D for label vectors, CL predicts the original label vector y by testing whether each binary vector D_i is included in y, i.e., whether $D_i \cap y = D_i$ and then recovers $y = \bigcup_{i\in\Omega} D_i$ (Ω is the index set of D_i included in y). The recovery is based on the fact that the random projection signs of y and D_i have explicit joint distributions in two cases, i.e, $D_i \cap y = D_i$ and $D_i \cap y = \emptyset$. In the recovery algorithm, a hypothesis test is designed to decide whether $D_i \cap y = D_i$ by comparing the Kullback-Leibler (KL) divergence [144] between the empirical joint distribution of the random projection signs and the two explicit joint distributions in the two cases. An available and natural choice of D_i is unit vector. In this case, the CL recovery algorithm element-wisely recovers y, i.e., each label is recovered independently. However, in CL, we develop LDM to obtain D. In particular, LDM extracts the most frequent label subsets, i.e., distilled labelsets (DLs) $D \in \{0,1\}^{d\times k}$, by recursive clustering and subtraction

of the label vectors (rows) in the label matrix Y of training set. Therefore, CL exploits label dependence via jointly recovering the correlated labels in y on DLs. We will show that DLs improves the accuracy of the recovery algorithm by increasing the KL divergence between the two explicit joint distributions in the two cases.

10.3.1 Labelset distilling method (LDM)

The discrete patterns frequently appearing in the label matrix Y refers to the label subsets that are frequently shared by the rows of Y. These patterns reveal the structural information of the binary matrix Y and label dependence embedded in the given label vectors.

In multi-label learning, BR simply ignores the label dependence information, while LP treats the unique labelsets independently and thus ignores the shared information of different label vectors. Recently, several methods have been developed to exploit the label dependence by building a tree-structural hierarchy for the labels. However, the correlation (e.g., co-occurrence and mutual exclusion) between two labels are probabilistic rather than deterministic. Therefore, it is hard to prune the tree hierarchy without discarding the minority instances. A tree hierarchy that retains most leafs explains few label dependence information and will significantly increase the problem size.

We propose "labelset distilling method (LDM)" method to exploit the correlations between the unique labelsets of L rather than single labels. LDM can be interpreted as a greedy search for the frequent discrete patterns of L. It decomposes $\{0,1\}$ label matrix L as:

$$L = UD, U \in \{0,1\}^{K\times d}, D \in \{0,1\}^{d\times k}. \tag{10.14}$$

The obtained dictionary D is called "distilled labelsets (DLs)", each row of which is a "distilled labelset (DL)".

In LDM, the above decomposition is accomplished by a greedy search of discrete patterns. The greedy search is a recursive clustering and subtraction of the labelsets (rows) in L. It is described by the following procedure:

1 The rows of L are clustered by using an existing clustering algorithms, e.g., spectral clustering [165][182] or k-means [171]. In our experiments, we use spectral clustering.

2 The shared binary pattern of each cluster is extracted as a row D_i in the DLs D and then subtracted from the labelsets L_i in the cluster.

3 For the clusters without shared pattern, labelsets in them are kept the same in the label powerset L. For the labelsets L_i that become all-zero vectors after substraction, we remove them from L. The other labelsets L_i after subtraction are updated in L.

4 Update coefficients in U corresponding to the newly extracted atoms in D.

The above procedure is iterated until the label powerset L is empty, i.e., all the unique labelsets L_i in the initial L are completely represented by the atoms in the dictionary D.

We use spectral clustering to group the rows of L, because the number of clusters obtained by spectral clustering in each iteration can be adaptively adjusted by a given threshold τ. In particular, we sort q eigenvalues of the Graph Laplacian from small to large, and compute the following metric for each one:

$$e_i = \frac{\sum_{j=i}^{q} \lambda_j}{\sum_{j=1}^{q} \lambda_j}, i = 2, \cdots, q. \tag{10.15}$$

Only the eigenvectors with $e_i < \tau$ are selected for the subsequent processing (including k-means and thresholding). The number of selected eigenvectors is the number of clusters in the iteration. A properly selected parameter τ can efficiently generate clusters with shared labelsets. Empirically, we select $0.01 \leq \tau \leq 0.25$ in all experiments. This fact will be verified in our experiments.

We show LDM in Algorithm 12.

10.3.2 Joint distribution of two random projection signs

An interesting phenomenon in CL is that given a label vector L_i, the signs of its projection and an arbitrary D_i's projection on a standard Gaussian random

Algorithm 12: labelset distilling method (LDM)

Input: Label powerset $L \in \{0,1\}^{K\times k}$, threshold τ
Output: Distilled labelsets D, coefficient matrix U
Initialize: $D := \emptyset$, $U := \emptyset$
while *The rows of* L *are not empty* **do**
  Cluster the rows of L into t clusters by `Spectral Clustering` with threshold τ;
  for $i \leftarrow 1$ **to** t **do**
    Extract the label subset $D_i \in \{0,1\}^k$ shared by the label vectors in $Cluster\ \ i$, i.e., $D_i = \bigcap_j \{L_j : L_j \in Cluster\ \ i\}$;
    Subtract the obtained D_i from all the label vectors in $Cluster\ \ i$, i.e., $L_j := L_j - D_i$ for $\{L_j : L_j \in Cluster\ \ i\}$;
    if $\{L_j : L_j \in Cluster\ \ i\} = \boldsymbol{0}$ **then**
      Remove L_j from L
    end
    Add the extracted D_i into the distilled labelsets D as a new row;
    Add the corresponding coefficient vector $U_i^T \in \{0,1\}^K = \left\{U_{ij}^T = 1 \ \text{ if } \ L_j \in Cluster, \text{else } \ U_{ij}^T = 0\right\}$ into the coefficient matrix U as a new column;
  end
end

vector α have an explicit joint distribution that can be quickly computed. Based on this fact, the existence of a distilled labelset D_i in an unknown label vector y can be tested by using the information of their random projection signs.

In particular, the following theorem states the random projection signs of y and D_i follows an explicit joint distribution.

Theorem 20. *(**Joint distribution of random projection signs**) Given a label vector $y \in \{L_1, L_2, \cdots, L_K\}$ that is selected from the rows of the label powerset L, let D be L's distilled labelsets (DLs) that satisfy $L = UD$. If D_i is included in y, i.e., $D_i \cap y = D_i$, the signs of their random projections on a standard*

Gaussian random vector α follows the following joint distribution $P1$:

$$P1(1) = \Pr(\text{sign}(\langle y, \alpha\rangle) \cdot \text{sign}(\langle D_i, \alpha\rangle) = -1) = \frac{1}{\pi}\arccos\left(\sqrt{\frac{\text{card}(D_i)}{\text{card}(y)}}\right), \tag{10.16}$$

$$P1(2) = \Pr(\text{sign}(\langle y, \alpha\rangle) \cdot \text{sign}(\langle D_i, \alpha\rangle) = 1) = 1 - \frac{1}{\pi}\arccos\left(\sqrt{\frac{\text{card}(D_i)}{\text{card}(y)}}\right). \tag{10.17}$$

If D_i is not included in y, i.e., $D_i \cap y = \emptyset$, the signs of their random projections on a standard Gaussian random vector α follows the following joint distribution $P2$:

$$P2(1) = \Pr(\text{sign}(\langle y, \alpha\rangle) \cdot \text{sign}(\langle D_i, \alpha\rangle) = -1) = \frac{1}{2}, \tag{10.18}$$

$$P2(2) = \Pr(\text{sign}(\langle y, \alpha\rangle) \cdot \text{sign}(\langle D_i, \alpha\rangle) = 1) = \frac{1}{2}. \tag{10.19}$$

Both $P1(l)$ and $P2(l)$ ($l = \{1, 2\}$) refer to the corresponding two probabilities associated with the two cases of the two random projection signs' product.

Proof. By substituting D_i and y into Theorem 18, the above distributions can be directly obtained. This completes the proof. Theorem 20 and the subsequent lemmas about the recovery bounds of CL are based on Theorem 18 and are independent of Theorem 19. □

Note that only the cardinality information of the label vector y is required in computing the joint distribution given in Theorem 20. Therefore, if we are given the distilled labelsets D and y's cardinality, the joint distribution of an arbitrary D_i and y's random projection signs can be explicitly computed by using Theorem 20.

10.3.3 KL divergence test for recovery

Given classifiers W obtained in the training stage, the CL labels of a sample $x \in \mathbb{R}^p$ can be predicted as $z \in \{0, 1\}^m$. Our goal in the recovery algorithm is

to reconstruct the corresponding original label vector $y \in \{0,1\}^k$ from z. In CL, we propose a recovery algorithm via testing the KL divergence between possible joint distributions of random projection signs and the corresponding empirical joint distribution.

According to Theorem 20, given the cardinality of y's, the joint distribution of D_i and y's random projection signs can be quickly computed in two scenarios: D_i is included in y and D_i is not included in y. The corresponding empirical joint distribution $\hat{P}$ can be calculated from $v = \text{sign}\,(D_i A)$ and $z = \text{sign}\,(yA)$, i.e.,

$$\hat{P}(1) = \hat{\Pr}(\text{sign}\,(\langle y, \alpha\rangle) \cdot \text{sign}\,(\langle D_i, \alpha\rangle) = -1) = \frac{|j : z_j v_j = -1|}{m}, \tag{10.20}$$

$$\hat{P}(2) = \hat{\Pr}(\text{sign}\,(\langle y, \alpha\rangle) \cdot \text{sign}\,(\langle D_i, \alpha\rangle) = 1) = \frac{|j : z_j v_j = 1|}{m}. \tag{10.21}$$

The $\hat{P}(l)$ $(l = \{1,2\})$ refers to the corresponding two estimated probabilities associated with the two cases of the two random projection signs' product.

In the following discussion, the joint distribution associated with the situation that D_i is included in y is marked as $P1$, while the joint distribution associated with the situation that D_i is not included in y is marked as $P2$. Both $P1$ and $P2$ are defined in Theorem 20. We denote the empirical joint distribution computed from $v = \text{sign}\,(D_i A)$ and z as $\hat{P}$.

The target of recovery in CL is to determine whether D_i is included in y or not. This can be done by comparing the distance between $P1$ and $\hat{P}$ and that between $P2$ and $\hat{P}$. A smaller distance between $P1$ and $\hat{P}$ indicates a higher probability that D_i is included in y, while a smaller distance between $P2$ and $\hat{P}$ indicates a higher probability that D_i is not included in y.

KL divergence, also known as relative entropy, measures the distance between two probability distributions P and Q,

$$D_{KL}\,(P \| Q) = \sum_i P(i) \log \frac{P(i)}{Q(i)}. \tag{10.22}$$

In the recovery algorithm of CL, given a y, we use KL divergence to measure the distance between $P1$ and $\hat{P}$ and the distance between $P2$ and $\hat{P}$ for different D_i. The differences between the two distances for all D_i are sorted. Then D_i

with $\hat{P}$ closer to $P1$ than $P2$ are sequentially added into y as a subset from larger distance difference to smaller one until y reaching its cardinality. In particular, given the predicted CL label vector z of y, we calculate the following two KL divergences on each distilled labelset D_i in D:

$$M1_i = D_{KL}\left(P1 \| \hat{P}\right) = \sum_{j=1}^{2} P1(j) \log \frac{P1(j)}{\hat{P}(j)}, \tag{10.23}$$

$$M2_i = D_{KL}\left(P2 \| \hat{P}\right) = \sum_{j=1}^{2} P2(j) \log \frac{P2(j)}{\hat{P}(j)}. \tag{10.24}$$

The difference between $M1_i$ and $M2_i$ on each D_i forms a difference vector $Diff$:

$$Diff = M2 - M1, Diff \in \mathbb{R}^d. \tag{10.25}$$

The positive entries of $Diff$ corresponds to the distilled labelsets whose empirical joint distribution $\hat{P}$ is closer to $P1$ than $P2$, which indicates a higher possibility that the corresponding D_i are included in y. In addition, a larger and positive $Diff_i$ implies D_i is more possible to be the subset of y than the other DLs with positive difference in $Diff$. In the recovery algorithm, we sort the positive entries of $Diff$ from large to small, choose D_i sequentially from the ones with large $Diff_i$ to the ones with small $Diff_i$, and add the selected D_i to y until the cardinality of y is arrived.

Since the cardinality of y cannot always be known previously in the prediction, we adds an outer loop to the above recovery procedure and searches the cardinality with the smallest recovery error $\|z - \text{sign}\,(y_i A)\|$ in a given range. In CL, we choose it as the cardinality range of the training label vectors in Y.

We show the recovery algorithm of CL in Algorithm (13) and highlight advantages of the recovery algorithm below,

1 The recovery algorithm based on the KL divergence comparison only includes simple computations, e.g., comparison, sorting and thresholding. Thus the CL recovery algorithm is much faster than the normal compressed sensing algorithms used in [122].

2 Since the joint distribution of the random projection signs and the cor-

Algorithm 13: Recovery algorithm of CL

Input: CL label vector $z \in \{-1, 1\}^m$, distilled labelsets D, Gaussian random matrix A used in compression, cardinality range $[card1, card2]$

Output: original label vector $y \in \{0, 1\}^k$

Calculate $V = \text{sign}(DA)$ with $V_i = \text{sign}(D_i A)$;
Calculate the joint distribution $P2$ by using (10.18) and (10.19);
for $i \leftarrow card1$ **to** $card2$ **do**
  for $j \in \{j : \text{card}(D_j) \leq i\}$ **do**
    Calculate the empirical joint distribution $\hat{P}_j$ by using (10.20) and (10.21) with $v = V_j$;
    Calculate the joint distribution $P1_j$ by using (10.16) and (10.17), wherein $\text{card}(y) = i$;
    Calculate `KL divergence` between $P2$ and $\hat{P}_j$, i.e., $M2_j = \sum_{l=1}^{2} P2(l) \log \frac{P2(l)}{\hat{P}_j(l)}$;
    Calculate `KL divergence` between $P1_j$ and $\hat{P}_j$, i.e., $M1_j = \sum_{l=1}^{2} P1_j(l) \log \frac{P1_j(l)}{\hat{P}_j(l)}$;
    Calculate the difference between the two KL divergences $M2_j$ and $M1_j$, i.e., $Diff_j = M2_j - M1_j$;
  end
  Sort the positive entries in $Diff$ from large to small;
  $y_i := \mathbf{0}, j := 1$;
  while $\text{card}(y_i) < i$ **do**
    Add the distilled labelset D_j which is associated with the j^{th} largest entry in sorted positive $Diff$ to y_i, i.e., $y_i := y_i \cup D_j$;
  end
  Calculate the recovery error $error_i = \|z - \text{sign}(y_i A)\|$;
end
Return $y = y_i$, wherein $i = \arg\min_i error_i$;

responding empirical distribution can be explicitly computed in the CL recovery algorithm, we can directly compare the distributions on their KL divergence rather than on their means or variances. This KL divergence comparison provides more useful information for testing whether a given DL is included in y. Compared with the mean test used in 1-bit com-

pressed sensing, a direct comparison of distributions outputs more precise test result.

3 In the recovery algorithm, we test the existence of the distilled labelsets D rather than the single labels or random labelsets in y. This is an application of label dependence information in the multi-label prediction, because the distilled labelsets are the most frequent label subsets and the significant discrete patterns mining from the training label matrix. To see the benefits brought by this exploiting of label dependence, we compare the gaps between $P1$ and $P2$ in two situations, i.e., using distilled labelsets and using single labels in the test. Assume the dictionary corresponding to the single label situation is:

$$E = [E_1; E_2; \cdots; E_K], E_i \in \{0,1\}^{n\times k}, \text{card}\,(E_i) = 1. \tag{10.26}$$

According to the two joint distributions $P1$ and $P2$ given in Theorem 20, we can calculate the differences between $P1(l)$ and $P2(l)$ for all the $l = \{1, 2\}$:

$$\|P1(l) - P2(l)\| = \left\| \frac{1}{2} - \frac{1}{\pi} \arccos\left(\sqrt{\frac{\text{card}(D_i)}{\text{card}(y)}} \right) \right\|. \tag{10.27}$$

When dictionary generated by the single labels E is used, we have $\text{card}(D_i) = 1$ in (10.27). When distilled labelsets D is used instead, we have $\text{card}(D_i) > 1$ in (10.27). Thus the gap between $P1$ and $P2$ measured by $\|P1(l) - P2(l)\|$ is larger when distilled labelsets D is adopted. A larger gap between $P1$ and $P2$ will substantially reduce the number of failures in the tests of $\hat{P}$. Therefore, the recovery accuracy is improved in CL by using the label dependence information embedded in DL.

We show the training and prediction algorithms of CL in Algorithm 14 and Algorithm 15, respectively.

10.3.4 Recovery bound

In order to investigate whether and when the CL recovery shown in Algorithm 13 is sufficiently accurate for reconstructing the original label vector y, we theoret-

Algorithm 14: Training algorithm of CL

Input: Data matrix $X \in \mathbb{R}^{n \times p}$, label matrix $Y \in \{0,1\}^{n \times k}$, label compression dimension m, threshold τ, parameters for `SVM solver`

Output: m classifiers $W \in \mathbb{R}^{p \times m}$, distilled labelsets D, Gaussian random matrix A

label compression via random projections;
Generate a standard Gaussian random matrix $A \in \mathbb{R}^{k \times m}$;
Calculate the CL label matrix $Z = \text{sign}(YA)$;
classification via support vector machines;
for $i \leftarrow 1$ **to** m **do**
    Train binary classifier W_i on training set $\{X, Z_i\}$ by standard `SVM solver`, wherein Z_i is the i^{th} column of Z;
end
$W = [W_1, W_2, \cdots, W_m]$;
distilled labelsets extraction;
Calculate the label powerset L, whose rows are unique label vectors in Y;
Extract the `Distilled labelsets (DL)` with input L and τ by using Algorithm 12;

Algorithm 15: Prediction algorithm of CL

Input: Sample $x \in \mathbb{R}^p$, CL classifiers $W \in \mathbb{R}^{p \times m}$, distilled labelsets D, Gaussian random matrix A used in compression, cardinality range $[card1, card2]$

Output: Label vector y

Calculate the predicted CL label vector $z \in \{-1,1\}^m$ of x by using CL classifiers W, i.e., $z = \text{sign}(xW)$;
Run `Recovery algorithm of CL` with input z, D, A and $[card1, card2]$ by using Algorithm 13;
Return the output y;

ically analyze the upper bounds for probabilities of 2 types of recovery failures, i.e.,

1 Type I failure: accepting labelset $b = D_i, \forall i = 1 : d$ included in y (i.e., $b \cap y = b$) when it is actually not (i.e., $b \cap y = \emptyset$). According to the recovery algorithm, this failure happens when the KL divergence between $P1$ and $\hat{P}$ is smaller than that between $P2$ and P, but P is actually estimated according to the samples from $P2$. The probability of this type of failure is

$$\Pr\left(D_{KL}\left(P1\|\hat{P}\right) < D_{KL}\left(P2\|\hat{P}\right)\right), \quad \text{when} \quad \hat{P} = \hat{P}2, i.e., b \cap y = \emptyset. \tag{10.28}$$

2 Type II failure: Excluding labelset $b = D_i, \forall i = 1 : d$ from y (i.e., $b \cap y = \emptyset$) when it is actually included in y (i.e., $b \cap y = b$). According to the recovery algorithm, this failure happens when the KL divergence between $P1$ and $\hat{P}$ is larger than that between $P2$ and P, but P is actually estimated according to the samples from $P1$. The probability of this type of failure is

$$\Pr\left(D_{KL}\left(P1\|\hat{P}\right) > D_{KL}\left(P2\|\hat{P}\right)\right), \quad \text{when} \quad \hat{P} = \hat{P}1, i.e., b \cap y = b. \tag{10.29}$$

If the recovery algorithm is applied as a prediction model like in CL, from the view point of classification, the type I failure corresponds to *false positive* and its probability denotes $1 - specificity$, while the type II failure corresponds to *false negative* and its probability denotes $1 - sensitivity$. Thus, if the probabilities of the two types of failures can both be upper bounded by small probabilities, the prediction of CL will produce a satisfactory ROC (receiver operating characteristic).

In the following proofs, we derive the upper bounds for the probabilities of the two types of failures. Firstly, we show the difference between the 2 KL divergences is determined by $\hat{P}_1$, which is a binomial random variable. We give the distributions of $\hat{P}_1$ in the two cases in Lemma 3. In Lemma 4, we study the conditions of $\hat{P}_1$ that lead to the two types of failures. Some significant properties of parameter δ in the conditions are provided in Lemma 5. By using the distributions of $\hat{P}_1$ and its conditions to cause the failures, we compute the probabilities of the failures in Proposition 10. Then the probabilistic bounds for the failures is derived in Theorem 21 based on Hoeffding's inequality. We analyze the improvement of measurement increase and distilled labelsets on the recovery bounds in Theorem 22.

For brevity of the analysis, we use the abbreviations $P_{1i} = P1(i)$, $P_{2i} = P2(i)$ and $\hat{P}_i = \hat{P}(i)$ for $i = 1, 2$. According to their definitions in (10.16)-(10.21), define

$$\gamma = P_{11} = \frac{1}{\pi} \arccos\left(\sqrt{\frac{\text{card}(b)}{\text{card}(y)}}\right) \in \left[0, \frac{1}{2}\right), \tag{10.30}$$

we calculate the difference between the 2 KL-divergences $D_{KL}\left(P1\|\hat{P}\right)$ and $D_{KL}\left(P2\|\hat{P}\right)$:

$$\begin{aligned}
&D_{KL}\left(P1\|\hat{P}\right) - D_{KL}\left(P2\|\hat{P}\right) = \sum_{i=1}^{2} P_{1i} \log \frac{P_{1i}}{\hat{P}_i} - \sum_{i=1}^{2} P_{2i} \log \frac{P_{2i}}{\hat{P}_i} \\
=&P_{11} \log \frac{P_{11}}{\hat{P}_1} + P_{12} \log \frac{P_{12}}{\hat{P}_2} - P_{21} \log \frac{P_{21}}{\hat{P}_1} - P_{22} \log \frac{P_{22}}{\hat{P}_2} \\
=&\gamma \log \frac{\gamma}{\hat{P}_1} + (1-\gamma) \log \frac{1-\gamma}{\hat{P}_2} - \frac{1}{2} \log \frac{1}{2\hat{P}_1} - \frac{1}{2} \log \frac{1}{2\hat{P}_2} \\
=&\left(\frac{1}{2} - \gamma\right) \log \frac{\hat{P}_1}{\hat{P}_2} + \gamma \log \gamma + (1-\gamma) \log (1-\gamma) + \log 2.
\end{aligned} \tag{10.31}$$

Therefore, the difference between the 2 KL-divergences is determined by the two variables $\hat{P}_1$ and $\hat{P}_2 = 1 - \hat{P}_1$, which distributions can be obtained by the following Lemma.

Lemma 3. *Given m measurements of random projection signs, the $m+1$ possible discrete values of $\{\hat{P}_1, \hat{P}_2\}$ are*

$$\hat{P}(j) = \left\{\hat{P}_1 = \frac{j}{m}, \hat{P}_2 = \frac{m-j}{m}\right\}, j = 0 : m. \tag{10.32}$$

where j is the number of -1s in the m values of $z_i v_i$ associated with $i = 1 : m$, while $m - j$ is the number of 1s. The probability of $\hat{P}(j)$ follows the following binomial distribution when $b \cap y = \emptyset$:

$$\Pr\left(\hat{P}(j)\right) = \binom{m}{j} \left(\frac{1}{2}\right)^m. \tag{10.33}$$

The probability of $\hat{P}(j)$ follows the following binomial distribution when $b \cap y = b$:

$$\Pr\left(\hat{P}(j)\right) = \binom{m}{j} (\gamma)^j (1-\gamma)^{m-j}. \tag{10.34}$$

Proof. According to the definition of $\hat{P}$ in (10.20) and (10.21), $\hat{P}_1$ is the estimation

for the probability of event $z_i v_i = -1$, while $\hat{P}_3 + \hat{P}_4$ is the estimation for the probability of event $z_i v_i = 1$. Thus the event $\hat{P}(j)$ is equivalent to

$$\hat{P}(j) = \{|i : z_i v_i = -1| = j, |i : z_i v_i = 1| = m - j\}, j = 0 : m. \tag{10.35}$$

By using the results of Theorem 20 and Theorem 18, the distribution of random projection sign $z_i v_i$ can be obtained:

$$\begin{cases} \Pr(z_i v_i = -1) = \frac{1}{2}, \Pr(z_i v_i = 1) = \frac{1}{2}, & b \cap y = \emptyset; \\ \Pr(z_i v_i = -1) = \gamma, \Pr(z_i v_i = 1) = 1 - \gamma, & b \cap y = b. \end{cases} \tag{10.36}$$

Since the m values of $z_i v_i$ for different i are independent to each other, and the 2 distributions of each $z_i v_i$ under the two conditions are both Bernoulli distributions, $\hat{P}(j)$ follows the following binomial distribution:

$$\Pr\left(\hat{P}(j)\right) = \begin{cases} \binom{m}{j} \left(\frac{1}{2}\right)^j \left(\frac{1}{2}\right)^{m-j}, & b \cap y = \emptyset; \\ \binom{m}{j} (\gamma)^j (1-\gamma)^{m-j}, & b \cap y = b. \end{cases} \tag{10.37}$$

This leads to Lemma 3. □

By substituting (10.32) into (10.31), the difference between the 2 KL-divergences can be expressed as a function of j:

$$D_{KL}\left(P1 \| \hat{P}(j)\right) - D_{KL}\left(P2 \| \hat{P}(j)\right) = \left(\frac{1}{2} - \gamma\right) \log \frac{j}{m-j} + \gamma \log \gamma + (1-\gamma) \log(1-\gamma) + \log 2. \tag{10.38}$$

Therefore, given the original label vector y and a labelset $b \in D$, the sign of the difference between the 2 KL-divergences is determined by j. In the following lemma, we study which j will lead to the 2 types of failures.

Lemma 4. *When the probability estimation $\hat{P} = \hat{P}(j)$, with definition*

$$\delta = \left[\frac{1}{4\gamma(1-\gamma)}\right]^{\frac{1}{1-2\gamma}} \cdot \frac{\gamma}{1-\gamma}, \tag{10.39}$$

for KL-divergences $D_{KL}\left(P1\|\hat{P}(j)\right)$, $D_{KL}\left(P2\|\hat{P}(j)\right)$ *and integer* $j \in [0, m]$, *we have*

$$D_{KL}\left(P1\|\hat{P}(j)\right) > D_{KL}\left(P2\|\hat{P}(j)\right), \forall j \in \left[\overline{\frac{m\delta}{1+\delta}}, m\right], \tag{10.40}$$

$$D_{KL}\left(P1\|\hat{P}(j)\right) < D_{KL}\left(P2\|\hat{P}(j)\right), \forall j \in \left[0, \underline{\frac{m\delta}{1+\delta}}\right], \tag{10.41}$$

where $\overline{x}$ *denotes the smallest integer larger than* x *and* $\underline{x}$ *denotes the largest integer smaller than* x.

Proof. According to (10.38), we have the following equivalences:

$$\begin{aligned}
&D_{KL}\left(P1\|\hat{P}(j)\right) > D_{KL}\left(P2\|\hat{P}(j)\right) && \Longleftrightarrow \\
&D_{KL}\left(P1\|\hat{P}(j)\right) - D_{KL}\left(P2\|\hat{P}(j)\right) > 0 && \Longleftrightarrow \\
&\left(\frac{1}{2} - \gamma\right)\log\frac{j}{m-j} > -\gamma\log\gamma - (1-\gamma)\log(1-\gamma) - \log 2 && \Longleftrightarrow \\
&\log\frac{j}{m-j} > \log\left(\left[\frac{1}{4\gamma(1-\gamma)}\right]^{\frac{1}{1-2\gamma}} \cdot \frac{\gamma}{1-\gamma}\right) && \Longleftrightarrow
\end{aligned}$$

$$j > \frac{m\delta}{1+\delta}. \tag{10.42}$$

Therefore, the necessary and sufficient condition for $D_{KL}\left(P1\|\hat{P}\right) > D_{KL}\left(P2\|\hat{P}\right)$ is

$$j > \frac{m\delta}{1+\delta}. \tag{10.43}$$

The same derivation leads to the necessary and sufficient condition for $D_{KL}\left(P1\|\hat{P}\right) < D_{KL}\left(P2\|\hat{P}\right)$:

$$j < \frac{m\delta^2}{1+\delta^2}. \tag{10.44}$$

Since j is an integer between 0 and m, these conditions lead to (10.40) and (10.41). This completes the proof. □

Before investigating the probabilities of the 2 types of failures, 5 significant properties of δ must be discussed.

Lemma 5. *The parameter δ defined in (10.39) has the following properties:*

$$\frac{\partial \delta}{\partial \gamma} > 0, \tag{10.45}$$

$$\delta < 1, \tag{10.46}$$

$$\frac{\delta}{1+\delta} - \frac{1}{2} < 0, \tag{10.47}$$

$$\frac{\delta}{1+\delta} - \gamma > 0, \tag{10.48}$$

$$\frac{\partial \left(\frac{\delta}{1+\delta} - \gamma\right)}{\partial \gamma} < 0. \tag{10.49}$$

Proof. Since δ is a function of γ, its derivative can be calculated according to fundamental differentiation rules:

$$\frac{\partial \delta}{\partial \gamma} = \frac{\partial \left(\frac{1}{4\gamma(1-\gamma)}\right)^{\frac{1}{1-2\gamma}}}{\partial \gamma} \cdot \frac{\gamma}{1-\gamma} + \left(\frac{1}{4\gamma\left(1-\gamma\right)}\right)^{\frac{1}{1-2\gamma}} \cdot \frac{\partial \frac{\gamma}{1-\gamma}}{\partial \gamma}. \tag{10.50}$$

Define function φ as

$$\varphi = \left(\frac{1}{4\gamma\left(1-\gamma\right)}\right)^{\frac{1}{1-2\gamma}}. \tag{10.51}$$

Then its derivative can be calculated by computing the logarithm of both sides:

$$\ln \varphi = \frac{1}{1-2\gamma} \ln \frac{1}{4\gamma\left(1-\gamma\right)}. \tag{10.52}$$

Computing the derivatives of both sides yields

$$\frac{\varphi'}{\varphi} = \frac{2}{\left(1-2\gamma\right)^2} \ln \frac{1}{4\gamma\left(1-\gamma\right)} - \frac{1}{\gamma\left(1-\gamma\right)}. \tag{10.53}$$

Thus we have

$$\varphi' = \frac{\partial \left(\frac{1}{4\gamma(1-\gamma)}\right)^{\frac{1}{1-2\gamma}}}{\partial \gamma} = \frac{2\varphi}{\left(1-2\gamma\right)^2} \ln \frac{1}{4\gamma\left(1-\gamma\right)} - \frac{\varphi}{\gamma\left(1-\gamma\right)}. \tag{10.54}$$

By substituting (10.51) into (10.47) and using the definition of φ, we obtain

$$\frac{\partial\delta}{\partial\gamma} = \frac{2\gamma\varphi}{(1-\gamma)(1-2\gamma)^2}\ln\frac{1}{4\gamma(1-\gamma)}. \tag{10.55}$$

The inequality $\gamma \in [0, 1/2)$ yields

$$1-\gamma > 0, \ln\frac{1}{4\gamma(1-\gamma)} > 0, \varphi > 0. \tag{10.56}$$

These lead to $\frac{\partial\delta}{\partial\gamma} > 0$, which completes the proof of (10.45).

Since $\frac{\partial\delta}{\partial\gamma} > 0$ and $\gamma \in [0, 1/2)$, we have

$$\begin{aligned}\log\delta &< \lim_{\gamma\to 1/2}\log\delta = \lim_{\gamma\to 1/2}\left[\frac{1}{1-2\gamma}\log\frac{1}{4\gamma(1-\gamma)} + \log\frac{\gamma}{1-\gamma}\right] \\ &= \lim_{\gamma\to 1/2}\left[\frac{-\log 4\gamma(1-\gamma)}{1-2\gamma}\right] = \lim_{\gamma\to 1/2}\left[\frac{-\partial\log 4\gamma(1-\gamma)}{\partial(1-2\gamma)}\right] \\ &= \frac{1-2\gamma}{2\gamma(1-2\gamma)} = 0.\end{aligned} \tag{10.57}$$

The monotonicity of logarithm and $\log\delta < 0$ yield $\delta < 1$. This completes the proof of (10.46).

By using (10.55), we have

$$\begin{aligned}\frac{\partial\frac{\delta}{1+\delta}}{\partial\gamma} &= \frac{\partial\frac{\delta}{1+\delta}}{\partial\delta}\cdot\frac{\partial\delta}{\partial\gamma} \\ &= \frac{1}{(1+\delta)^2}\cdot\frac{2\gamma\varphi}{(1-\gamma)(1-2\gamma)^2}\ln\frac{1}{4\gamma(1-\gamma)} &(10.58)\\ &= \frac{2\delta}{(1+\delta)^2(1-2\gamma)^2}\ln\frac{1}{4\gamma(1-\gamma)} > 0. &(10.59)\end{aligned}$$

Since $\gamma \in [0, 1/2)$, we have

$$\frac{\delta}{1+\delta} < \left.\frac{\delta}{1+\delta}\right|_{\gamma=1/2} = \frac{1}{2}. \tag{10.60}$$

This completes the proof of (10.47).

It is tedious and unnecessary to prove inequalities (10.48) and (10.49) via

similar derivation to above ones, because very high (more than 6^{th}) order derivatives of $(\delta/(1+\delta)-\gamma)^2$ have to be computed in this case, while this function is too complex to compute its high order derivatives. Hence we prove (10.48) and (10.49) by plotting $\delta/(1+\delta)-\gamma$ as a function of $\gamma \in [0, 1/2)$ in Figure 10.3.

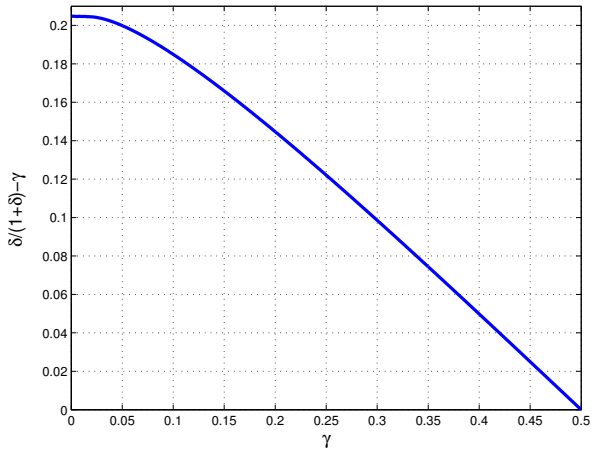

Figure 10.3: Plot of $\delta/(1+\delta)-\gamma$ as a function of $\gamma \in [0, 1/2)$ on 5000 points between 0 and 1/2.

According to the curve of the function $\delta/(1+\delta)-\gamma$ shown in Figure 10.3, the function is always larger than 0 and decreases as γ increasing in interval $\gamma \in [0, 1/2)$. Since $\delta/(1+\delta^2)-\gamma$ is a continues function of γ, there is no spike or singularity on its curve. Therefore, we obtain (10.48) and (10.49). This completes the proof. □

According to Lemma 3 and Lemma 4, we have the following proposition about the probabilities of the 2 types of recovery failures.

Proposition 10. *The probabilities of the 2 types of recovery failures are functions*

of m and γ:

$$\Pr\left(D_{KL}\left(P1\|\hat{P}\right) < D_{KL}\left(P2\|\hat{P}\right)\right) = \sum_{j=1}^{\frac{m\delta}{1+\delta}} \binom{m}{j} \left(\frac{1}{2}\right)^m, b \cap y = \emptyset. \tag{10.61}$$

$$\Pr\left(D_{KL}\left(P1\|\hat{P}\right) > D_{KL}\left(P2\|\hat{P}\right)\right) = \sum_{j=\overline{\frac{m\delta}{1+\delta}}}^{m} \binom{m}{j} \gamma^j (1-\gamma)^{m-j}, b \cap y = b. \tag{10.62}$$

Proof. Lemma 3 gives the distribution of $\hat{P}(j)$ follows binomial models (10.33) and (10.34) respectively under the 2 different facts, i.e., $b \cap y = \emptyset$ in type I failure and $b \cap y = b$ in type II failure. Lemma 4 provides the 2 ranges of j that leads to the 2 kinds of failures in (10.40) and (10.41), respectively. Therefore, the probability of each kind of failure is the sum of all the probabilities $\hat{P}(j)$ with j in the respective range. Thus we have

$$\Pr\left(D_{KL}\left(P1\|\hat{P}\right) < D_{KL}\left(P2\|\hat{P}\right)\right) = \sum_{j=1}^{\frac{m\delta}{1+\delta}} \Pr\left(\hat{P}(j)\right), b \cap y = \emptyset. \tag{10.63}$$

$$\Pr\left(D_{KL}\left(P1\|\hat{P}\right) > D_{KL}\left(P2\|\hat{P}\right)\right) = \sum_{j=\overline{\frac{m\delta}{1+\delta}}}^{m} \Pr\left(\hat{P}(j)\right), b \cap y = b. \tag{10.64}$$

Substitute $\Pr\left(\hat{P}(j)\right)$ given in Lemma 3 into the above probabilities, we obtain (10.61) and (10.62) in Proposition 10. This completes the proof. □

Since $\hat{P}(j)$ follows binomial distribution and the probabilities of the 2 kinds of failures are CDFs of binomial distributions, we apply Hoeffding's inequality to the 2 probabilities in Proposition 10 and obtain the upper bounds for them.

Theorem 21. ***(Probabilistic bounds for recovery failures)*** *The upper*

bounds for the probabilities of the two types of recovery failures are:

$$\Pr\left(D_{KL}\left(P1\|\hat{P}\right) < D_{KL}\left(P2\|\hat{P}\right)\right) \leq \frac{1}{2}\exp\left[-2\left(\frac{\delta}{1+\delta}-\frac{1}{2}\right)^2 m\right], b \cap y = \emptyset. \tag{10.65}$$

$$\Pr\left(D_{KL}\left(P1\|\hat{P}\right) > D_{KL}\left(P2\|\hat{P}\right)\right) \leq \frac{1}{2}\exp\left[-2\left(\frac{\delta}{1+\delta}-\gamma\right)^2 m\right], b \cap y = b. \tag{10.66}$$

Proof. For type I failure, its probability (10.61) is a CDF of a binomial distribution (10.33). According to the condition of Hoeffding's inequality

$$\frac{m\delta}{1+\delta} \leq \frac{m\delta}{1+\delta} < \frac{1}{2}\cdot m, \tag{10.67}$$

we apply Hoeffding's inequality to (10.61) and obtain

$$\begin{aligned}\Pr\left(D_{KL}\left(P1\|\hat{P}\right) < D_{KL}\left(P2\|\hat{P}\right)\right) &= \sum_{j=1}^{\frac{m\delta}{1+\delta}} \binom{m}{j}\left(\frac{1}{2}\right)^m \\ &\leq \frac{1}{2}\exp\left[-2\frac{\left(\frac{m\delta}{1+\delta}-\frac{m}{2}\right)^2}{m}\right] \\ &\leq \frac{1}{2}\exp\left[-2\left(\frac{\delta}{1+\delta}-\frac{1}{2}\right)^2 m\right], b \cap y = \emptyset.\end{aligned} \tag{10.68}$$

The second inequality in the above derivation is due to (10.47) in Lemma 5, which yields

$$\frac{m\delta}{1+\delta} - \frac{m}{2} \leq \left(\frac{\delta}{1+\delta}-\frac{1}{2}\right) m < 0. \tag{10.69}$$

This completes the proof of (10.65).

For type II failure, its probability (10.62) can be written as a CDF of a

binomial distribution related to (10.34):

$$\begin{aligned}\Pr\left(D_{KL}\left(P1\|\hat{P}\right) > D_{KL}\left(P2\|\hat{P}\right)\right) &= \sum_{j=\overline{\frac{m\delta}{1+\delta}}}^{m} \binom{m}{j} \gamma^j (1-\gamma)^{m-j} \\ &= \sum_{j=0}^{m-\overline{\frac{m\delta}{1+\delta}}} \binom{m}{j} (1-\gamma)^j \gamma^{m-j}. \end{aligned} \tag{10.70}$$

The inequality (10.48) in Lemma 5 yields the condition of Hoeffding's inequality.

$$m - \overline{\frac{m\delta}{1+\delta}} \leq m - \frac{m\delta}{1+\delta} < (1-\gamma) \cdot m. \tag{10.71}$$

We apply Hoeffding's inequality to (10.70) and obtain

$$\begin{aligned}\Pr\left(D_{KL}\left(P1\|\hat{P}\right) > D_{KL}\left(P2\|\hat{P}\right)\right) &\leq \frac{1}{2}\exp\left[-2\frac{\left[\left(m - \overline{\frac{m\delta}{1+\delta}}\right) - m(1-\gamma)\right]^2}{m}\right] \\ &\leq \frac{1}{2}\exp\left[-2\left(\frac{\delta}{1+\delta} - \gamma\right)^2 m\right], b \cap y = b. \end{aligned} \tag{10.72}$$

The second inequality in the above derivation is due to (10.48) in Lemma 5, which yields

$$\left(m - \overline{\frac{m\delta}{1+\delta}}\right) - m(1-\gamma) = m\gamma - \overline{\frac{m\delta}{1+\delta}} \leq \left(\gamma - \frac{\delta}{1+\delta}\right) m < 0. \tag{10.73}$$

This completes the proof of (10.66). □

The probabilistic bounds given in Theorem 21 are exponential functions of m and γ, wherein m is the number of measurements (number of random projection signs in the recovery algorithm, i.e., dimension of CL label vector in CL), and $\gamma \in [0, 1/2)$ is a monotonically decreasing function of card(b) according to (10.36). As a compression-recovery algorithm, it is essential to investigate how many measurements are sufficient to ensure the success of the recovery in CL. As a multi-label prediction algorithm, it is essential to analyze whether the applica-

tion of distilled labelsets (DLs) can improve the prediction performance. These two significant questions can be well answered by Theorem 22 that analyzes how the probabilistic bounds of recovery failures change with m and γ.

Theorem 22. *1) The upper bounds of the probabilities for the 2 types of recovery failures exponentially shrink with the increasing of measurements' number m. 2) The upper bounds of the probabilities for the 2 types of recovery failures exponentially shrink with the increasing of the distilled labelset b's cardinality.*

Proof. 1) In Theorem 21, the two upper bounds

$$\frac{1}{2}\exp\left[-2\left(\frac{\delta}{1+\delta}-\frac{1}{2}\right)^2 m\right] \quad \text{and} \quad \frac{1}{2}\exp\left[-2\left(\frac{\delta}{1+\delta}-\gamma\right)^2 m\right]. \tag{10.74}$$

They are both exponentials of negative linear functions of m. Therefore, they exponentially decreases with the increase in the number of measurements m. This completes the proof of conclusion 1.

2) We study the monotonicities of the 2 functions of γ in the 2 upper bounds from Theorem 21:

$$f_1(\gamma) = \left(\frac{\delta}{1+\delta}-\frac{1}{2}\right)^2 \quad \text{and} \quad f_2(\gamma)\left(\frac{\delta}{1+\delta}-\gamma\right)^2. \tag{10.75}$$

Their partial derivatives with respect to γ are

$$\frac{\partial f_1}{\partial \gamma} = 2\left(\frac{\delta}{1+\delta}-\frac{1}{2}\right)\cdot\frac{\partial \frac{\delta}{1+\delta}}{\partial \gamma}, \tag{10.76}$$

$$\frac{\partial f_2}{\partial \gamma} = 2\left(\frac{\delta}{1+\delta}-\gamma\right)\cdot\frac{\partial\left(\frac{\delta}{1+\delta}-\gamma\right)}{\partial \gamma}. \tag{10.77}$$

By using inequalities (10.47), (10.45), (10.48), (10.49) and (10.59), we have

$$\frac{\partial f_1}{\partial \gamma} < 0 \quad \text{and} \quad \frac{\partial f_2}{\partial \gamma} < 0. \tag{10.78}$$

Thus f_1 and f_2 increase with the decreasing of γ. Since both f_1 and f_2 are

nonnegative, the 2 upper bounds

$$\frac{1}{2}\exp\left[-2f_1(\gamma)m\right] \quad \text{and} \quad \frac{1}{2}\exp\left[-2f_2(\gamma)m\right] \tag{10.79}$$

exponentially decrease with the decreasing of γ.

According to (10.36), $\gamma \in [0, 1/2)$ is a monotonically decreasing function of $\text{card}(b)$, so the 2 upper bounds in Theorem 21 exponentially decreases with the increase in the cardinality of the DL b. This completes the proof of conclusion 2. □

Theorem 22 shows that the upper bounds for the probabilities of the 2 types of failures both the recovery accuracy exponentially shrink with the increasing of either the measurements or the cardinality of DL. It indicates that the the multi-label prediction performance of CL will not be harmed by the label compression and will be significantly improved by applying LDM. In summary, Theorem 22 This theoretically shows CL saves the time costs and balances the training data, and LDM explores the label correlations. When DL is replaced with unit vectors whose elements are 0 except one element is 1, it is easy to derive from Theorem 22 that $m = \mathcal{O}(\log k)$ can produce a sufficiently accurate recovery. Note that DL can produce more robust and accurate recovery than the unit vectors (cf. (10.27) and the analysis below it), though the value of m ensuring accurate recovery cannot be precisely identified in this case.

10.4 Discussion

In this section, we discuss CL's contributions to multi-label learning, and analyze CL's relationships with compressed sensing (CS) and error-correcting output codes (ECOC).

10.4.1 Contributions to multi-label learning

In essence, CL is a label transformation method for multi-label learning. It solves or at least substantially alleviates the three aforementioned problems harassing

multi-label learning field via label random projection based compression and recovery. In particular, the label compression in CL generates a new label matrix with improved sample balance for each label, mutual independence between different labels and lower label dimensionality than the original one. According to the analyses in Section 10.2, these are attributed to a series of properties of random projection signs. The sample balance and mutual independence of CL labels remove the two obstacles of applying conventional binary classification methods to multi-label learning problems. Thus the label compression method in CL avoids the problems of directly applying single-label learning methods to multi-label learning tasks, while inherits their advantages in binary classification tasks. Besides, the single-label learning methods are directly invoked in CL without any modification. For example, the SVM based CL presented in this chapter retains the optimality and robustness of margin maximization, and can be extended to nonlinear kernel space. The dimension reduction of the label matrix significantly decreases the problem size and yields an efficient training stage. These characteristics of CL improves both effectiveness and efficiency of multi-label learning.

Although the three problems have been more or less considered by existing methods, they have rarely been simultaneously considered without introducing other problems. For instance, methods for exploiting the label dependence usually expand the problem size and aggravate the sample imbalance. Methods for label dimension reduction often transform the classification problem to other problems that ignore the label dependence and have complex label recovery algorithms.

Moreover, CL is a general method whose training process is isolated from the label compression and recovery, and thus various existing single-label and multi-label learning methods can be directly invoked in the training stage of CL by using the compressed label matrix. The benefits of these methods can be completely retained in their CL variants. Hence CL is not only a multi-label classification solver but also a general method that can be directly applied to most existing multi-label leaning techniques for improving their performance and speed. LDM in CL is a greedy binary matrix decomposition technique that exploits the discrete patterns of a given binary matrix. Thus LDM can be isolated from the CL scheme and applied independently to multi-label learning in order to obtain better labelsets for LP [216] or build the nodes in tree-structural hierarchy

[20][217].

Furthermore, CL significantly decreases the problem size of multi-label learning by adding a label compression and recovery procedure. However, the computation of the additional procedure is simple and fast. In particular, the compression is composed of random projections and hard thresholding, and the recovery includes only comparison, thresholding and sorting. Thus the additional time cost brought by compression is negligible comparing with the tremendous reduced time cost caused by CL. Compared with the multi-label learning via CS [122] whose recovery needs to solve an ℓ_1 minimization, CL targets on a more powerful classification model and develops a more efficient recovery algorithm by exploiting the $\{0, 1\}$ nature and discrete pattern of the label matrix. Therefore, CL brings significant acceleration to multi-label learning and makes the large-scale problems [141] computationally tractable.

10.4.2 Relationship with compressed sensing

CS [71][40] is a sparse signal compression and recovery scheme that achieves remarkable success in recent years. It proves that a sparse signal x can be exactly recovered from a small number of its random projections $y = xA$ if the projection matrix A follows the Restricted Isometry Property (RIP). Similarly, CL proposes a $\{0, 1\}$ label matrix compression and recovery scheme. Thus it is interesting to discuss the relationship between CS and CL.

1 CS and CL both use random projections in their compression of a sparse signal and a $\{0, 1\}$ label matrix, respectively. Random projection can be deemed as a pseudorandom generator that encodes certain information of the original data into random variables. In CS, random projections provide linear random measurements that satisfy RIP, which guarantees the success of exact recovery of sparse signal x from a few of its random measurements y via ℓ_1 minimization. In CL, random projection signs of two binary vectors have an explicit joint distribution, so a row of $\{0, 1\}$ label matrix can be exactly recovered from its random projection signs by testing the joint distributions of the random projection signs of the row and several given $\{0, 1\}$ vectors, i.e., distilled labelsets. Note that the measurements of CS

are real-valued while the measurements (i.e., CL labels) are 1-bit. From the viewpoint of transmission, the space costs of CL compression is significantly reduced comparing with the CS compression.

2 CS and CL develop different recovery algorithms by exploiting the different properties of sparse signals and the $\{0, 1\}$ label matrix, respectively. In CS, the sparsity leads to the minimization of ℓ_1 norm of the unknown signal in the recovery algorithm, because ℓ_1 norm is a convex relaxation of the signal cardinality. In CL, although the rows of the $\{0, 1\}$ label matrix satisfy the sparse assumption as well, they have more specific characteristics. In particular, 1) each entry is either 0 or 1 in value; and 2) there exist discrete patterns that are frequently shared by the rows of the label matrix. These two characteristics of the $\{0, 1\}$ label matrix inspire the KL divergence based hypothesis tests in the recovery algorithm of CL.

A more related CS problem to the CL problem is 1-bit compressed sensing [106]. Different from CS which recovers the sparse signal from a few of its random projections, 1-bit CS aims to recover the support set of the sparse signal x from a few of its random projection signs $y = \text{sign}(xA)$. In the passive algorithm of 1-bit CS, the expectation of $y_i \text{sign}(A_{ji})$ over different i can be explicitly computed in a similar spirit of the geometric inference in Theorem 18. A hypothesis test of the expectation is then conducted to determine whether j is included in the support set of x.

The similarity between 1-bit CS and CL is as follows: the geometric inference of the expectation in 1-bit CS and the geometric inference of the joint distribution in CL both study the properties of random projection on a vector drawn uniformly on a hypersphere.

However, 1-bit CS and CL are different in the following three aspects:

1 Their problems are different. 1-bit CS targets on recovering the support set of a sparse signal, while CL aims at recovering a $\{0, 1\}$ label matrix. Recovering the support set in 1-bit CS only cares about the positions of the nonzero entries in a single sparse vector, while recovering $\{0, 1\}$ label matrix in CL explores the frequent $\{0, 1\}$ patterns appearing in all the rows of the matrix via LDM.

2 1-bit CS studies the random variable $y_i \text{sign}(A_{ji})$, which can be explained as the product of random projection signs of sparse signal x and a unit vector e with 1 on the j^{th} entry and 0 otherwise. CL studies the random projection signs of $\{0, 1\}$ label vector y and a given distilled labelsets D_i.

3 1-bit CS recovers the support set through tests of the expectation of two random projection signs' product, while CL recovers the $\{0, 1\}$ label matrix via tests of the joint distribution of two random projection signs based on KL divergence comparison.

Although 1-bit CS and CL are two different techniques for two different problems, it is possible to extend the methods used in CL to 1-bit CS for improving its performance and recovery bound. In particular, the expectation of $y_i \text{sign}(A_{ji})$ used in the 1-bit CS recovery can be replaced by $y_i \text{sign}(zA_i)$, wherein z is a vector revealing the structure information of the original signal x. This will lead to a "structured 1-bit CS". Another modification from CL is to adopt a distribution test based on KL divergence rather than an expectation test in the recovery of 1-bit CS, because the distribution includes more information about x than its expectation.

10.4.3 Relationship with error-correcting output codes

Error-correcting output codes (ECOC) [66][79] transforms multi-class problem to several binary problems. ECOC consists of two stages, 1) coding stage that encodes each class label into a $\{-1, +1\}$ or $\{-1, 0, +1\}$ codeword, and 2) decoding stage that seeks for the class codeword closest to the predicted codeword of a test sample. Each codeword of the training sample is a $\{-1, +1\}$ or $\{-1, 0, +1\}$ vector of dimension d that augments with the increasing of the class numbers k. In the training step, the codewords of training samples are deemed as their new label vectors, and d binary classifiers are learned from the training set to predict the d dimensions of the codeword of a given sample, respectively. It has been shown that the error-correcting properties of the decoding stage are helpful to reduce the bias and variance of the learning algorithm. Please refer to [79] for a complete review of ECOC methods.

ECOC is related to CL for the following two reasons:

1 In label compression via random projections, CL transforms the original label matrix into a new one, i.e., the CL label matrix. This compression step is similar to the coding stage of ECOC, which assigns each class a new label vector called codeword. Both CL labels and codewords in ECOC are used as the labels of the training samples when training the subsequent binary classifiers.

2 In the recovery by KL-divergence test, CL recovers the original label vector of a test sample from its CL label vector predicted by the binary classifiers. This recovery step is similar to the decoding stage of ECOC, which finds the class codeword closest to the predicted codeword of a test sample and assigns the corresponding class label to the sample. The final prediction results of CL and ECOC are inferred from the predicted CL label vector and ECOC codeword, respectively.

Although CL has a coding stage and a decoding stage similar to the two stages in ECOC, these two schemes are essentially different on their targeted problems, coding and decoding algorithms, and methodologies. Details are given below.

1 ECOC is designed for a multi-class problem, while CL is designed for a multi-label learning problem. Although a multi-class problem can be viewed as a special case of multi-label learning, their discrepancy leads to the key differences in developing ECOC and CL. For example, there are only k possible codewords for a multi-class problem in ECOC, so the training set can include all the k codewords for most datasets. Thus these k codewords can be independently generated in the coding stage, and treated isolatedly in the training and decoding stages. However, there are at most $2^k - 1$ possible label vectors (exponentially increased with k) for a multi-label learning problem in CL, so the training sets in most datasets cannot include all the possible CL label vectors. This fact indicates the label vector that needs to be predicted could never appear in the training set before. For this reason, the correlation between different label vectors and the dependence between different labels play significantly roles in multi-label learning and

provide critical information for prediction. Thus the CL label vectors cannot be generated independently in compression. This is why we apply the same random matrix A to different label vectors in CL. This is also why we develop LDM to extract the label correlations, preserve them in DL and use DL in the CL recovery. Another difference caused by the problem discrepancy is the decoding stage. It is possible to search for the closest codeword among the k possible ones in ECOC. But it is impossible to search for the most accordable label vector among the 2^k-1 possible ones in CL, at least from the aspect of computation. Thus we have to develop an accurate recovery algorithm for finding the real label vector in CL.

2 The coding stage in ECOC and the label compression in CL are different in the dimension of a codeword (the number of CL labels) and the correlation preservation. Common coding strategies in ECOC include one-versus-all, one-versus-one, randomized design such as dense random and sparse random, and problem-dependent design such as DECOC, Forest-ECOC and ECOC-ONE. One-versus-all and one versus-one suffer from the problem of serious imbalance in the coding matrix whose rows are composed of codewords. Moreover, one-versus-one needs to train $k(k-1)/2$ binary classifiers in the subsequent training step, which is computationally intractable for most datasets. Randomized design generates the codeword for each class randomly and independently. The burden brought by randomness is that high dimensionality of the codeword is required to preserve sufficient information of the data. Problem-dependent design extracts the codewords by mining the structure of the k classes. This method requires codewords of dimension at least $k-1$. However, the imbalance problem of the coding matrix is ignored in this method. In summary, the dimensionality of codewords in ECOC is much higher than or at least around k in most strategies to guarantee successful coding, and the correlations between different classes can be abandoned after coding. However, CL can compress the k-dimensional label vector to $m = \mathcal{O}(\log k)$ (cf. the last paragraph of Section 10.3.4) in order to solve the problem of label high dimensionality. Moreover, the distribution and pairwise distance of the original label vectors are main-

tained after compression in CL (cf. the last paragraph of Section 10.2.3). The preserved graph structure is helpful for training classifiers on the CL labels. Note CL adopts random projection signs as the compression of the original labels, which has not been used in the coding method of ECOC.

3 The decoding stage of ECOC and the KL divergence test based recovery in CL are different in algorithm development. The decoding stage in ECOC is based on the error-correcting principles and finds the closest codeword among the k possible ones, while the recovery of CL precisely reconstructs the original label vector dimension-wisely or DL-wisely (by testing whether each DL belongs to the original label vector or not). The decoding algorithm of ECOC can be grouped into three types, i.e., comparison of distances between the predicted codeword and existing ones, class membership possibility estimation and pattern space transformation. But the recovery of CL is based on a series of KL divergence tests on DL.

In ML-CS [122], the regret transform in SECOC [146] (one method of ECOC) provides a theoretical guarantee similar in formulation to RIP in compressed sensing. Therefore, to some extent, ML-CS [122] can be deemed as an extension of ECOC in multi-label learning. Compared with this method, CL performs promisingly on three aspects: 1) less measurements, ML-CS needs $m = \mathcal{O}(\log k)$ real-valued labels, while CL requires only $m = \mathcal{O}(\log k)$ 0-1 (1-bit) labels; 2) faster recovery, ML-CS invokes existing convex optimization based CS recovery algorithms, while CL develops a non-iterative recovery algorithm with linear time; 3) ML-CS transforms a 0-1 prediction problem to a regression problem, while CL transforms a large 0-1 prediction problem to a smaller one. Thus CL does not change the nature of the problem; 4) the associated improvement on prediction performance by exploring label correlation in ML-CS cannot be analyzed, while CL thoroughly explores the label correlation via developing LDM and applying DL in recovery. The benefits of applying DL can be clearly analysed (cf. the last two paragraphs in Section 10.3.3).

In addition, CL can be applied to a multi-class problem without the application of LDM and DL. Comparing with the existing ECOC methods, CL brings the following advantages for multi-class problems: 1) smaller dimensionality of

the codewords; 2) randomized coding that can simultaneously preserve class correlations and eliminate imbalance in the coding matrix; and 3) more robust and faster decoding based on an accurate recovery. Therefore, CL on unit vectors can be deemed as a powerful and novel ECOC method for multi-class problems.

10.5 Experiments

In this section, we evaluate CL via 5 groups of experiments on 21 datasets obtained from real-world problems including text classification, image annotation, scene classification, music categorization, genomics and web page classification. In the first group of experiments, we evaluate the label compression and recovery in CL by measuring the sample balance, mutual independence of CL label matrix Z and the recovery error rate of Algorithm 13. In the second group of experiments, we compare CL with BR in multi-label prediction when SVM is adopted as the binary classifier in both methods. Time cost and prediction performance are evaluated on different C parameters of SVM for a comparison of robustness. In the third group of experiments, we compare CL with 3 popular multi-label learning methods, i.e., ML-knn [246], MDDM [253] and multi-label prediction via compressed sensing (ML-CS) [122]. ML-knn and MDDM are extensions of knn and dimension reduction [202][105] for multi-label learning, respectively. ML-CS is related to CL because it adopts a compressed label embedding. In the fourth group of experiments, we compare CL with 2 SVM methods dealing with imbalanced data, i.e., SVM-SMOTE [45] and SVM-WEIGHT [184]. This group of experiments separates the performance improvements caused by label correlation and sample balance in CL. In the fifth group of experiments, we study the trade-off between time cost reduced by label compression and prediction performance enhanced by sample balance in CL. In order to eliminate the influence of label correlation, DL is replaced by the dictionary composed of unit vectors. In the last 4 groups of experiments, the performance of multi-label prediction is evaluated in terms of 5 metrics, i.e., Hamming loss, precision, recall, F1 score and accuracy, and CPU seconds for time cost contest. All the experiments are implemented and run in MatLab on a server with dual quad-core 3.33 GHz Intel Xeon processors and 32 GB RAM.

10.5.1 Evaluation metrics

In the experiments of label compression and recovery, three metrics are used to measure the sample balance, label independence and recovery error rate, respectively. Given a $\{0,1\}$ label matrix $Y \in \{0,1\}^{n\times k}$, its balance-of-degree is defined as the average proportion of positive samples on all the different labels in (10.2). A degree-of-balance close to 0.5 indicates a good sample balance in Y.

The label independence in Y is measured by the average χ^2 score after Yates' correction [237] over all the label pairs in Y. In particular, for a label pair $\{i,j\}$, assume

$$a = |p : Y_{pi} = 1, Y_{pj} = 1|, b = |p : Y_{pi} = -1, Y_{pj} = 1|, \tag{10.80}$$

$$c = |p : Y_{pi} = 1, Y_{pj} = -1|, d = |p : Y_{pi} = -1, Y_{pj} = -1|. \tag{10.81}$$

The χ^2 score for label pair $\{i,j\}$ is defined as:

$$\chi^2_{ij} = \frac{n\left(\|ad - bc\|_1 - n/2\right)^2}{(a+b)(c+d)(b+d)(a+c)}. \tag{10.82}$$

The average χ^2 score is:

$$\chi^2 = \text{mean}\left(\chi^2_{ij}\right). \tag{10.83}$$

A large χ^2 score indicates a strong mutual independence between labels.

Given two label matrices $Y1, Y2 \in \{0,1\}^{n\times k}$, wherein $Y1$ is the real one an $Y2$ is the recovered one, the recovery error rate in CL is measured by the Hamming loss:

$$HamLoss = \frac{1}{nk}\sum_{i=1}^{n}\sum_{j=1}^{k} Y1_{ij} \oplus Y2_{ij}, \tag{10.84}$$

where $\oplus$ is the exclusive disjunction, i.e., the XOR operation.

In the experiments of multi-label prediction, five metrics, i.e., Hamming loss, precision, recall, F1 score and accuracy, are used to measure the prediction performance. Hamming loss is defined in (10.84). The other four metrics are defined

as:

$$Precision = \frac{1}{n}\sum_{i=1}^{n}\frac{\text{card}\left(Y1_i \cap Y2_i\right)}{\text{card}\left(Y2_i\right)}, \tag{10.85}$$

$$Recall = \frac{1}{n}\sum_{i=1}^{n}\frac{\text{card}\left(Y1_i \cap Y2_i\right)}{\text{card}\left(Y1_i\right)}, \tag{10.86}$$

$$F1 = \frac{1}{n}\sum_{i=1}^{n}\frac{2\text{card}\left(Y1_i \cap Y2_i\right)}{\text{card}\left(Y1_i\right) + \text{card}\left(Y2_i\right)}, \tag{10.87}$$

$$Accuracy = \frac{1}{n}\sum_{i=1}^{n}\frac{\text{card}\left(Y1_i \cap Y2_i\right)}{\text{card}\left(Y1_i \cup Y2_i\right)}. \tag{10.88}$$

These five metrics have been broadly applied on general binary data. However, their importances differs when used in multi-label prediction evaluation, because there are much more 1s than 0s in the label matrix. Hamming loss could be very small when the labels of all the test samples are predicted as 0, and thus it cannot indicates a successful prediction in this case. Precision and recall should be considered together, because high precision always accompanies low recall when most positive samples are falsely predicted as positive. F1-score and accuracy are less sensitive to the imbalance of label matrix. Therefore, the evaluation of prediction performance should be a integrative consideration of all the five metrics with rough importances according to $F1-score, Accuracy > Precision, Recall > Hammingloss$.

10.5.2 Datasets

We evaluate the performance of label compression and recovery, and multi-label prediction of CL on 21 datasets from different domains and of different scales, including Bibtex [136], Corel5k [76], Mediamill [203], IMDB [194], Enron [221], Genbase [69], Medical [221], Emotions [213], Scene [26], Slashdot [194] and 11 sub datasets included in Yahoo dataset [223]. These datasets are collected from different practical problems such as text classification, image annotation, scene classification, music categorization, genomics and web page classification. Table 10.1 shows the number of samples n, number of features p, number of labels k

and number of unique labelsets K, the average cardinality of all label vectors $Card$, and the average nonzero entry proportion of all label vectors $Density$ of different datasets.

ID	Datasets	Domain	n	p	k	K	Card	Density
1	Bibtex	text	7395	1836	159	2856	2.402	0.015
2	Corel5k	image	5000	499	374	3175	3.522	0.009
3	Mediamill	video	43907	120	101	6555	4.376	0.043
4	IMDB	text	120919	1001	28	4503	1.9997	0.0714
5	Enron	text	1702	1001	53	753	3.378	0.064
6	Genbase	genomics	662	1186	27	32	1.252	0.046
7	Medical	text	978	1449	45	94	1.245	0.028
8	Emotions	music	593	72	6	27	1.869	0.311
9	Scene	scene	2407	294	6	15	1.074	0.179
10	Slashdot	text	3782	1079	22	156	1.1809	0.0537
11	Yahoo-Arts	web	5000	462	26	462	1.6360	0.0629
12	Yahoo-Business	web	5000	438	30	161	1.5878	0.0529
13	Yahoo-Computers	web	5000	681	33	253	1.5082	0.0457
14	Yahoo-Education	web	5000	550	33	308	1.4606	0.0443
15	Yahoo-Entertainment	web	5000	640	21	232	1.4204	0.0676
16	Yahoo-Health	web	5000	612	32	257	1.6622	0.0519
17	Yahoo-Recreation	web	5000	606	22	322	1.4232	0.0647
18	Yahoo-Reference	web	5000	793	33	217	1.1694	0.0354
19	Yahoo-Science	web	5000	743	40	398	1.4506	0.0363
20	Yahoo-Social	web	5000	1047	39	226	1.2834	0.0329
21	Yahoo-Society	web	5000	636	27	582	1.6920	0.0627

Table 10.1: Information of datasets that are used in label compression and recovery experiments and multi-label prediction experiments. In the table, n refers to the number of samples, p refers to the number of features, k refers to the number of labels, K refers to the number of unique label vectors, "Card" refers to the average cardinality of all label vectors, "Density" refers to the average nonzero entry proposition of all label vectors.

10.5.3 Label compression and recovery

In this group of experiments, we test the sample balance and label independence on the compressed CL labels, and the recovery accuracy of the corresponding

recovery algorithm given in Algorithm 13. We only test the label compression and recovery methods proposed in CL in this group of experiments, and leave the classification and prediction to the second group of experiments. That is because the classification error in CL is caused by two issues, i.e., the label recovery error introduced by compression and the classification error on CL labels. Since the former one is determined by the proposed CL, while the latter one is mainly determined by the binary classification method CL invokes, independent evaluations of different errors are important to the analysis of the effectiveness of CL.

For each of the datasets listed in Table 10.1, we compress the label matrix to its random projection signs and extract its distilled labelsets by using DL algorithm given in Algorithm 12, and then the recovery algorithm of CL given in Algorithm 13 is invoked to recover the original labels from their CL labels. We measure the degree-of-balance and the χ^2 score of CL labels, and calculate the Hamming loss between the original labels and the recovered ones on different compression ratios. For the two datasets with 6 labels, i.e., Scene and Emotions, the compression ratio is settled from 0 to 3, because the empirical joint distribution estimation will become unstable when the number of random projection signs is too small. For the other datasets, the compression ratio changes from 0 to 1. There are two parameters, i.e., the threshold τ in DL algorithm and the cardinality range in the recovery algorithm. We select $0.01 \leq \tau \leq 0.25$ in all the experiments. In particular, a large τ is recommended for datasets with a large number of unique labelsets, because a small τ often generates clusters without shared label subset in this case. The cardinality range is chosen as the cardinality range of the rows in the original label matrix.

We show the experimental results of label compression and recovery on multiple datasets in Figure 10.4 to Figure 10.8.

In the all the figures, the degree-of-balance of CL labels are kept 0.5 ± 0.1 on different compression ratios. For several datasets with large number of labels, e.g., Bibtex and Corel5k, the degree-of-balance stays very close to the ideal value 0.5. Compared to the degree-of-balance of the original labels, which is shown in the figures as well, CL labels highly improve the sample balance.

In the experiments, the χ^2 score is used to measure the label independence,

namely, a larger χ^2 score indicates different labels are more independent. In χ^2 test, when the degree of freedom is 1, a χ^2 score larger than 10.83 associates with a P-value 0.001 and thus indicates the two discrete variables are independent with a very high probability 99.9%. In the figures, most χ^2 scores of the CL labels are much higher than 10.83 and thus the different CL labels are independent. This result is consistent with our theoretical analysis. Compared to the χ^2 score of the original labels, CL labels removes the label dependence, and thus the sequel classification by using conventional binary classification method will be proper and will not downgrade the prediction performance.

The recovery error rates in the figures show that the recoveries are still very accurate with very small Hamming losses when the compression ratio is very small. Thus the computational complexity of CL can be tremendously decreased by using a very low dimensionality, while the learning performance will not be jeopardized. Another observation is that the Hamming loss drops quite fast when the compression ratio is increased. This leads to an efficient compression and precise recovery. Some noises appear on the Hamming loss curve in some figures and makes the curves not monotonically deceasing. That is because the random vectors in A are randomly selected on the hypersphere and thus their distribution on the hypershpere is not absolutely even. However, these noises are of small amplitude and thus will not harm the recovery accuracy.

10.5.4 Multi-label prediction: comparison with BR

In this group of experiments, we compare the multi-label prediction performance and the time cost of CL with BR on 21 datasets with different parameter settings. Since we choose SVM as the single-label learning method in the training stage of CL, BR is compared with CL in the experiments as a baseline method by applying SVM directly on the original labels. This group of experiments aims at verifying the improvement and robustness of prediction performance brought by the training on the CL labels rather than on the original ones.

We use linear SVM in the training stages of both BR and CL. This is because linear SVM has only one parameter C, thus the robustness of the performances can be conveniently compared under different parameter settings.

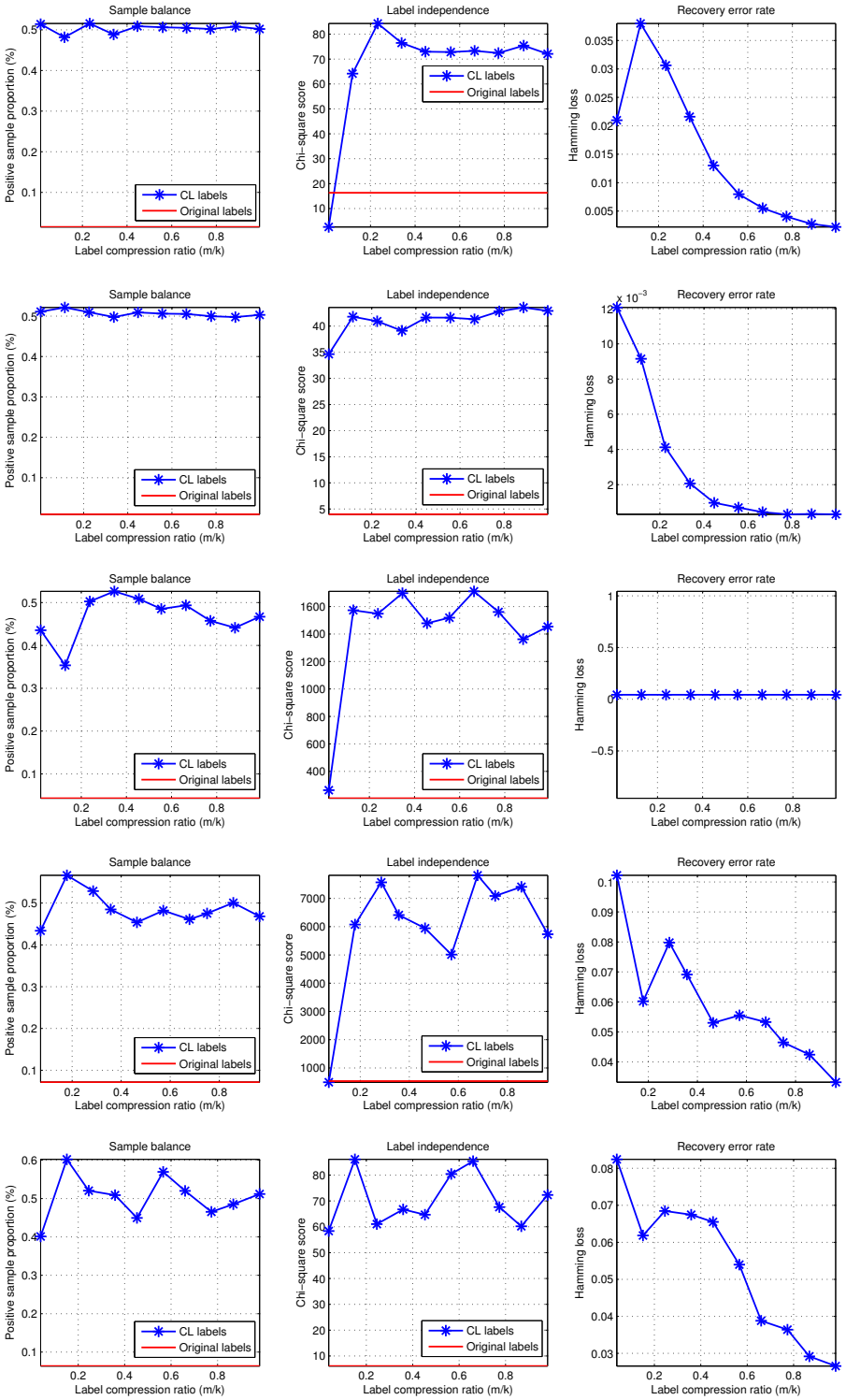

Figure 10.4: Sample balance, label independence and recovery error rate on 21 datasets (1). From top to bottom: Bibtex, Corel5k, Mediamill, IMDB, Enron.

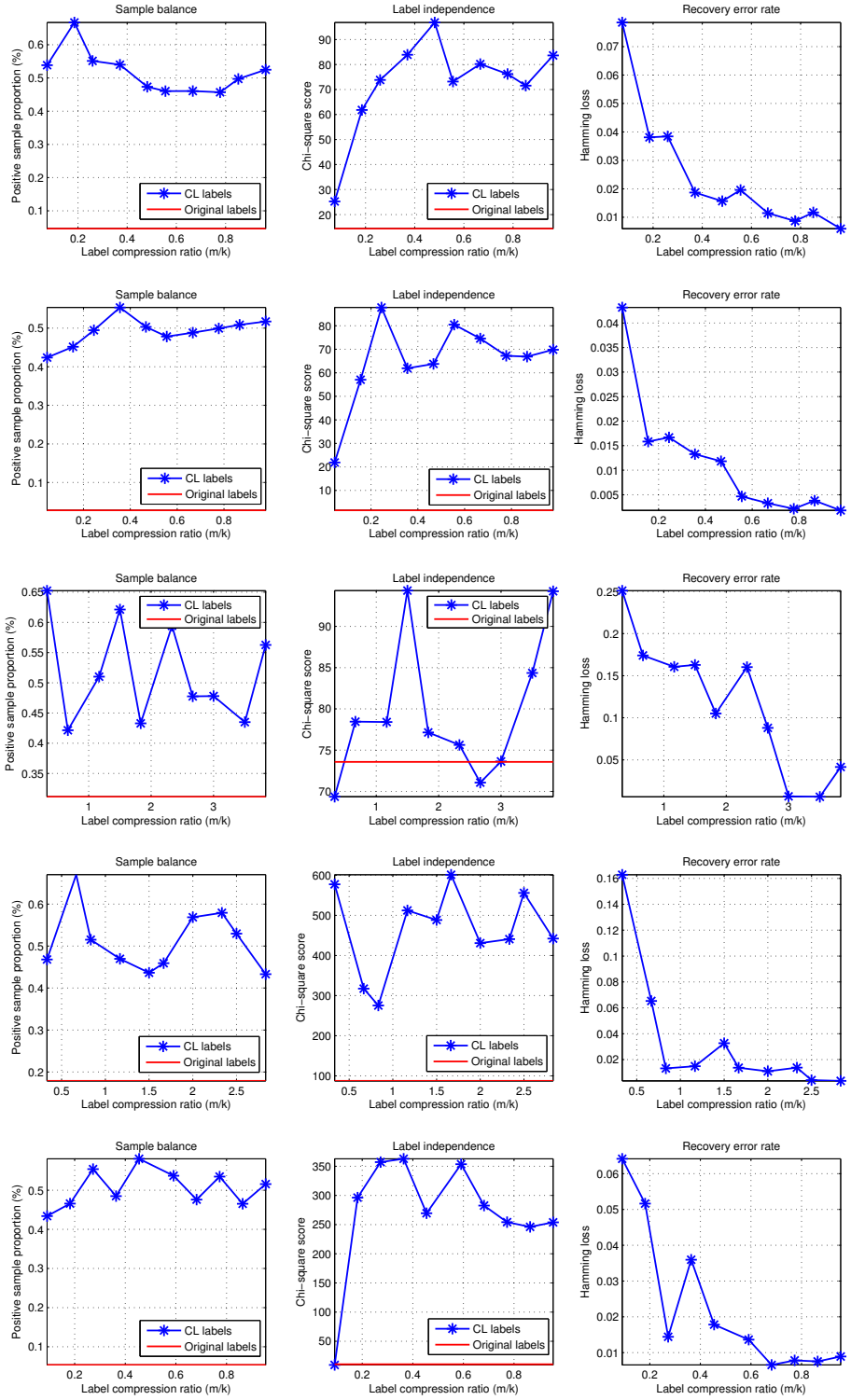

Figure 10.5: Sample balance, label independence and recovery error rate on 21 datasets (2). From top to bottom: Genbase, Medical, Emotions, Scene, Slashdot.

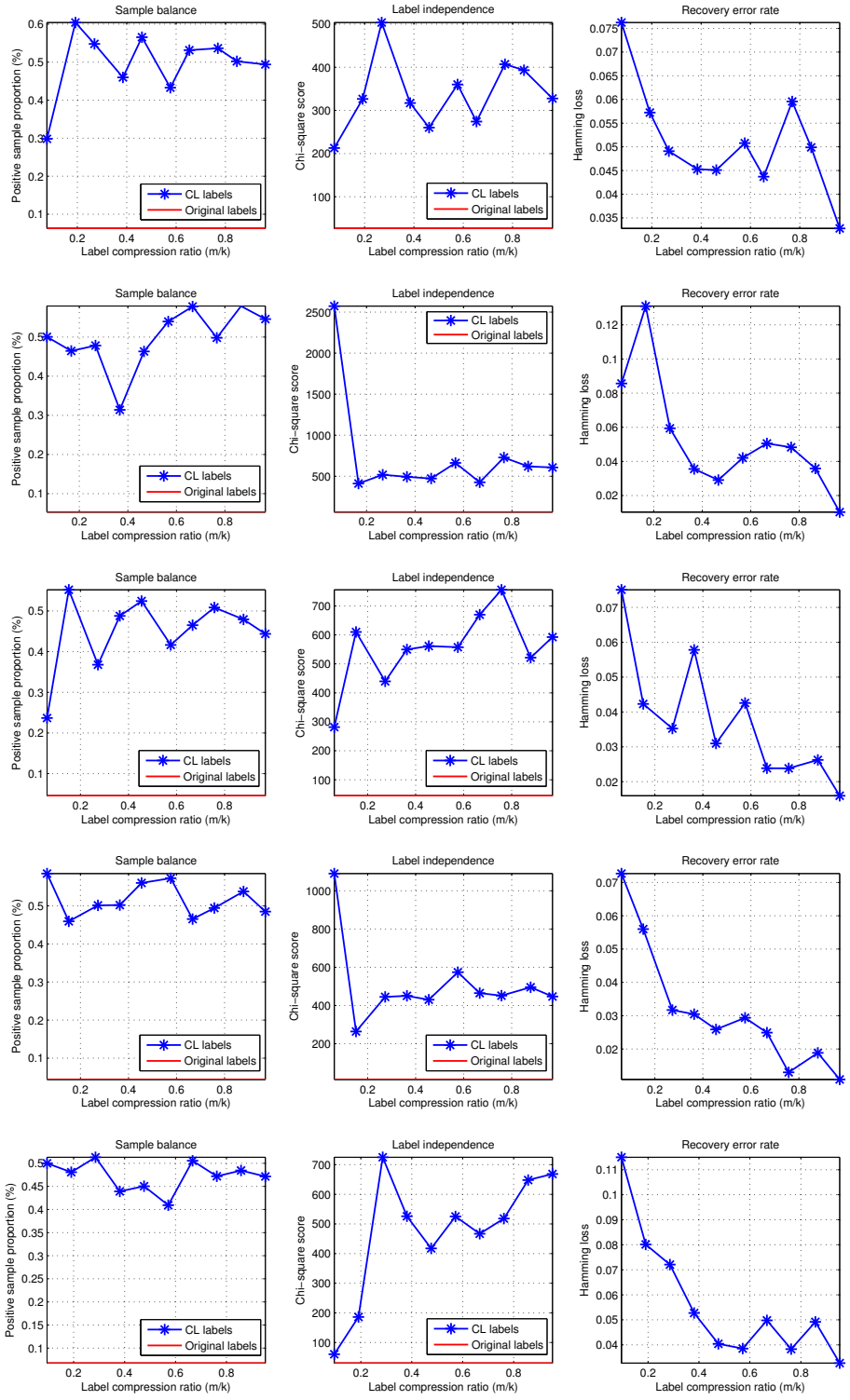

Figure 10.6: Sample balance, label independence and recovery error rate on 21 datasets (3). From top to bottom: Yahoo-Arts, Yahoo-Business, Yahoo-Computers, Yahoo-Education, Yahoo-Entertainment.

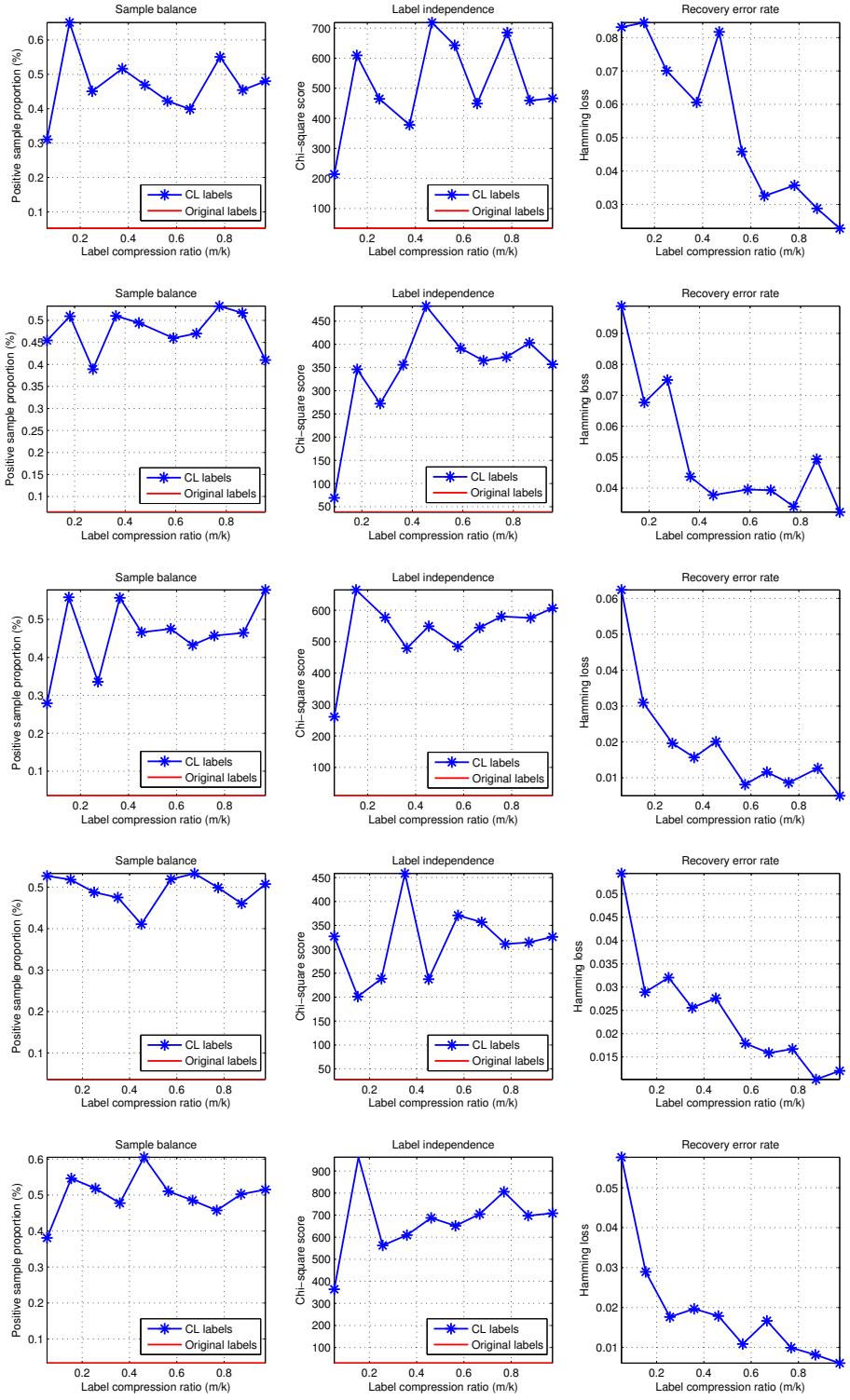

Figure 10.7: Sample balance, label independence and recovery error rate on 21 datasets (4). From top to bottom: Yahoo-Health, Yahoo-Recreation, Yahoo-Reference, Yahoo-Science, Yahoo-Social.

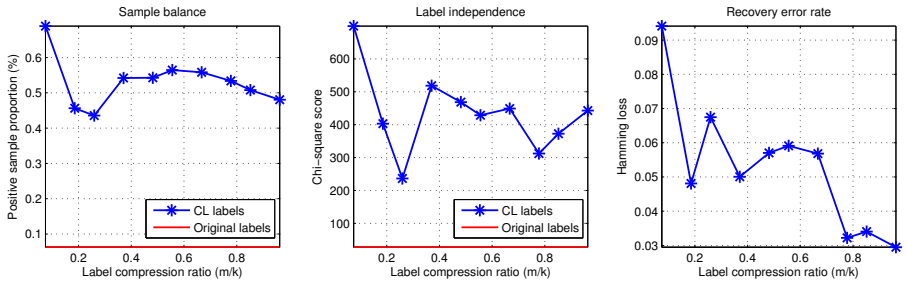

Figure 10.8: Sample balance, label independence and recovery error rate on 21 datasets (5) on Yahoo-Society.

The broadly used standard SVM solver "LIBSVM [44]" is applied to all the datasets except Mediamill and IMDB, which are of large scale and thus intractable for LIBSVM. For these two datasets, we alternatively apply NESVM [270], which is a fast gradient SVM solver for large-scale problems.

Table 10.2 summarizes the information about the training set, the test set and the obtained distilled labelsets of each datasets used in the multi-label prediction experiments. Both training set and test set are randomly selected from the original datasets. The number of distilled labelsets for most datasets is between the number of labels and that of the unique labelsets. When the unique labelsets is of large size, the number of distilled labelsets keeps close to the number of labels. Thus the computational complexity of recovery algorithm will not be significantly increased when the number of unique labelsets is augmented.

In each experiment, the training algorithm given in Algorithm 14 and the prediction algorithm given in Algorithm 15 are applied to training set and test set, respectively. We test the performance on different C in SVM. The parameter C can be interpreted as the weight of hinge loss in the objective function of SVM. The threshold τ and the cardinality range are chosen according to the same method in the label compression and recovery experiments. BR is compared with CL on 3-7 different C values between 10^{-3} to 10^{3} in all experiments. For most datasets, 4 C values $[10^{-3}, 10^{-2}, 10^{-1}, 1]$ are used. Since BR fails on some datasets with these C values when it overwhelmingly predicts all the test samples as negative, we choose different C values in this case to guarantee a thorough comparison on sufficient C values.

ID	Datasets	Training	Test	DL	k	K
1	Bibtex	4880	2515	253	159	2856
2	Corel5k	4500	500	418	374	3175
3	Mediamill	30993	12914	161	101	6555
4	IMDB	62050	58869	61	28	4503
5	Enron	1123	579	73	53	753
6	Genbase	463	199	27	27	32
7	Medical	333	645	46	45	94
8	Emotions	391	202	14	6	27
9	Scene	1211	1196	6	6	15
10	Slashdot	2338	1444	20	22	156
11	Yahoo-Arts	2000	3000	37	26	462
12	Yahoo-Business	2000	3000	35	30	161
13	Yahoo-Computers	2000	3000	39	33	253
14	Yahoo-Education	2000	3000	35	33	308
15	Yahoo-Entertainment	2000	3000	26	21	232
16	Yahoo-Health	2000	3000	35	32	257
17	Yahoo-Recreation	2000	3000	31	22	322
18	Yahoo-Reference	2000	3000	38	33	217
19	Yahoo-Science	2000	3000	47	40	398
20	Yahoo-Social	2000	3000	47	39	226
21	Yahoo-Society	2000	3000	39	27	582

Table 10.2: Training set size, test set size and the obtained distilled labelsets size of each datasets in the multi-label prediction experiments. In order to compare the number of distilled labelsets d with the number of labels k and the number of unique labelsets K, we list k and K of each datasets in the table as well.

The prediction performance is evaluated by five metrics, which are Hamming loss, precision, recall, F1 score and accuracy. The good performance is indicated by a small Hamming loss with the large values of the other four metrics. The evaluation of performance should be an integrative consideration of all the five metrics, because one good metric is possibly associated with poor other metrics and a trivial prediction result. For instance, a small hamming loss can associate with other four metrics that are nearly zeros, when most test samples are predicted as negative in all the binary classifications and the number of positive samples is very small.

	C		HL	Prec	Rec	F1	Acc	Time	Labels
Bibtex	10^3	BR	0.1261	0.0158	0.1227	0.0258	0.0139	4630	159
		CL	0.1070	0.1609	0.5876	0.2049	0.1403	1580	80
	10^2	BR	0.0638	0.0158	0.0422	0.0197	0.0112	5310	159
		CL	0.1032	0.1912	0.6009	0.2315	0.1651	1450	80
	10^1	BR	0.2242	0.0118	0.1417	0.0211	0.0110	5890	159
		CL	0.1075	0.2210	0.5658	0.2445	0.1839	1640	80
	1	BR	-	-	-	-	-	-	159
		CL	0.1509	0.1453	0.4493	0.1618	0.1204	1810	80
Corel5k	10^3	BR	0.0871	0.0252	0.2035	0.0439	0.0231	2240	374
		CL	0.0157	0.2236	0.2365	0.2201	0.1473	911.4	160
	10^2	BR	0.0869	0.0139	0.1027	0.0232	0.0122	1870	374
		CL	0.0153	0.2418	0.2595	0.2392	0.1585	526.5	160
	10^1	BR	0.0311	0.0074	0.0167	0.0102	0.0057	1270	274
		CL	0.0164	0.2051	0.2473	0.2181	0.1397	476.1	160
	1	BR	0.2101	0.0093	0.2011	0.0178	0.0091	1050	374
		CL	0.0159	0.1957	0.2245	0.2058	0.1295	450.9	160
Mediamill	10^3	BR	0.036	0.6913	0.3545	0.4315	0.3308	203.3	101
		CL	0.040	0.5680	0.4630	0.4601	0.3436	77.94	40
	10^2	BR	0.036	0.6913	0.3545	0.4315	0.3308	202.4	101
		CL	0.043	0.5141	0.4635	0.4412	0.3210	72.21	40
	10^1	BR	0.036	0.6913	0.3545	0.4315	0.3308	205.1	101
		CL	0.039	0.5751	0.4629	0.4659	0.3467	76.70	40
	1	BR	0.036	0.6913	0.3545	0.4315	0.3308	204.8	101
		CL	0.063	0.4076	0.5637	0.4191	0.2610	76.03	40

Table 10.3: Multi-label performances and time costs of BR and CL on 21 datasets with different C parameters (1). HL-Hamming Loss, Prec-Precision, Rec-Recall, Acc-Accuracy, Time-CPU seconds, labels-Number of Labels in training stage. "-" denotes the failed experiment that incorrectly predicts all the test samples as negative. The best performances of BR and CL are highlighted with different colors.

	C		HL	Prec	Rec	F1	Acc	Time	Labels
IMDB	10^3	BR	0.0689	0.6338	0.0024	0.0061	0.0024	5352	28
		CL	0.1321	0.2825	0.4567	0.3035	0.2186	3628	20
	10^2	BR	0.0689	0.6338	0.0024	0.0060	0.0024	5318	28
		CL	0.1492	0.2621	0.5178	0.3080	0.2187	3273	20
	10^1	BR	-	-	-	-	-	3529	28
		CL	0.1737	0.2408	0.5048	0.2743	0.1944	1434	20
	1	BR	-	-	-	-	-	2700	28
		CL	0.1971	0.1646	0.5431	0.2254	0.1463	898.2	20
Enron	10^3	BR	0.2283	0.1482	0.5184	0.2183	0.1280	122.8	53
		CL	0.0971	0.3940	0.5080	0.4053	0.2887	36.59	36
	10^2	BR	0.2031	0.1633	0.4360	0.2229	0.1347	168.6	53
		CL	0.0854	0.4549	0.5542	0.4580	0.3370	31.00	36
	10^1	BR	0.0915	0.1307	0.2762	0.2552	0.1718	147.2	53
		CL	0.0779	0.5027	0.5926	0.5076	0.3905	35.60	36
	1	BR	0.0636	0.4957	0.2733	0.3429	0.2324	1050	53
		CL	0.0548	0.3949	0.6019	0.4267	0.2929	37.56	36
	10^{-1}	BR	0.0624	0.5138	0.2851	0.3573	0.2415	77.10	53
		CL	0.1336	0.2708	0.6726	0.3693	0.2398	29.77	36
	10^{-2}	BR	0.1372	0.1742	0.2888	0.2115	0.1250	56.74	53
		CL	0.1020	0.2957	0.5015	0.3510	0.2219	26.11	36
	10^{-3}	BR	0.0624	0.5138	0.2851	0.3573	0.2410	39.93	53
		CL	0.1165	0.2990	0.6482	0.3911	0.2588	15.58	36
Genbase	10^3	BR	0.2051	0.0236	0.1164	0.0387	0.0234	1.0065	32
		CL	0.0106	0.9522	0.9661	0.9527	0.9341	0.4279	20
	10^2	BR	0.0798	0.0377	0.0754	0.0503	0.0377	2.0437	32
		CL	0.0101	0.9605	0.9728	0.9568	0.9441	0.3217	20
	10^1	BR	0.1463	0.0469	0.1214	0.0653	0.0429	1.9739	32
		CL	0.1130	0.4408	0.7278	0.4913	0.4312	0.6293	20
	1	BR	0.3110	0.0521	0.3978	0.0917	0.0516	1.9939	32
		CL	0.1362	0.0921	0.2730	0.1374	0.0915	0.7671	20

Table 10.4: Multi-label performances and time costs of BR and CL on 21 datasets with different C parameters (2).

	C		HL	Prec	Rec	F1	Acc	Time	Labels
Medical	10^3	BR	0.3067	0.0283	0.2693	0.0494	0.0273	4.880	45
		CL	0.0273	0.5904	0.8147	0.6565	0.5694	1.4332	25
	10^2	BR	0.4105	0.0022	0.0261	0.0041	0.0022	4.3789	45
		CL	0.0317	0.5432	0.7858	0.6107	0.5171	1.6326	25
	10^1	BR	0.3990	0.0168	0.2282	0.0311	0.0164	5.8486	45
		CL	0.0567	0.2209	0.3788	0.2683	0.1995	2.0320	25
Emotions	10^3	BR	0.2962	0.5547	0.5380	0.5095	0.4205	0.6797	6
		CL	0.3201	0.5099	0.5666	0.5140	0.4190	1.2970	15
	10^2	BR	0.2970	0.5546	0.5363	0.5082	0.4210	0.3724	6
		CL	0.3408	0.4864	0.5528	0.4946	0.4012	0.6971	15
	10^1	BR	0.2756	0.5785	0.5099	0.5017	0.4274	0.2745	6
		CL	0.3201	0.5208	0.6007	0.5353	0.4373	0.5170	15
	1	BR	0.2649	0.6600	0.4142	0.4538	0.3738	0.2514	6
		CL	0.3350	0.5029	0.5743	0.5134	0.4050	0.4628	15
Scene	10^3	BR	0.1111	0.5263	0.6848	0.6470	0.6141	4.3647	6
		CL	0.0803	0.5957	0.8027	0.6562	0.5794	15.631	15
	10^2	BR	0.0956	0.5502	0.6773	0.6600	0.6377	4.1998	6
		CL	0.0780	0.6080	0.8023	0.6627	0.5887	16.502	15
	10^1	BR	0.0973	0.4877	0.5811	0.5800	0.5661	4.7052	6
		CL	0.0989	0.5850	0.8687	0.6617	0.5674	18.710	15
	1	BR	0.1406	0.2358	0.2341	0.2391	0.2337	5.7473	6
		CL	0.2386	0.4727	0.8031	0.5695	0.4568	20.742	15
	10^{-1}	BR	-	-	-	-	-	-	6
		CL	0.3502	0.2646	0.5059	0.3408	0.2595	26.715	15
Slashdot	10^3	BR	0.2098	0.0692	0.2896	0.1032	0.0651	36.256	22
		CL	0.0878	0.3810	0.6525	0.4600	0.3671	30.864	12
	10^2	BR	0.1314	0.1162	0.2226	0.1377	0.1032	140.46	22
		CL	0.0833	0.4269	0.6514	0.4913	0.4122	27.683	12
	10^1	BR	-	-	-	-	-	-	22
		CL	0.0925	0.3738	0.5993	0.4379	0.3627	30.228	12

Table 10.5: Multi-label performances and time costs of BR and CL on 21 datasets with different C parameters (3).

	C		HL	Prec	Rec	F1	Acc	Time	Labels
Arts	10^3	BR	-	-	-	-	-	-	26
		CL	0.1835	0.1528	0.4650	0.2174	0.1339	33.810	22
	$\frac{10^3}{3}$	BR	0.0542	0.7639	0.2505	0.2649	0.2473	46.826	26
		CL	0.1836	0.3032	0.6452	0.3583	0.2693	29.723	22
	10^2	BR	0.0588	0.8678	0.1008	0.1081	0.1006	47.738	26
		CL	0.2009	0.2684	0.7056	0.3408	0.2433	44.073	22
	10^1	BR	-	-	-	-	-	-	26
		CL	0.1334	0.2003	0.4581	0.2655	0.177	34.3238	22
Business	10^3	BR	0.0253	0.8713	0.7440	0.7663	0.7132	28.858	30
		CL	0.0440	0.7097	0.7770	0.7010	0.6199	26.162	25
	10^2	BR	0.0261	0.8749	0.7188	0.7610	0.7077	29.503	30
		CL	0.0357	0.7735	0.7588	0.7312	0.6625	27.469	25
	10^1	BR	0.0287	0.8633	0.6797	0.7332	0.6797	29.087	30
		CL	0.0287	0.8633	0.6797	0.7332	0.6797	22.501	25
	1	BR	0.0287	0.8633	0.6797	0.7332	0.6797	23.152	30
		CL	0.0287	0.8633	0.6797	0.7332	0.6797	21.982	25
Computers	10^3	BR	0.0348	0.7952	0.3918	0.4110	0.3885	62.604	33
		CL	0.0973	0.4648	0.6950	0.4969	0.4164	41.350	25
	10^2	BR	0.0348	0.7952	0.3918	0.4110	0.3885	61.857	33
		CL	0.0736	0.4846	0.6850	0.5196	0.4334	51.694	25
	10^1	BR	0.0450	0.5194	0.3967	0.4294	0.3967	62.559	33
		CL	0.0716	0.2898	0.4329	0.3298	0.2383	54.732	25
Education	10^3	BR	0.0377	0.6924	0.2774	0.2877	0.2696	50.061	33
		CL	0.1079	0.3123	0.6383	0.3749	0.2844	36.026	25
	10^2	BR	0.0415	0.9079	0.0761	0.0818	0.0761	50.095	33
		CL	0.1157	0.2734	0.6324	0.3456	0.2525	41.059	25
	10^1	BR	-	-	-	-	-	-	33
		CL	0.1857	0.0926	0.3815	0.1407	0.0857	44.649	25

Table 10.6: Multi-label performances and time costs of BR and CL on 21 datasets with different C parameters (4).

	C		HL	Prec	Rec	F1	Acc	Time	Labels
Entertainment	10^3	BR	0.0520	0.8519	0.3360	0.3456	0.3332	50.617	21
		CL	0.1433	0.4028	0.6573	0.4500	0.3697	28.229	16
	10^2	BR	0.0588	0.9444	0.1769	0.1815	0.1769	51.003	21
		CL	0.1321	0.3557	0.6247	0.4137	0.3278	30.079	16
	10^1	BR	0.0672	1.0000	0.0052	0.0052	0.0052	50.480	21
		CL	0.3286	0.1078	0.5770	0.1731	0.1027	34.782	16
	1	BR	-	-	-	-	-	-	21
		CL	0.2413	0.1349	0.5464	0.2078	0.1266	31.675	16
Health	10^3	BR	0.0329	0.8021	0.5512	0.5675	0.5299	47.840	32
		CL	0.0299	0.6140	0.7062	0.6039	0.5636	21.181	25
	10^2	BR	0.0388	0.7955	0.4028	0.4248	0.3985	51.812	32
		CL	0.0419	0.5459	0.6604	0.4832	0.4288	27.785	25
	10^1	BR	0.0504	0.5243	0.3824	0.4139	0.3824	54.038	32
		CL	0.0801	0.2743	0.4151	0.3148	0.2276	27.287	25
	1	BR	0.0515	0.5060	0.3859	0.4190	0.3859	50.227	32
		CL	0.0744	0.3197	0.4747	0.3633	0.2648	26.910	25
Recreation	10^3	BR	0.0551	0.8498	0.2303	0.2380	0.2286	53.232	22
		CL	0.2210	0.3125	0.6966	0.3721	0.2864	23.714	12
	10^2	BR	0.0605	0.9373	0.0944	0.0974	0.0944	55.610	22
		CL	0.2895	0.2381	0.7115	0.3048	0.2234	28.807	12
	10^1	BR	-	-	-	-	-	-	22
		CL	0.2856	0.2155	0.7102	0.3149	0.2187	28.995	12
Reference	10^3	BR	-	-	-	-	-	-	33
		CL	0.0929	0.3474	0.4751	0.3686	0.3292	50.251	25
	$\frac{10^3}{3}$	BR	0.0255	0.8397	0.3902	0.3979	0.3894	58.442	33
		CL	0.0161	0.4892	0.6582	0.5252	0.4618	36.298	25
	10^2	BR	0.0293	0.9554	0.2111	0.2144	0.2111	65.395	33
		CL	0.0605	0.4748	0.5582	0.4796	0.4420	43.302	25
	10^1	BR	0.0322	0.9886	0.1117	0.1131	0.1117	62.886	33
		CL	0.0992	0.2759	0.4484	0.3236	0.2612	42.010	25
	1	BR	-	-	-	-	-	-	33
		CL	0.0375	0.4697	0.4207	0.4362	0.4207	41.621	25

Table 10.7: Multi-label performances and time costs of BR and CL on 21 datasets with different C parameters (5).

	C		HL	Prec	Rec	F1	Acc	Time	Labels
Science	10^3	BR	-	-	-	-	-	-	40
		CL	0.0639	0.2170	0.3417	0.2543	0.1860	48.368	30
	$\frac{10^3}{3}$	BR	0.0312	0.7908	0.1906	0.1991	0.1889	84.891	40
		CL	0.0957	0.2725	0.5939	0.3381	0.2521	59.105	30
	10^2	BR	0.0350	0.9318	0.0245	0.0254	0.0245	84.291	40
		CL	0.1295	0.1758	0.5346	0.2423	0.1643	66.323	30
	10^1	BR	-	-	-	-	-	-	40
		CL	0.1319	0.1046	0.3918	0.1596	0.0989	70.387	30
Scocial	10^3	BR	0.0208	0.9126	0.4876	0.4992	0.4869	85.668	39
		CL	0.0597	0.5642	0.7544	0.5913	0.5288	59.048	29
	10^2	BR	0.0234	0.9708	0.3632	0.3706	0.3632	83.811	39
		CL	0.0782	0.4739	0.7200	0.5103	0.4475	63.289	29
	10^1	BR	0.0307	0.9964	0.0849	0.0872	0.0849	89.083	39
		CL	0.0607	0.3076	0.5526	0.3711	0.2817	75.651	29
	1	BR	-	-	-	-	-	-	32
		CL	0.0492	0.3423	0.6019	0.4267	0.3239	69.870	29
Scociety	10^3	BR	0.0516	0.7593	0.3344	0.3546	0.3318	71.275	27
		CL	0.1429	0.3785	0.5924	0.4044	0.3191	43.211	19
	10^2	BR	0.0545	0.7937	0.2327	0.2476	0.2327	74.344	27
		CL	0.2092	0.2886	0.6073	0.3267	0.2480	42.359	19
	10^1	BR	0.0594	0.9761	0.0695	0.0729	0.0695	73.640	27
		CL	0.1041	0.2986	0.4161	0.3212	0.2433	46.004	19
	1	BR	-	-	-	-	-	-	27
		CL	0.0928	0.2945	0.4289	0.3289	0.2398	41.521	19

Table 10.8: Multi-label performances and time costs of BR and CL on 21 datasets with different C parameters (6).

Another example is that a high precision can associate with other four small metrics when the number of positive samples in the predicted one is very small. Compared with precision and recall, F1 score and accuracy are more robust to

the above problems.

We show the multi-label prediction experimental results of CL and BR given different C in Table 10.3 to Table 10.8. In these tables, we use "-" as the mark of prediction failure, namely, predicting all the samples into one class in each binary classification. The "Time" column in the tables is the sum of training and test CPU seconds. Thus the CPU seconds for label compression and recovery in CL is included in "Time". The best performance of BR and CL are shadowed with different colors in the tables. For each dataset, we choose the number of CL labels much less than the number of the original ones except in datasets Scene and Emotions, where the numbers of labels are too small (≤ 6) and thus random projection signs less than it are not stable for estimating the empirical joint distribution.

In most experiments, CL significantly outperforms BR on different metrics and different values of C. In particular, CL has overwhelming privilege on large-scale datasets with large number of labels (which is one of the most difficult cases in multi-label learning), e.g., Bibtex, Corel5k, Enron and Medical. CL also has appealing prediction performance on small datasets whose density is small, e.g., Genbase, Slashdot and most sub datasets of Yahoo. Another interesting phenomenon is that the dimension increasing of label matrix on Scene dataset generates a satisfying prediction. This indicates that CL can also be used to improve multi-label prediction on dataset with small number of labels by increasing the dimensionality of CL labels. There exist rare cases when BR has better performance, e.g., on Yahoo-Business with $C = 10^3$ and $C = 10^2$, but CL and BR arrive the same metrics when $C = 10^1$ and $C = 1$.

CL also brings a tremendous reduction of the time cost. According to the CPU seconds and the number of labels shown in the tables, CL can compress the original label matrix to a low dimensionality, which saves great computation in the training stage. Therefore, CL can reduce the problem size of multi-label learning and makes the large-scale problems computationally tractable.

Since the unique difference between CL and BR is that CL invokes linear SVM on the compressed labels rather than on the original ones as BR does, this group of experimental results shows that single-label classification methods can be conveniently extended to solve multi-label learning problems with improved

performance and reduced time cost by using the label compression and recovery scheme proposed in CL. The leverages in both effectiveness and efficiency are attributed to the removal of the three main problems of applying single-label learning method to multi-label learning tasks. In particular, the performance improvement benefits from the sample balance and exploration of label correlation in CL, while the time cost is decreased by label compression. In addition, the benefits and the strength of conventional single-label learning methods are well inherited in the scheme of CL. Therefore, CL proposes a general extension of single-label learning method to multi-label scenarios rather than merely a specific algorithm for multi-label learning.

CL is robust to the change of parameters in the training stage. In the experimental results, CL can generate appealing prediction result when BR fails on some C values. Moreover, its performance changes a little when C decreases from 10^3 to 10^{-3}, e.g., on the Scene dataset. We explain the reasons for the improvement of robustness on two aspects:

1 SVM is more robust to the parameter C on more balanced data. In imbalanced data, the insufficient number of samples in the minor class are more likely to be within the margin when C decreases, i.e., when the margin becomes larger. This leads to less support vectors for the minor class. Thus the prediction performance of BR drops dramatically with the decreasing of C. CL trains SVM classifiers on compressed labels with improved sample balance, and thus the prediction performance is more robust to a wide range of C.

2 In CL, although the number of labels are reduced due to the label compression, the number of labels that each sample belongs to is increased, because the label matrix is much more dense and balanced. Thus the information of the original labels are disseminated rather than condensed in the CL labels. In the prediction stage, each original label is predicted according to the statistics of all the CL labels. This scheme makes the prediction of the original label robust to the failures in predicting one or two CL labels.

Therefore, CL improves the robustness of the multi-label learning via random projection signs of the labels.

10.5.5 Multi-label prediction: comparison with other multi-label learning methods

In this group of experiments, we compare CL with 3 popular multi-label learning methods, i.e., ML-knn [246], MDDM [253] and multi-label prediction based CS (ML-CS) [122]. ML-knn is an extension of knn. It obtains the label prior distribution from the k nearest neighbors and maximizes a posterior to the label prediction. ML-knn aims at solving the imbalance problem of the k nearest neighbors' labels. MDDM tackles the "curse of dimensionality" in multi-label data and formulates the problem as a discriminative dimension reduction [208]. It maximizes the dependence between feature space and label space via maximizing the empirical estimate of Hilbert-Schmidt Independence Criterion. Then ML-knn can be applied to the obtained low-dimensional subspace. MDDM aims at decreasing the time complexity of the sequential ML-knn.

ML-CS compresses the original $0-1$ label matrix Y via random projection $Y' = YA$ to a real valued matrix Y', and then builds a new linear regression model $Y' = XW'$. In its prediction, the original label vector y is recovered from y' via compressed sensing or model selection algorithms such as least angle regression (LARS) [77]. ML-CS is related to CL because it builds models on the low-dimensional embedding of the original label. Their main difference is that CL compresses the original $0-1$ label matrix to another $0-1$ label matrix and builds binary classification models on it rather than regression models in the training stage.

In the experiments, we set the number of neighbors in ML-knn as 20 and the dimensions of the subspace obtained in MDDM as 40% of the dimensions of the original data. In ML-CS, the low-dimensional embedding of the label matrix has the same dimension as the compressed label matrix in CL. We use LARS as the recovery algorithm in its prediction stage. The sparse level is determined by selecting the solution with the minimum least square measurement error in the same cardinality range as CL. This is similar to the last step in the recovery algorithm of CL in Algorithm 13. The performance of CL is collected from the first group of experiments.

We show the experimental results in Table 10.9 and Table 10.10.

	Methods	HL	Prec	Rec	F1	Acc	Time	Labels
Bibtex	ML-knn	0.0140	0.5511	0.0397	0.2034	0.0364	42697	159
	MDDM	0.0140	0.5334	0.0534	0.2138	0.0492	970.7	159
	ML-CS	0.1645	0.0798	0.6499	0.1026	0.0760	1901.5	80
	CL	0.1075	0.2210	0.5658	0.2445	0.1839	1640	80
Corel5k	ML-knn	0.0093	0.6197	0.0103	0.0321	0.0098	2106	374
	MDDM	0.0093	0.6166	0.0161	0.0755	0.0146	458	374
	ML-CS	0.0857	0.0789	0.5108	0.1012	0.0741	612.6	160
	CL	0.0153	0.2418	0.2595	0.2392	0.1585	526.5	160
Mediamill	ML-knn	0.0314	0.4132	0.0632	0.5377	0.0534	5713	101
	MDDM	0.0319	0.3679	0.0517	0.5278	0.0439	48237	101
	ML-CS	0.1116	0.2503	0.7033	0.3402	0.2206	99.48	40
	CL	0.0390	0.5751	0.4629	0.4659	0.3467	76.70	40
IMDB	ML-knn	-	-	-	-	-	$> 10^5$	28
	MDDM	-	-	-	-	-	$> 10^5$	28
	ML-CS	0.1890	0.1654	0.4228	0.2286	0.1406	3608	20
	CL	0.1492	0.2621	0.5178	0.3080	0.2187	3273	20
Enron	ML-knn	0.0518	0.5083	0.0665	0.4632	0.0532	527	53
	MDDM	0.0505	0.5000	0.0883	0.4966	0.0715	29	53
	ML-CS	0.3857	0.1315	0.6732	0.1747	0.1189	40.36	36
	CL	0.0779	0.5027	0.5926	0.5076	0.3905	35.60	36
Genbase	ML-knn	0.0065	1.0000	0.5071	0.9231	0.5071	9.38	32
	MDDM	0.0063	0.9881	0.5170	0.9258	0.5133	6.09	32
	ML-CS	0.0143	0.9140	0.9949	0.8627	0.9104	0.327	20
	CL	0.0101	0.9605	0.9728	0.9568	0.9441	0.321	20
Medical	ML-knn	0.0204	0.7554	0.0743	0.4879	0.0657	22.8	45
	MDDM	0.0249	0.7395	0.0323	0.3058	0.0258	32.3	45
	ML-CS	0.3306	0.0896	0.9356	0.1334	0.0893	3.14	25
	CL	0.0273	0.5904	0.8147	0.6565	0.5694	1.43	25
Emotions	ML-knn	0.2855	0.6852	0.2819	0.4155	0.2237	0.66	6
	MDDM	0.2962	0.5437	0.2885	0.4163	0.2239	0.66	6
	ML-CS	0.3960	0.4339	0.6113	0.5171	0.3927	0.68	15
	CL	0.3201	0.5208	0.6007	0.5353	0.4373	0.51	15
Scene	ML-knn	0.1016	0.7788	0.6257	0.6881	0.5373	14.3	6
	MDDM	0.1044	0.7568	0.6393	0.6870	0.5338	7.59	6
	ML-CS	0.4123	0.2672	0.6768	0.3734	0.2635	20.42	15
	CL	0.0780	0.6080	0.8023	0.6627	0.5887	16.50	15
Slashdot	ML-knn	0.0481	0.7104	0.1016	0.3067	0.0800	708	22
	MDDM	0.0518	0.3973	0.0129	0.0452	0.0118	114	22
	ML-CS	0.3866	0.0881	0.6774	0.1530	0.0866	31.86	12
	CL	0.0833	0.4269	0.6514	0.4913	0.4122	27.68	12

Table 10.9: Prediction performances and time costs of ML-knn, MDDM, ML-CS and CL on 10 datasets. "-" denotes the failed experiment whose time cost exceeds 10^5 secondes.

	Methods	HL	Prec	Rec	F1	Acc	Time	Labels
Arts	ML-knn	0.0587	0.6257	0.0708	0.2477	0.0639	77.6	26
	MDDM	0.0595	0.6783	0.0614	0.2123	0.0562	37.4	26
	ML-CS	0.4538	0.1004	0.7994	0.1753	0.0975	35.3	22
	CL	0.1836	0.3032	0.6452	0.3583	0.2693	29.7	22
Business	ML-knn	0.0266	0.6789	0.0920	0.7042	0.0822	93.2	30
	MDDM	0.0279	0.6609	0.0733	0.6878	0.0661	42.7	30
	ML-CS	0.1089	0.3975	0.8186	0.4246	0.3663	30.3	25
	CL	0.0287	0.8633	0.6797	0.7332	0.6797	21.9	25
Computers	ML-knn	0.0408	0.6959	0.0344	0.3328	0.0303	124	33
	MDDM	0.0414	0.5669	0.0497	0.4075	0.0403	50	33
	ML-CS	0.1329	0.3149	0.6432	0.2914	0.2817	54.0	25
	CL	0.0736	0.4846	0.6850	0.5196	0.4334	51.6	25
Education	ML-knn	0.0389	0.5883	0.0618	0.3159	0.0560	99.8	33
	MDDM	0.0398	0.5914	0.0502	0.2655	0.0468	45.2	33
	ML-CS	0.1382	0.2406	0.6218	0.2755	0.2188	40.9	25
	CL	0.1079	0.3123	0.6383	0.3749	0.2844	36.0	25
Entertain	ML-knn	0.0575	0.6662	0.1126	0.3328	0.1067	108	21
	MDDM	0.0585	0.6485	0.1093	0.3005	0.1040	43.8	21
	ML-CS	0.2202	0.2192	0.5965	0.2550	0.2027	28.8	16
	CL	0.1433	0.4028	0.6573	0.4500	0.3697	28.2	16
Health	ML-knn	0.0362	0.7316	0.1618	0.5657	0.1471	101	32
	MDDM	0.0391	0.7040	0.1448	0.5138	0.1310	46.6	32
	ML-CS	0.1092	0.3739	0.7319	0.3900	0.3384	23.3	25
	CL	0.0299	0.6140	0.7062	0.6039	0.5636	21.1	25
Recreation	ML-knn	0.0595	0.7016	0.0862	0.2310	0.0804	112	22
	MDDM	0.0612	0.6655	0.0725	0.1801	0.0671	41.9	22
	ML-CS	0.2264	0.1794	0.5383	0.2228	0.1653	24.6	12
	CL	0.2210	0.3125	0.6966	0.3721	0.2864	23.7	12
Reference	ML-knn	0.0274	0.6663	0.0699	0.4696	0.0638	136	33
	MDDM	0.0290	0.6620	0.0676	0.4325	0.0598	51.6	33
	ML-CS	0.1404	0.2610	0.6823	0.2484	0.2480	24.1	25
	CL	0.0161	0.4892	0.6582	0.5252	0.4618	36.2	25
Science	ML-knn	0.0330	0.5999	0.0479	0.2056	0.0446	139	40
	MDDM	0.0336	0.6603	0.0456	0.1986	0.0424	53	40
	ML-CS	0.1851	0.1496	0.6130	0.1792	0.1405	45.8	30
	CL	0.0957	0.2725	0.5939	0.3381	0.2521	59.1	30
Social	ML-knn	0.0219	0.7759	0.0718	0.5537	0.0675	175	39
	MDDM	0.0238	0.6751	0.0577	0.5003	0.0535	70.8	39
	ML-CS	0.1443	0.2215	0.7271	0.2386	0.2123	65.2	29
	CL	0.0597	0.5642	0.7544	0.5913	0.5288	59.0	29
Society	ML-knn	0.0547	0.6258	0.0574	0.3422	0.0499	111	27
	MDDM	0.0542	0.6077	0.0530	0.3136	0.0488	44.1	27
	ML-CS	0.2130	0.2214	0.5983	0.2351	0.1975	49.9	19
	CL	0.1429	0.3785	0.5924	0.4044	0.3191	43.2	19

Table 10.10: Prediction performances and time costs of ML-knn, MDDM, ML-CS and CL on 11 sub datasets from Yahoo dataset.

In most experiments, CL outperforms the other 3 multi-label learning methods on most performance metrics. Since ML-knn is more robust to sample imbalance than BR based on SVM, and ML-CS eliminates the sample imbalance via random projections, sample balance problem are considered in these 3 methods. Moreover, ML-knn explores the label correlation by its instance-based learning scheme. Thus the 3 methods have better prediction performance than BR in the first group of experiments. However, ML-knn and MDDM are vulnerable to dataset with ultra sample imbalance, e.g., Bibtex, Corel5k and Mediamill, where they have high precision and F1 score but near 0 recall and accuracy. ML-knn and MDDM have similar performance on most datasets, but MDDM exceeds ML-knn on datasets of high data dimensionality such as Bibtex and Enron. This indicates the advantages of MDDM on high dimensional data when ML-knn is applied as the classification model. ML-CS performs better than ML-knn and MDDM because of the independence among the dimensions of the label embedding brought by random projection. The main reason that ML-CS does not outperform CL is that ML-CS transforms the original discrete classification model to a continuous linear regression model, which small estimation error may induce large error in the label recovery. Although both the estimation error of linear regression model and the recovery error of the compressed sensing algorithm can be theoretically bounded, it is not clear how the linear regression estimation error as an input of the recovery influences the final classification error. CL remains training discrete classification model (of smaller size than the original one) on CL labels and thus does not meet this problem. Another advantage of CL comparing with ML-CS is that the label correlations are explicitly and directly explored via LDM, and thus the prediction is improved.

The time cost of CL is less than those of ML-knn and MDDM in all the experiments, especially on the large-scale datasets such as Bibtex, Corel5k and Mediamill. ML-knn and MDDM fail on IMDB because we fail to train them in 10^5 CPU seconds. This is due to the expensive time complexity $\mathcal{O}(n^2p)$ of the pairwise distance calculation in ML-knn and MDDM. MDDM is more efficient than ML-knn with a decreased p. The time costs of CL and ML-CS are close to each other, and CL performs slightly faster than ML-CS on most datasets. However, it is worthy to note that CL and ML-CS have different training time

and prediction time. In particular, the training of SVM in CL is slower than the training of linear regression in ML-CS. But the prediction time of CL is much less than that of ML-CS, because the recovery algorithm in CL is based on a simple statistical test, while the recovery algorithm in ML-CS invokes time consuming ℓ_1 minimization. In addition, CL can be substantially faster than ML-CS if we replace LIBSVM with other efficient SVM solvers such as NESVM.

10.5.6 Multi-label prediction: comparison with 2 SVM algorithms dealing with imbalanced data

In this group of experiments, we compare CL with 2 representative SVM extensions dealing with the sample imbalance problem, i.e., SVM-SMOTE [45] and SVM-WEIGHT [184]. The goal of these experiments is to 1) study whether the sample balance obtained via random projection signs in CL works well as the other methods which imposes over-sampling or larger weight to the minor class; 2) evaluate the performance improvement caused by exploring label correlation in CL isolated from that caused by the sample balance. In the experiments of SVM-SMOTE and SVM-WEIGHT, we replace the ordinary SVM in BR with these two modified versions, respectively. SVM-SMOTE invokes SMOTE algorithm to generate synthetic but pseudo samples for minor (positive) class and then train SVM on an dataset with over-sampled positive samples and the original negative ones. We let the positive sample has the same amount of the negative ones after over-sampling. SVM-WEIGHT assigns a larger weight penalty to the major class, which leads to a larger margin for the major class than that for the minor class. We set the weights for the negative samples and the positive ones in proportion to the ratio of their amounts. We show the experimental results in Table 10.11.

Table 10.11 shows that CL outperforms SVM-SMOTE and SVM-WEIGHT on both effectiveness and efficiency. Compared with the performance of BR using the ordinary SVM in the first group of experiments, the 2 SVM algorithms are designed to alleviate the harm brought by sample imbalance to SVM, and thus outperform BR.

	Methods	HL	Prec	Rec	F1	Acc	Time	Labels
Medical	SMOTE	0.9679	0.0129	0.3964	0.0253	0.0128	2.435	45
	WEIGHT	0.3413	0.0388	0.3078	0.0548	0.0377	2.708	45
	CL	0.0273	0.5904	0.8147	0.6565	0.5694	1.433	25
Emotions	SMOTE	0.2962	0.4600	0.57178	0.4562	0.3982	1.625	6
	WEIGHT	0.2962	0.4547	0.5379	0.4426	0.3805	0.253	6
	CL	0.3201	0.5208	0.6007	0.5353	0.4373	0.517	15
Scene	SMOTE	0.1128	0.5301	0.6182	0.5631	0.5261	23.42	6
	WEIGHT	0.1128	0.5318	0.6207	0.4938	0.5269	9.244	6
	CL	0.0780	0.6080	0.8023	0.6627	0.5887	16.50	15
Genbase	SMOTE	0.9885	0.0086	0.1264	0.0164	0.0082	2.098	32
	WEIGHT	0.5736	0.0497	0.7089	0.0945	0.0488	0.964	32
	CL	0.0101	0.9605	0.9728	0.9568	0.9441	0.321	20
Slashdot	SMOTE	0.9311	0.0270	0.4361	0.0516	0.0268	43.65	22
	WEIGHT	0.3102	0.0797	0.4310	0.1366	0.0778	121.5	22
	CL	0.0833	0.4269	0.6514	0.4913	0.4122	27.68	12
Arts	SMOTE	0.1200	0.3172	0.5360	0.3421	0.2696	115.4	26
	WEIGHT	0.1195	0.3143	0.5300	0.3403	0.2674	54.54	26
	CL	0.1836	0.3032	0.6452	0.3583	0.2693	29.72	22
Business	SMOTE	0.0448	0.6763	0.8006	0.6462	0.5958	86.36	30
	WEIGHT	0.0458	0.6577	0.7171	0.6179	0.5535	38.99	30
	CL	0.0287	0.8633	0.6797	0.7332	0.6797	21.98	25
Computers	SMOTE	0.0724	0.4435	0.5849	0.4161	0.3765	141.1	33
	WEIGHT	0.0709	0.4505	0.5722	0.4154	0.3790	79.93	33
	CL	0.0736	0.4846	0.6850	0.5196	0.4334	51.69	25
Education	SMOTE	0.0678	0.3341	0.5516	0.3529	0.2279	111.9	33
	WEIGHT	0.0680	0.3327	0.5548	0.3531	0.2290	53.89	33
	CL	0.1079	0.3123	0.6383	0.3749	0.2844	36.02	25
Entertain	SMOTE	0.0920	0.4587	0.5948	0.4437	0.4001	121.4	21
	WEIGHT	0.0920	0.4566	0.5882	0.4415	0.3961	62.37	21
	CL	0.1433	0.4028	0.6573	0.4500	0.3697	28.22	16
Health	SMOTE	0.0652	0.5637	0.6616	0.5417	0.4709	102.8	32
	WEIGHT	0.0650	0.5605	0.6567	0.5409	0.4692	48.46	32
	CL	0.0299	0.6140	0.7062	0.6039	0.5636	21.18	25
Recreation	SMOTE	0.109	0.3062	0.5339	0.3116	0.2773	121.5	22
	WEIGHT	0.1081	0.3036	0.5295	0.3208	0.2843	59.22	22
	CL	0.2210	0.3125	0.6966	0.3721	0.2864	23.71	12
Reference	SMOTE	0.0413	0.5096	0.5366	0.4595	0.4468	123.4	33
	WEIGHT	0.0411	0.5091	0.5330	0.4583	0.4452	69.04	33
	CL	0.0161	0.4892	0.6582	0.5252	0.4618	36.29	25
Science	SMOTE	0.0563	0.3584	0.4780	0.3604	0.3023	186.5	40
	WEIGHT	0.0556	0.3622	0.4669	0.3579	0.3011	97.09	40
	CL	0.0957	0.2725	0.5939	0.3381	0.2521	59.10	30
Social	SMOTE	0.0293	0.6698	0.6374	0.5739	0.5566	163.8	39
	WEIGHT	0.0292	0.6689	0.6356	0.5732	0.5532	87.57	39
	CL	0.0597	0.5642	0.7544	0.5913	0.5288	59.04	29
Society	SMOTE	0.1100	0.3272	0.5022	0.3240	0.2532	173.5	27
	WEIGHT	0.1088	0.3306	0.5038	0.3263	0.2567	92.26	27
	CL	0.1429	0.3785	0.5924	0.4044	0.3191	43.211	19

Table 10.11: Prediction performances and time costs of SVM-SMOTE, SVM-WEIGHT and CL on 5 datasets.

The phenomenon of "high precision low recall" is rare in this group of experiments (though still exists on Genbase and Slashdot), which indicates that the sample balance is helpful for improving multi-label prediction. Different from over-sampling and weighting in SVM-SMOTE and SVM-WEIGHT, CL eliminates the sample balance via random projection signs. The experimental results show this method works well as the other two. Compared with SVM-SMOTE and SVM-WEIGHT, the improvement of prediction performance in CL is due to the application of DLs that preserve label correlation. Therefore, the sample balance and label correlation indeed help to improve the prediction in CL.

SVM-SMOTE adopts over-sampling technique and thus increases the time complexity of the original SVM in BR by augmenting the training samples. SVM-WEIGHT has similar time complexity of the original SVM in BR, but the changing of weight in major (negative) samples may influence the convergence of the SVM training, and thus the time cost will be slightly different. CL decreases the time complexity of BR by reducing the number of SVM models that need to be trained, and thus has the smallest time cost among the 3 methods in this group of experiments.

10.5.7 Compression-performance trade-off

In this group of experiments, we study the trade-off between the recovery error due to label compression and the improvement brought by sample balance for prediction performance in CL. In particular, we evaluate the time cost and the 5 performance metrics of CL without using DL on 10 different label compression ratios between 0 and 1. The C parameter for each dataset is selected as the C with the best performance on the same dataset in the first group of experiments. We use NESVM [270] as the SVM solver. In order to isolate the improvement caused by sample balance, we eliminate the influence of label correlation via replacing DLs in CL with the dictionary D composed of unit vectors, i.e.,

$$D = \left\{e^i\right\}_{i=1,\cdots,k}, e^i = [0, \cdots, 0, 1, 0, \cdots, 0], e^i_i = 1. \tag{10.89}$$

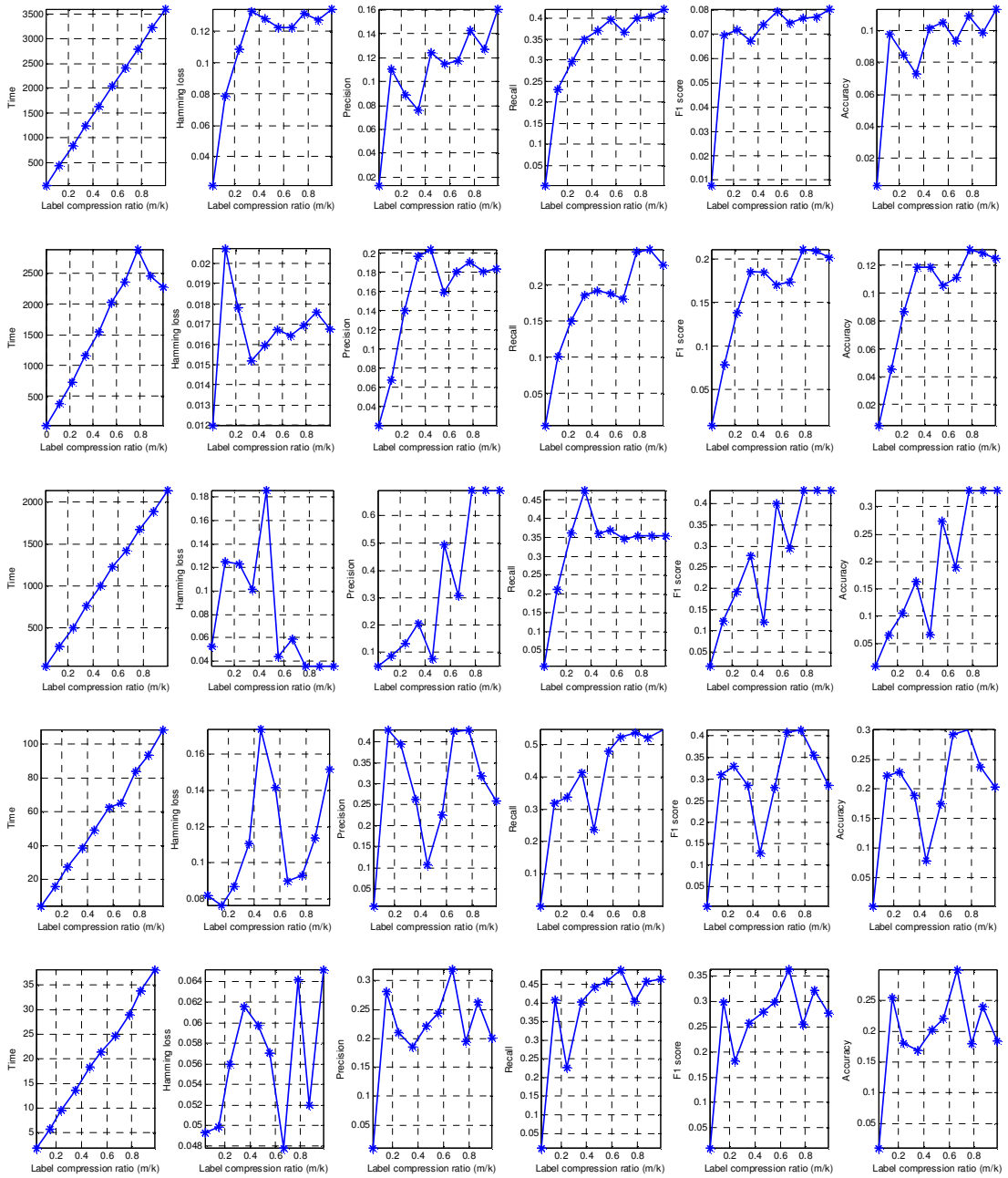

Figure 10.9: Trade-off between label compression and 5 prediction performance metrics, time costs on 5 datasets. From top to bottom: Bibtex, Corel5k, Mediamill, Enron and Medical.

In order to ensure that the least label compression ratio can generate more

than 2 CL labels, we choose the 5 datasets with the largest number of labels k which is larger than 40 among the 21 datasets and show their trade-off curves in Figure 10.9. There are some noises that make the curves not strictly monotonic. These noises are generated by the randomness of CL labels. However, these noises slightly affect the trends of the trade-off curves.

Among the 6 trade-off curves, the time cost linearly grows with the increasing of the label compression ratio. This can be explained by that the increasing of SVM models needs to be trained in CL with the increasing compression ratio. The hamming loss does not affect the trend because it is unstable when the sample imbalance is severe. In this case, a failed prediction that assigns all the test samples to negative classes can produce an extremely small hamming loss. The other 4 metrics, i.e., precision, recall, F1 score and accuracy, rapidly increase with the increasing of the compression ratio. Moreover, they quickly reach satisfactory performances when the compression ratio is fairly small (less than 0.4) on all the 5 datasets. These results indicate that the prediction performance of CL stably and rapidly improves with the increasing of CL labels. Furthermore, CL reaches satisfactory sample balance when the compression ratio is small. Thus CL finds the equilibrium with both competitive efficiency and effectiveness.

Another interesting property shown on the trade-off curves is the improvement brought by exploring the label correlation in CL. For example, the best performance of CL on Bibtex in Figure 10.9 with compression ratio 1 and without using of DLs is much less than the performance with compression ratio 0.5 shown in Table 10.9 where DLs is applied. This difference suggests that LDM preserves label correlations in improving multi-label prediction performance.

10.6 Conclusion

In this chapter, we have proposed a label transformation method "compressed labeling (CL)" for multi-label learning. In the training stage of CL, the original label matrix is compressed to the sign matrix of its low-dimensional random projections on a standard Gaussian ensemble. Existing binary classification is then directly applicable to the labels in the new label matrix. In the prediction stage of CL, the CL label vector of a given sample is initially estimated by using

the classifiers obtained in the training stage. Afterward, a fast recovery algorithm is used to reconstruct the original label vector from the CL label vector.

We have developed a greedy binary matrix decomposition method, the "labelset distilling method (LDM)", to extract the discrete patterns, i.e., distilled labelsets (DLs) that are frequently shared by the label vectors in the original label matrix. LDM recursively divides the label vectors in the label matrix into several clusters, extracts the shared label subset of each cluster as a distilled labelset (DL), adds it to the DLs and subtracts it from the label vectors in the corresponding cluster. We have discovered that the random projection signs of an arbitrary DL and a given label vector have an explicit joint distribution in two cases, i.e., the DL is included in the label vector, and the DL is not included in the label vector. The computation of these two kinds of joint distributions is accomplished by a geometric inference.

In the recovery algorithm of CL, the empirical estimation of the joint distribution is calculated from the predicted CL label vector and the random projection signs of distilled labelsets. A series of statistical tests are conducted on the empirical joint distributions of all the DLs to determine whether each DL is included in the original label vector. This test is based on a comparison of the KL divergences between the empirical joint distribution and the two explicit joint distributions in two different cases.

CL solves or at least substantially alleviates the three main problems harassing multi-label learning, which are the problem of sample imbalance, the problem of label dependence and the problem of label high dimensionality. CL is a general method that allows direct embedding of most existing multi-label and single-label learning techniques in its training stage, their advantages will be inherited in their CL variants for multi-label learning. Since CL significantly reduces the problem size by label compression and compressed CL labels can be efficiently recovered, it provides an efficient solver for large-scale multi-label learning tasks. In CL, the label dependence information is stored in DLs after compression and used in the recovery algorithm. LDM is also an isolated method that can be used in other multi-label learning methods to exploit the label dependence. To our best knowledge, CL is the first multi-label learning method with model complexity much less than that of BR, while the label correlations are simultaneously explored.

Theoretically, we have proved that the probabilistic upper bounds of recovery failures exponentially shrink with increasing either the dimensionality of the CL label vector or the cardinality of DL. These analyses demonstrate the effectiveness of label compression and prediction improvement brought by LDM. In the near future, we will theoretically study how the error (or regret) of the learned binary classifiers for the subproblems is transformed into the error (or regret) of the final classifier for the multi-label problem.

We have evaluated the performance of CL in label compression/recovery and multi-label prediction via 5 groups of experiments on 21 datasets from different real-world problems. In the experiments, we have compared CL with BR, 3 multi-label learning methods (ML-knn, MDDM and ML-CS) and 2 SVM algorithms dealing with imbalanced data on 5 prediction performance metrics and time cost. We also have studied the trade-off between compression and performance improvement in CL without using DLs. The experimental results of label compression/recovery have verified our theoretical analyses about the improvement of the sample balance, elimination of label independence and accurate recovery brought by CL. The five groups of multi-label prediction experiments demonstrate the competitive prediction performance, efficiency and robustness of CL in multi-label learning.

Chapter 11

Conclusions

This era is highly featured by an avalanche of massive data collected from the rapidly expanding Internet, increasing consumer digital products and sensors. Data become much cheaper and easier to obtain than ever before. At the first glance, the big data provide affluent resources for machine learning technique to train and create more intelligent systems. But the brutal truth is that the sheer volume and high dimensions of big data make various machine learning algorithms computationally prohibitive. What is worse, the underlying complicated structures, contaminating noise, missing values, unexpected outliers and varying individuality across data instances easily lead to the failure of the ideal and inflexible data assumptions in lots of machine learning methods, which may be effective on well designed or carefully processed data. Furthermore, the stability, accuracy and other good statistical properties of many sophisticatedly designed approaches no longer hold in the case of big data. Therefore, new challenges arise in machine learning due to those changes of their research objects.

The data structure played a vital role in the development of previous machine learning techniques. Two influential examples are low-rank structure existing in multiple data samples, and sparse structure representing a data sample by its spiky spectrum. These two structures have been also broadly studied in signal processing and information theory. They show the exploration of data redundancy in machine learning, which aims at bridging the extracted structures with the objectives of learning tasks.

In this thesis, we propose the concept of "compressed learning", which lever-

ages the possibly compressible structures in both data and learning procedures to not only produce effective learning methods on big data, but also design scalable algorithms fast in speed. In particular, beyond low-rank and sparse structures, we develop more interpretable and effective structures for feature extraction which not only compresses data on dimension but also on cardinality (manifold elastic net (MEN) in Chapter 2 and double shrinking (DS) in Chapter 3 encouraging sparse features in low dimensions), more expressive and general structures revealing both the relations and differences across data samples by decomposing data as sum mixture of low-rank and sparse parts (GO decomposition (GoDec) in Chapter 6 and greedy bilateral (GreB) paradigm in Chapter 7), and their more complicated but more adaptive variants considering the nonlinear transformation, linear functional and subcomponents of low-rank part (shifted subspace tracking (SST), multi-label subspace ensemble (MSE) and linear functional GoDec (Lin-GoDec) in Chapter 9).

Moreover, we study how to extract the compressible structures in noisy case (BRP and GreBske in Chapter 5, GoDec in Chapter 6, GreB in Chapter 7) and missing value case (GoDec and greedy bilateral completion (GreBcom) for matrix completion in Chapter 8), which are common in big data but difficult for conventional machine learning methods. Furthermore, we develop randomized low-rank approximation with closed form (bilateral random projections (BRP) in Chapter 5, GoDec in Chapter 6 and Chapter 8, SST and MSE in Chapter 9), greedy strategy (GreBske in Chapter 5, GreB and GreBsmo in Chapter 7, GreBcom in Chapter 8), and divide-and-conquer method (divide-and-conquer anchoring (DCA) in Chapter 4) that can produce scalable and distributable algorithms for practical applications on the sheer volume of data.

Furthermore, rather than merely compressing data features, we show different novel insights of compressed learning, e.g., the compression of dataset by selecting representative samples even from highly incomplete data (DCA for near-separable NMF in Chapter 4), the compression of responses and classifiers to learn when solving related multiple supervised learning tasks (compressed labeling (CL) in Chapter 10), and the compression of relatedness among discrete variables by inverse mapping from label space to feature subspaces (MSE in Chapter 9).

Compressed learning offers new methodology for solving learning problems

from conventional clustering, classification and dimensional reduction to multi-label learning, sparse learning, manifold learning, motion segmentation, recommendation, and relational learning, especially when dealing with big data. The compressed learning approaches presented in this thesis successfully address the challenging practical problems from computer vision (detection, tracking, motion analysis), recommendation systems, image and web text annotation, face recognition, handwritten letter clustering, complex data visualization, and compressed sensing.

However, it is important to point out that the techniques we proposed in this thesis are not adequate to solve all the problems related to compressed learning. For example, the low-rank assumption plays a critical role in most techniques and indicates the existence of low-rank linear representation of all the data points. But it is more interesting to explore more complicated nonlinear structures in many other real applications like we show in SST for motion segmentation. In addition, the sparse structure could be more complex on real data. So structured sparsity such group sparsity, total variation and fused lasso penalty might be more effective when solving specific problems. Moreover, it is possible to relate the matrix factorization/completion tools proposed in this thesis to graphical model learning problems, in which the compressible structures might be able to produce more efficient algorithms. Furthermore, since all the proposed methods are featured by their competitive computational speed and small time cost, it is significant to study the online learning versions of some proposed approaches, and the practical implementation of these techniques in real systems. Last but not the least, besides applying compression to features, samples and outputs in learning problems, other compressible components also need to be studied and explore.

We believe that compressed learning, as a novel learning methodology emphasizing the exploration of compressible structures in developing machine learning and also a nontrivial extension rooted from classic machine learning techniques, will receive growing attention and encourage influential research results in the future, due to its promising effectiveness, adaptivity, robustness, scalability and easy implementation on big data analytics problems.

References

[1] M. Aharon, M. Elad, and A. Bruckstein. "K-SVD: An Algorithm for Designing Overcomplete Dictionaries for Sparse Representation". In: *IEEE Transactions on Signal Processing* 54.11 (2006), pp. 4311–4322. (Cit. on pp. 6, 81, 183).

[2] N. Ailon and B. Chazelle. "Approximate nearest neighbors and the fast Johnson-Lindenstrauss transform". In: *STOC '06: The 28th annual ACM symposium on Theory of computing*. 2006, pp. 557–563. (Cit. on p. 185).

[3] S. Ali and M. Shah. "A Lagrangian Particle Dynamics Approach for Crowd Flow Segmentation and Stability Analysis". In: *IEEE Conference on Computer Vision and Pattern Recognition*. 2007. (Cit. on pp. 186, 188).

[4] A. Alizadeh et al. "Distinct types of diffuse large b-cell lymphoma identified by gene expression profiling". In: *Nature* 403 (2000), pp. 503–511. (Cit. on pp. 85, 96).

[5] U. Alon et al. "Broad patterns of gene expression revealed by clustering analysis of tumor and normal colon tissues probed by oligonucleotide arrays". In: *Cell Biology* 96 (1999), pp. 6745–6750. (Cit. on pp. 85, 96).

[6] R. Ando and T. Zhang. "A framework for learning predictive structures from multiple tasks and unlabeled data". In: *Journal of Machine Learning Research* 6 (2005), pp. 1817–1853. (Cit. on p. 6).

[7] A. Argyriou, T. Evgeniou, and M. Pontil. "Multi-task feature learning". In: *NIPS*. 2007. (Cit. on p. 6).

[8] E. Arias-Castro, D. L. Donoho, and X. Huo. "Near-optimal detection of geometric objects by fast multiscale methods". In: *IEEE Transactions on Information Theory* 51.7 (2005), pp. 2402–2425. (Cit. on p. 54).

[9] S. Arora et al. "Computing a nonnegative matrix factorization - provably". In: *Proceedings of the 44th Symposium on Theory of Computing (STOC)*. 2012. (Cit. on pp. 106, 118).

[10] A. Asuncion and D. Newman. "UCI Machine Learning Repository". In: *Available: http://www.ics.uci.edu/∼mlearn/MLRepository.html* (2007). (Cit. on pp. 85, 90).

[11] G. D. B. and A. N. M. "Characterizing Virtual Eigensignatures for General Purpose Face Recognition". In: *Face Recognition: From Theory to Applications, NATO ASI series F, Computer and System Science* 163 (1936), pp. 446–456. (Cit. on pp. 38, 85, 86).

[12] L. Balzano, R. Nowak, and B. Recht. "Online Identification and Tracking of Subspaces from Highly Incomplete Information". In: *arXiv:1006.4046v1* (2010). (Cit. on pp. 9, 175).

[13] P. N. Belhumeur, J. a. P. Hespanha, and D. J. Kriegman. *Eigenfaces vs. Fisherfaces: Recognition Using Class Specific Linear Projection*. 1997. (Cit. on p. 38).

[14] P. N. Belhumeur, J. a. P. Hespanha, and D. J. Kriegman. "Eigenfaces vs. Fisherfaces: Recognition Using Class Specific Linear Projection". In: *IEEE Transactions on Pattern Analysis and Machine Intelligence* 12.7 (1997), pp. 711–720. (Cit. on pp. 85, 86).

[15] M. Belkin and P. Niyogi. "Laplacian Eigenmaps and Spectral Techniques for Embedding and Clustering". In: *Advances in Neural Information Processing Systems 14*. Vol. 14. 2001, pp. 585–591. (Cit. on pp. 4, 55).

[16] M. Belkin, P. Niyogi, and V. Sindhwani. "Manifold Regularization: A Geometric Framework for Learning from Labeled and Unlabeled Examples". In: *Journal of Machine Learning Research* 7 (2006), pp. 2399–2434. (Cit. on p. 5).

[17] Y. Bengio et al. "Out-of-sample extensions for lle, isomap, mds, eigenmaps, and spectral clustering". In: *Advances in Neural Information Processing Systems 16 (NIPS '03)*. Vol. 16. 2004, pp. 177–184. (Cit. on pp. 55, 87).

[18] W. Bian and D. Tao. "Harmonic mean for subspace selection". In: *IEEE ICPR*. 2008, pp. 1–4. (Cit. on p. 4).

[19] W. Bian and D. Tao. "Max-Min Distance Analysis by Using Sequential SDP Relaxation for Dimension Reduction". In: *IEEE Transactions on Pattern Analysis and Machine Intelligence* 33.5 (2011), pp. 1037–1050. (Cit. on p. 221).

[20] N. C. Bianchi, C. Gentile, and L. Zaniboni. "Incremental Algorithms for Hierarchical Classification". In: *Journal of Machine Learning Research* 7 (2006), pp. 31–54. (Cit. on pp. 187, 201, 219, 220, 257).

[21] C. M. Bishop. *Pattern Recognition and Machine Learning (Information Science and Statistics)*. Secaucus, NJ, USA: Springer-Verlag New York, Inc., 2006. (Cit. on p. 1).

[22] C. M. Bishop and C. K. I. Williams. "GTM: The generative topographic mapping". In: *Neural Computation* 10 (1998), pp. 215–234. (Cit. on p. 4).

[23] V. Bittorf et al. "Factoring nonnegative matrices with linear programs". In: *Advances in Neural Information Processing Systems (NIPS)*. 2012. (Cit. on pp. xviii, 106, 118, 120).

[24] J. Bobin, S. Becker, and E. Candès. "NESTA: A Fast and Accurate First-order Method for Sparse Recovery". In: *technical report* (2009). (Cit. on p. 57).

[25] J. Bobin et al. "Morphological Component Analysis: An Adaptive Thresholding Strategy". In: *IEEE Transactions on Image Processing* 16.11 (2007), pp. 2675–2681. (Cit. on p. 184).

[26] M. R. Boutell et al. "Learning multi-label scene classification". In: *Pattern Recognition* 37.9 (2004), pp. 1757–1771. (Cit. on p. 265).

[27] C. Boutsidis, M. W. Mahoney, and P. Drineas. "An improved approximation algorithm for the column subset selection problem". In: *SODA*. 2009, pp. 968–977. (Cit. on pp. 4, 142).

[28] S. Boyd and L. Vandenberghe. *Convex Optimization*. New York, NY, USA: Cambridge University Press, 2004. (Cit. on p. 59).

[29] K. Bredies and D. A. Lorenz. "Iterated Hard Shrinkage for Minimization Problems with Sparsity Constraints". In: *SIAM Journal on Scientific Computing* 30.2 (2008), pp. 657–683. (Cit. on pp. 57, 149).

[30] L. Breiman and J. H. Friedman. "Predicting multivariate responses in multiple linear regression (with discussion)". In: *The Journal of the Royal Statistical Society Series B* 54 (1997), pp. 5–54. (Cit. on pp. 202, 221).

[31] D. Cai et al. "Non-negative Matrix Factorization on Manifold". In: *IEEE International Conference on Data Mining (ICDM)*. 2008. (Cit. on p. 103).

[32] D. Cai, X. He, and J. Han. "Document Clustering Using Locality Preserving Indexing". In: *IEEE Transactions on Knowledge and Data Engineering* 17.12 (2005), pp. 1624–1637. (Cit. on p. 90).

[33] D. Cai, X. He, and J. Han. "Spectral Regression for Efficient Regularized Subspace Learning". In: *Computer Vision, 2007. ICCV 2007. IEEE 11th International Conference on*. 2007, pp. 1–8. (Cit. on p. 18).

[34] D. Cai, X. He, and J. Han. "SRDA: An Efficient Algorithm for Large-Scale Discriminant Analysis". In: *IEEE Transactions on Knowledge and Date Engineering* 20.1 (2008), pp. 1–12. (Cit. on p. 18).

[35] J. Cai, E. J. Candès, and Z. Shen. "A Singular Value Thresholding Algorithm for Matrix Completion". In: *SIAM Journal on Optimization* 20.4 (2010), pp. 1956–1982. (Cit. on pp. xxiii, 8, 173, 176).

[36] E. J. Candès and T. Tao. "The Power of Convex Relaxation: Near-Optimal Matrix Completion". In: *IEEE Transactions on Information Theory* 56.5 (2009), pp. 2053–2080. (Cit. on pp. 8, 107, 173, 175, 183).

[37] E. J. Candès et al. "Robust Principal Component Analysis?" In: *Journal of the ACM (submitted)* (2009). (Cit. on pp. 7, 149, 161, 164–166, 168, 183, 184, 189, 214).

[38] E. Candès and T. Tao. "The Dantzig selector: statistical estimation when p is much larger than n". In: *Annals of Statistics* 35 (2005), pp. 2392–2404. (Cit. on p. 6).

[39] E. J. Candès and B. Recht. "Exact Matrix Completion via Convex Optimization". In: *Foundations of Computational Mathematics* 9 (2008), pp. 717–772. (Cit. on pp. 8, 107, 122, 164, 173, 183).

[40] E. J. Candès, J. K. Romberg, and T. Tao. "Robust uncertainty principles: exact signal reconstruction from highly incomplete frequency information". In: *IEEE Transactions on Information Theory* 52.2 (2006), pp. 489–509. (Cit. on pp. 6, 54, 226, 257).

[41] E. J. Candès and T. Tao. "Near-Optimal Signal Recovery From Random Projections: Universal Encoding Strategies?" In: *IEEE Transactions on Information Theory* 52.12 (2006), pp. 5406–5425. (Cit. on pp. 3, 75, 164, 183).

[42] E. J. Candès, M. B. Walkin, and S. Boyd. "Enhancing Sparsity by Reweighted L1 Minimization". In: *special issue on sparsity, Journal of Fourier Analysis and Applications* 14.5 (2008), pp. 877–905. (Cit. on pp. 52, 75, 76).

[43] V. Chandrasekaran et al. "Rank-Sparsity Incoherence for Matrix Decomposition". In: *arXiv:0906.2220* (2009). (Cit. on pp. 7, 150, 165, 166, 183).

[44] C.-C. Chang and C.-J. Lin. "LIBSVM: a library for support vector machines". In: *Software available at http://www.csie.ntu.edu.tw/ cjlin/libsvm* (2001). (Cit. on p. 273).

[45] N. V. Chawla et al. "SMOTE: Synthetic Minority Over-sampling Technique". In: *Journal of Artificial Intelligence Research* 16 (2002), pp. 321–357. (Cit. on pp. 263, 287).

[46] J. Chen, J. Liu, and J. Ye. "Learning Incoherent Sparse and Low-Rank Patterns from Multiple Tasks". In: *SIGKDD*. 2010. (Cit. on pp. 8, 166, 183, 202, 222).

[47] S. S. Chen, D. L. Donoho, and M. A. Saunders. "Atomic Decomposition by Basis Pursuit". In: *SIAM Journal on Scientific Computing* 20.1 (1999), pp. 33–61. (Cit. on p. 56).

[48] L. Cheng et al. "Real-time discriminative background subtraction". In: *to appear in IEEE Trans on Image Processing* (2010). (Cit. on p. 160).

[49] W. Cheng and E. Hüllermeier. "Combining instance-based learning and logistic regression for multilabel classification". In: *Machine Learning* 76.2-3 (2009), pp. 211–225. (Cit. on pp. 187, 221).

[50] A. Cichocki et al. *Non-negative Matrix and Tensor Factorizations: Applications to Exploratory Multi-way Data Analysis and Blind Source Separation*. Wiley, 2009. (Cit. on pp. 103, 105).

[51] A. Clare and R. D. King. "Knowledge Discovery in Multi-label Phenotype Data". In: *PKDD '01: Proceedings of the 5th European Conference on Principles of Data Mining and Knowledge Discovery*. London, UK: Springer-Verlag, 2001, pp. 42–53. (Cit. on p. 222).

[52] K. L. Clarkson. "Tighter bounds for random projections of manifolds". In: *SCG '08: Proceedings of the 24 annual symposium on Computational geometry*. 2008, pp. 39–48. (Cit. on p. 226).

[53] K. L. Clarkson and D. P. Woodruff. "Numerical linear algebra in the streaming model". In: *Proceedings of the 41st annual ACM symposium on Theory of computing*. 2009. (Cit. on pp. 4, 127, 142, 185).

[54] D. Comaniciu, V. Ramesh, and P. Meer. "Kernel-Based Object Tracking". In: *IEEE Transactions on Pattern Analysis and Machine Intelligence* 25.5 (2003), pp. 564–575. (Cit. on p. 189).

[55] P. Comon. "Independent component analysis, a new concept?" In: *Signal Processing* 36.3 (1994), pp. 287–314. (Cit. on p. 184).

[56] K. Crammer and Y. Singer. "A family of additive online algorithms for category ranking". In: *Journal of Machine Learning Research.* 3 (2003), pp. 1025–1058. ISSN: 1532-4435. (Cit. on p. 222).

[57] S. Dasgupta. "Experiments with Random Projection". In: *UAI '00: Proceedings of the 16th Conference on Uncertainty in Artificial Intelligence*. San Francisco, CA, USA, 2000, pp. 143–151. (Cit. on p. 226).

[58] S. Dasgupta and Y. Freund. "Random projection trees and low dimensional manifolds". In: *STOC '08: Proceedings of the 40th annual ACM symposium on Theory of computing*. 2008, pp. 537–546. (Cit. on p. 226).

[59] A. d'Aspremont, F. Bach, and L. E. Ghaoui. "Optimal Solutions for Sparse Principal Component Analysis". In: *Journal of Machine Learning Research* 9 (2008), pp. 1269–1294. ISSN: 1532-4435. (Cit. on pp. 58, 80, 81, 85, 91).

[60] A. d'Aspremont et al. "A Direct Formulation for Sparse PCA Using Semidefinite Programming". In: *SIAM Review* 49.3 (2007), pp. 434–448. (Cit. on pp. 18, 58, 80, 91).

[61] I. Daubechies, M. Defrise, and C. D. Mol. "An iterative thresholding algorithm for linear inverse problems with a sparsity constraint". In: *Communications on Pure and Applied Mathematics* 57.11 (2004), pp. 1413–1457. (Cit. on p. 57).

[62] J. Dean and S. Ghemawat. "MapReduce: simplified data processing on large clusters". In: *Communications of the ACM* 51.1 (2008), pp. 107–113. (Cit. on p. 108).

[63] K Dembczyński, W. Cheng, and E. Hüllermeier. "Bayes Optimal Multilabel Classification via Probabilistic Classifier Chains". In: *The 27th International Conference on Machine Learning (ICML 2010)*. 2010. (Cit. on p. 222).

[64] K. Dembczyński et al. “On Label Dependence in Multi-Label Classification”. In: *ICML 2010 Workshop on Learning from Multi-label data (MLD 10)*. 2010, pp. 5–13. (Cit. on p. 218).

[65] A. Deshpande and S. Vempala. “Adaptive sampling and fast low-rank matrix approximation”. In: *RANDOM ’06: The 10th International Workshop on Randomization and Computation*. 2006, pp. 292–303. (Cit. on p. 126).

[66] T. G. Dietterich and G. Bakiri. “Solving multiclass learning problems via error-correcting output codes”. In: *Journal of Artificial Intelligence Research* 2 (1995), pp. 263–282. (Cit. on p. 259).

[67] C. H. Q. Ding, T. Li, and M. I. Jordan. “Convex and Semi-Nonnegative Matrix Factorizations”. In: *IEEE Transactions on Pattern Analysis and Machine Intelligence* 32.1 (2008), pp. 45–55. (Cit. on p. 103).

[68] C. Ding and T. Li. “Adaptive dimension reduction using discriminant analysis and K-means clustering”. In: *ICML ’07: Proceedings of the 24th international conference on Machine learning*. Corvalis, Oregon: ACM, 2007, pp. 521–528. (Cit. on p. 17).

[69] S. Diplaris et al. “Protein Classification with Multiple Algorithms”. In: *Proceedings of the 10th Panhellenic Conference on Informatics (PCI 2005)*. 2005, pp. 448–456. (Cit. on p. 265).

[70] D. Donoho and V. Stodden. “When does non-negative matrix factorization give a correct decomposition into parts?” In: *Advances in Neural Information Processing Systems (NIPS)*. 2003. (Cit. on p. 104).

[71] D. L. Donoho. "Compressed sensing". In: *IEEE Transactions on Information Theory* 52.4 (2006), pp. 1289–1306. (Cit. on pp. 3, 6, 9, 54, 149, 164, 183, 222, 257).

[72] D. L. Donoho and C. Grimes. "Hessian eigenmaps: Locally linear embedding techniques for high-dimensional data". In: *PNAS* 100.10 (2003), pp. 5591–5596. (Cit. on pp. 4, 55).

[73] D. L. Donoho, A. Maleki, and A. Montanari. "Message Passing Algorithms for Compressed Sensing". In: *Proceedings of the National Academy of Sciences* (2009). (Cit. on p. 57).

[74] A. Doucet, N. de Freitas, and N. Gordon. *Sequential Monte Carlo Methods in Practice*. New York: Springer-Verlag, 2001. (Cit. on p. 189).

[75] P. Drineas, R. Kannan, and M. W. Mahoney. "Fast Monte Carlo Algorithms for Matrices III: Computing a Compressed Approximate Matrix Decomposition". In: *SIAM Journal on Computing* 36.1 (2006), pp. 184–206. (Cit. on pp. 4, 106, 123, 142).

[76] P. Duygulu et al. "Object Recognition as Machine Translation: Learning a Lexicon for a Fixed Image Vocabulary". In: *ECCV '02: Proceedings of the 7th European Conference on Computer Vision-Part IV*. London, UK: Springer-Verlag, 2002, pp. 97–112. ISBN: 3-540-43748-7. (Cit. on p. 265).

[77] B. Efron et al. "Least angle Regression". In: *Annals of Statistics* 32.2 (2004), pp. 407–499. (Cit. on pp. 6, 16, 18, 80, 283).

[78] E. Elhamifar and R. Vidal. "Sparse subspace clustering". In: *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*. 2009. (Cit. on p. 106).

[79] S. Escalera, O. Pujol, and P. Radeva. "On the Decoding Process in Ternary Error-Correcting Output Codes". In: *IEEE Transactions on Pattern Analysis and Machine Intelligence* 32.1 (2010). (Cit. on p. 259).

[80] E. Esser et al. "A Convex Model for Nonnegative Matrix Factorization and Dimensionality Reduction on Physical Space". In: *IEEE Transactions on Image Processing* 21.7 (2012), pp. 3239–3252. (Cit. on p. 103).

[81] T. Evgeniou and M. Pontil. "Regularized multi–task learning". In: *KDD 2004*. 2004, pp. 109–117. (Cit. on p. 221).

[82] J. Fan and Y. Fan. "High dimensional classification using features annealed independence rules". In: *The Annals of Statistics* 36 (2008), pp. 2605–2637. (Cit. on p. 54).

[83] J. Fan and J. Lv. "A selective overview of variable selection in high dimensional feature space (Editor's invited paper)". In: *invited review article* 20 (2010), pp. 101–148. (Cit. on p. 58).

[84] J. Fan and J. Lv. "Sure independence screening for ultrahigh dimensional feature space". In: *Journal Of The Royal Statistical Society Series B* 70.5 (2008), pp. 849–911. (Cit. on p. 6).

[85] M. Fazel et al. "Compressed sensing and robust recovery of low rank matrices". In: *42nd Asilomar Conference on Signals, Systems and Computers*. 2008. (Cit. on pp. 127, 143).

[86] M. A. T. Figueiredo, R. D. Nowak, and S. J. Wright. "Gradient projection for sparse reconstruction: Application to compressed sensing and other inverse problems". In: *IEEE Journal of Selected Topics in Signal Processing* 1.4 (2007), pp. 586–597. (Cit. on p. 57).

[87] R. A. Fisher. "The use of multiple measurements in taxonomic problems". In: *Annals of Eugenics* 7.2 (1936), pp. 179–188. (Cit. on pp. 4, 54, 86, 183).

[88] M. Fornasier, H. Rauhut, and R. Ward. "Low-rank Matrix Recovery via Iteratively Reweighted Least Squares Minimization". In: *SIAM Journal on Optimization* 21.4 (2011), pp. 1614–1640. (Cit. on p. 164).

[89] R. Foygel and N. Srebro. "Concentration-Based Guarantees for Low-Rank Matrix Reconstruction". In: *COLT*. 2011. (Cit. on pp. 8, 174).

[90] K. Fragkiadaki and J. Shi. "Detection free tracking: Exploiting motion and topology for segmenting and tracking under entanglement". In: *IEEE Conference on Computer Vision and Pattern Recognition*. 2011, pp. 2073–2080. (Cit. on pp. 186, 188).

[91] J. Fürnkranz et al. "Multilabel classification via calibrated label ranking". In: *Machine Learning* 73.2 (2008), pp. 133–153. ISSN: 0885-6125. (Cit. on p. 220).

[92] C. Fyfe. "Two topographic maps for data visualisation". In: *Data Minining and Knowledge Discovery* 14.2 (2007), pp. 207–224. (Cit. on p. 4).

[93] F. Galasso, R. Cipolla, and B. Schiele. "Video Segmentation with Superpixels". In: *Asian Conference on Computer Vision*. 2012. (Cit. on p. 188).

[94] Y. Ge et al. "Collaborative Filtering with Collective Training". In: *ACM International Conference on Recommender Systems*. 2011. (Cit. on p. 105).

[95] N. Ghamrawi and A. McCallum. "Collective multi-label classification". In: *CIKM '05: Proceedings of the 14th ACM international conference on Information and knowledge management*. Bremen, Germany, 2005, pp. 195–200. (Cit. on p. 221).

[96] N. Gillis and R. Luce. "Robust Near-Separable Nonnegative Matrix Factorization Using Linear Optimization". In: *arXiv:1302.4385*. 2013. (Cit. on p. 106).

[97] N. Gillis and S. Vavasis. "Fast and robust recursive algorithms for separable nonnegative matrix factorization". In: *arXiv:1208.1237*. 2012. (Cit. on pp. xvii, 107, 119).

[98] M. X. Goemans and D. P. Williamson. "Improved approximation algorithms for maximum cut and satisfiability problems using semidefinite programming". In: *Journal of the ACM* 42.6 (1995), pp. 1115–1145. (Cit. on pp. 229, 233).

[99] G. H. Golub and C. F. Van Loan. *Matrix computations (3rd ed.)* Baltimore, MD, USA: Johns Hopkins University Press, 1996. (Cit. on pp. 32, 75).

[100] J. Gomez, E. Boiy, and M.-F. Moens. "Highly discriminative statistical features for email classification". In: *Knowledge and Information Systems* (2011), pp. 1–31. (Cit. on p. 219).

[101] A. Gretton et al. "Measuring statistical dependence with Hilbert-Schmidt norms". In: *Proceedings Algorithmic Learning Theory*. Springer-Verlag, 2005, pp. 63–77. (Cit. on p. 221).

[102] M. Gu and S. C. Eisenstat. "Efficient algorithms for computing a strong rank-revealing QR factorization". In: *SIAM Journal on Scientific Computing* 17.4 (1996), pp. 848–869. (Cit. on p. 106).

[103] N. Guan et al. "MahNMF: Manhattan Non-negative Matrix Factorization". In: *CoRR* abs/1207.3438 (2012). (Cit. on p. 103).

[104] N. Guan et al. "NeNMF: An Optimal Gradient Method for Nonnegative Matrix Factorization". In: *IEEE Transactions on Signal Processing* 60.6 (2012), pp. 2882–2898. (Cit. on p. 103).

[105] N. Guan et al. "Non-Negative Patch Alignment Framework". In: *IEEE Transactions on Neural Networks* 22.8 (2011), pp. 1218–1230. (Cit. on p. 263).

[106] A. Gupta, R. Nowak, and B. Recht. "Sample Complexity for 1-bit Compressed Sensing and Sparse Classification". In: *Proceedings of the IEEE International Symposium on Information Theory (ISIT)*. 2010. (Cit. on p. 258).

[107] E. T. Hale, W. Yin, and Y. Zhang. "Fixed-Point Continuation for ℓ_1-Minimization: Methodology and Convergence". In: *SIAM Journal on Optimization* 19.3 (2008), pp. 1107–1130. ISSN: 1052-6234. (Cit. on p. 57).

[108] N. Halko, P. G. Martinsson, and J. A. Tropp. "Finding structure with randomness: Stochastic algorithms for constructing approximate matrix

decompositions". In: *arXiv: 0909.4061* (2009). (Cit. on pp. 4, 126, 127, 132, 134, 135, 138, 142, 144, 146, 149, 164, 168, 178, 185, 226, 233).

[109] T. Hastie, R. Tibshirani, and J. Friedman. *The Elements of Statistical Learning: Data Mining, Inference, and Prediction, Second Edition*. 2nd ed. 2009. Corr. 3rd printing. Springer Series in Statistics. Springer, 2009. (Cit. on pp. 1, 3, 17, 86, 89, 222).

[110] X. He and P. Niyogi. "Locality Preserving Projections". In: *In Advances in Neural Information Processing Systems 16*. MIT Press, 2003. (Cit. on p. 5).

[111] X. He et al. "Face Recognition Using Laplacianfaces". In: *IEEE Trans. Pattern Anal. Mach. Intell.* 27.3 (2005), pp. 328–340. (Cit. on p. 43).

[112] X. He et al. "Neighborhood preserving embedding". In: *Proceedings of IEEE International Conference on Computer Vision*. Vol. 2. 2005, pp. 1208–1213. (Cit. on pp. 5, 55, 86).

[113] R. Hess and A. Fern. "Discriminatively trained particle filters for complex multi-object tracking". In: *IEEE Conference on Computer Vision and Pattern Recognition*. 2009. (Cit. on pp. 186, 189).

[114] T. Hofmann. "Probabilistic latent semantic indexing". In: *Proceedings of the 22nd Annual International SIGIR Conference on Research and Development in Information Retrieval (SIGIR)*. 1999. (Cit. on p. 105).

[115] S. C. H. Hoi et al. "Batch mode active learning and its application to medical image classification". In: *ICML*. 2006, pp. 417–424. (Cit. on p. 1).

[116] S. C. H. Hoi et al. "FANS: face annotation by searching large-scale web facial images". In: *WWW*. 2013, pp. 317–320. (Cit. on p. 17).

[117] S. C. H. Hoi et al. "Online Multiple Kernel Classification". In: *Machine Learning* 90.2 (2013), pp. 289–316. (Cit. on p. 14).

[118] Z. Hong, X. Mei, and D. Tao. "Dual-Force Metric Learning for Robust Distracter-Resistant Tracker". In: *European Conference on Computer Vision*. 2012. (Cit. on p. 189).

[119] H Hotelling. "Analysis of A Complex of Statistical Variables into Principal Components". In: *Journal of Educational Phychology* 24 (1936), pp. 417–441. (Cit. on pp. 4, 54, 56, 86, 107, 123, 183).

[120] C. J. Hsieh and I. S. Dhillon. "Fast coordinate descent methods with variable selection for nonnegative matrix factorization". In: *Proceedings of the 17th ACM SIGKDD international conference on Knowledge discovery and data mining (KDD)*. 2011. (Cit. on p. 103).

[121] D. Hsu, S. Kakade, and T. Zhang. "Robust Matrix Decomposition with Sparse Corruptions". In: *IEEE Transactions on Information Theory* (2011). (Cit. on pp. 183, 185, 189).

[122] D. Hsu et al. "Multi-Label Prediction via Compressed Sensing". In: *Advances in Neural Information Processing Systems 23*. 2009. (Cit. on pp. 9, 200, 202, 222, 240, 257, 262, 263, 283).

[123] H. Huang and C. Ding. "Robust tensor factorization using R1 norm". In: *Computer Vision and Pattern Recognition, 2008. CVPR 2008. IEEE Conference on*. 2008, pp. 1–8. (Cit. on p. 6).

[124] J. Huang, T. Zhang, and D. Metaxas. "Learning with structured sparsity". In: *ICML '09: Proceedings of the 26th Annual International Conference on Machine Learning*. Montreal, Quebec, Canada: ACM, 2009, pp. 417–424. ISBN: 978-1-60558-516-1. (Cit. on pp. 6, 75, 77).

[125] E. Hüllermeier et al. "Label ranking by learning pairwise preferences". In: *Artificial Intelligence* 172.16-17 (2008), pp. 1897–1916. (Cit. on p. 220).

[126] A. Hyvärinen and E. Oja. "Independent component analysis: algorithms and applications". In: *Neural Networks* 13.4-5 (2000), pp. 411–430. (Cit. on p. 184).

[127] P. Indyk and R. Motwani. "Approximate nearest neighbors: towards removing the curse of dimensionality". In: *STOC '98: Proceedings of the thirtieth annual ACM symposium on Theory of computing*. 1998, pp. 604–613. (Cit. on p. 226).

[128] F. J. and L. R. "Variable Selection via Nonconcave Penalized Likelihood and its Oracle Properties". In: *Journal of the American Statistical Association* 96 (2001), pp. 1348–1360. (Cit. on pp. 6, 52).

[129] P. Jain, R. Meka, and I. S. Dhillon. "Guaranteed Rank Minimization via Singular Value Projection". In: *NIPS*. 2010. (Cit. on pp. 9, 174, 183).

[130] G. M. James, P. Radchenko, and J. Lv. "DASSO: connections between the Dantzig selector and lasso". In: *Journal Of The Royal Statistical Society Series B* 71.1 (2009), pp. 127–142. (Cit. on p. 6).

[131] R. Jenatton, J.-Y. Audibert, and F. Bach. *Structured Variable Selection with Sparsity-Inducing Norms*. 2009. URL: http://www.citebase.org/abstract?id=oai:arXiv.org:0904.3523. (Cit. on pp. 6, 75, 77).

[132] S. Ji and J. Ye. "An accelerated gradient method for trace norm minimization". In: *International Conference on Machine Learning (ICML)*. 2009. (Cit. on pp. 8, 107, 173, 183, 222).

[133] S. Ji and J. Ye. "Linear dimensionality reduction for multi-label classification". In: *IJCAI*. 2009. (Cit. on p. 202).

[134] S. Ji et al. "A Shared-subspace Learning Framework for Multi-label Classification". In: *ACM Trans on Knowledge Discovery from Data* 2.1 (2010). (Cit. on pp. 187, 202, 221).

[135] W. Johnson and J. Lindenstrauss. "Extensions of Lipschitz mappings into a Hilbert space". In: *Conference in modern analysis and probability (New Haven, Conn., 1982)*. Vol. 26. Contemporary Mathematics. American Mathematical Society, 1984, pp. 189–206. (Cit. on p. 226).

[136] I. Katakis, G. Tsoumakas, and I. Vlahavas. "Multilabel Text Classification for Automated Tag Suggestion". In: *Proceedings of the ECML/PKDD 2008 Discovery Challenge*. 2008. (Cit. on p. 265).

[137] R. Keshavan and S. Oh. "OptSpace: A Gradient Descent Algorithm on Grassman Manifold for Matrix Completion". In: *Submitted to IEEE Trans on Signal Processing* (2009). (Cit. on pp. xxiii, 9, 149, 174–176, 183).

[138] J. Kim and H. Park. "Toward Faster Nonnegative Matrix Factorization: A New Algorithm and Comparisons". In: *IEEE International Conference on Data Mining (ICDM)*. 2008. (Cit. on p. 103).

[139] S.-J. Kim et al. "An Interior-Point Method for Large-Scale L1-Regularized Least Squares". In: *IEEE Journal of In Selected Topics in Signal Processing* 1.4 (2007), pp. 606–617. (Cit. on p. 57).

[140] X. Kong and P. Yu. "gMLC: a multi-label feature selection framework for graph classification". In: *Knowledge and Information Systems* (2011), pp. 1–25. (Cit. on p. 221).

[141] A. Koufakou, J. Secretan, and M. Georgiopoulos. "Non-derivable itemsets for fast outlier detection in large high-dimensional categorical data". In: *Knowledge and Information Systems* 29 (3 2011), pp. 697–725. (Cit. on p. 257).

[142] K. Kreutz-Delgado et al. "Dictionary Learning Algorithms for Sparse Representation". In: *Neural Computation* 15.2 (2003), pp. 349–396. (Cit. on pp. 6, 81, 183).

[143] H.-P. Kriegel et al. "Future trends in data mining". In: *Data Min. Knowl. Discov.* 15.1 (2007), pp. 87–97. (Cit. on pp. 3, 17).

[144] S. Kullback and R. A. Leibler. "On Information and Sufficiency". In: *The Annals of Mathematical Statistics* 22.1 (1951), pp. 79–86. (Cit. on p. 234).

[145] A. Kumar, V. Sindhwani, and P. Kambadur. "Fast conical hull algorithms for near-separable nonnegative matrix factorization". In: *International*

Conference on Machine Learning (ICML). 2013. (Cit. on pp. xvii, 107, 119).

[146] J. Langford and A. Beygelzimer. "Sensitive Error Correcting Output Codes." In: *COLT'05: Annual Conference on Learning Theory*. Vol. 3559. 2005, pp. 158–172. (Cit. on p. 262).

[147] Y. Lecun and C. Cortes. "The MNIST database of handwritten digits". In: *Available: http://yann.lecun.com/exdb/mnist/* (). (Cit. on p. 86).

[148] D. D. Lee and H. S. Seung. "Algorithms for non-negative matrix factorization". In: *Advances in Neural Information Processing Systems (NIPS)*. 2001. (Cit. on p. 103).

[149] D. D. Lee and H. S. Seung. "Learning the parts of objects by non-negative matrix factorization". In: *Nature* 401 (1999), pp. 788–791. (Cit. on pp. 17, 103).

[150] H. Lee et al. "Efficient sparse coding algorithms". In: *Advances in Neural Information Processing Systems 18 (NIPS '06)*. 2006. (Cit. on pp. 6, 81, 183).

[151] J. Lee et al. "Practical Large-Scale Optimization for Max-Norm Regularization". In: *NIPS*. 2010. (Cit. on pp. 8, 174).

[152] K. Lee and Y. Bresler. "ADMiRA: atomic decomposition for minimum rank approximation". In: *IEEE Trans on Information Theory* 56.9 (2010), pp. 4402–4416. (Cit. on pp. 8, 174).

[153] A. S. Lewis and J. Malick. "Alternating Projections on Manifolds". In: *Mathematics of Operations Research* 33.1 (2008), pp. 216–234. (Cit. on pp. 150, 153, 155).

[154] P. Li. "Approximating Higher-Order Distances Using Random Projections". In: *The 26th Conference on Uncertainty in Artificial Intelligence (UAI 2010)*. 2010. (Cit. on p. 226).

[155] P. Li. "Estimators and tail bounds for dimension reduction in $\ell_\alpha(0 < \alpha \leq 2)$ using stable random projections". In: *SODA '08: Proceedings of the nineteenth annual ACM-SIAM symposium on Discrete algorithms*. San Francisco, California: Society for Industrial and Applied Mathematics, 2008, pp. 10–19. (Cit. on p. 226).

[156] T. Li, S. Zhu, and M. Ogihara. "Text categorization via generalized discriminant analysis". In: *Inf. Process. Manage.* 44.5 (2008), pp. 1684–1697. (Cit. on p. 17).

[157] X. Li et al. "Discriminant Locally Linear Embedding With High-Order Tensor Data". In: *IEEE Transactions on Systems, Man, and Cybernetics, Part B* 38.2 (2008), pp. 342–352. (Cit. on p. 4).

[158] C.-J. Lin. "Projected Gradient Methods for Nonnegative Matrix Factorization". In: *Neural Computation* 19.10 (2007), pp. 2756–2779. (Cit. on p. 103).

[159] J. Liu, S. Ji, and J. Ye. "Multi-task feature learning via efficient l2,1-norm minimization". In: *UAI*. 2009. (Cit. on p. 6).

[160] J. Liu, J. Chen, and J. Ye. "Large-scale sparse logistic regression". In: *KDD '09: Proceedings of the 15th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*. 2009, pp. 547–556. (Cit. on pp. 6, 54).

[161] J. Liu, S. Ji, and J. Ye. *SLEP: Sparse Learning with Efficient Projections*. Arizona State University. 2009. URL: http://www.public.asu.edu/~jye02/Software/SLEP. (Cit. on pp. 6, 54, 207, 208, 212).

[162] L. Liu and Q. Liang. "A high-performing comprehensive learning algorithm for text classification without pre-labeled training set". In: *Knowledge and Information Systems* 29 (3 2011), pp. 727–738. (Cit. on p. 219).

[163] Q. Liu et al. "Personalized Travel Package Recommendation". In: *IEEE International Conference on Data Mining (ICDM)*. 2011. (Cit. on p. 114).

[164] W. Liu, D. Tao, and J. Liu. "Transductive Component Analysis". In: *ICDM '08: Proceedings of the 2008 Eighth IEEE International Conference on Data Mining*. Washington, DC, USA: IEEE Computer Society, 2008, pp. 433–442. (Cit. on p. 5).

[165] U. Luxburg. "A tutorial on spectral clustering". In: *Statistics and Computing* 17.4 (2007), pp. 395–416. (Cit. on p. 236).

[166] J. Lv and Y Fan. "A unified approach to model selection and sparse recovery using regularized least squares". In: *The Annals of Statistics* 37 (2009), pp. 3498–3528. (Cit. on pp. 6, 25, 54).

[167] J. Lv and J. S. Liu. "Model selection principles in misspecified models". In: *Manuscript* (2010). (Cit. on p. 58).

[168] S. Ma, D. Goldfarb, and L. Chen. "Fixed point and Bregman iterative methods for matrix rank minimization". In: *Mathematical Programming* 128.1-2 (2011), pp. 321–353. (Cit. on pp. 8, 174).

[169] L. Mackey. "Deflation Methods for Sparse PCA". In: *Advances in Neural Information Processing Systems 22 (NIPS '08)*. 2008. (Cit. on p. 71).

[170] L. W. Mackey, A. S. Talwalkar, and M. I. Jordan. "Divide-and-Conquer Matrix Factorization". In: *Advances in Neural Information Processing Systems 24*. 2011, pp. 1134–1142. (Cit. on pp. 9, 109, 175).

[171] J. B. MacQueen. "Some Methods for Classification and Analysis of Multi-Variate Observations". In: *Proceeding of the fifth Berkeley Symposium on Mathematical Statistics and Probability*. Vol. 1. 1967, pp. 281–297. (Cit. on p. 236).

[172] J. Mairal, M. Elad, and G. Sapiro. "Sparse Representation for Color Image Restoration". In: *IEEE Transactions on Image Processing* 17.1 (2008), pp. 53–69. (Cit. on p. 54).

[173] J. Mairal et al. "Online Learning for Matrix Factorization and Sparse Coding". In: *Journal of Machine Learning Research* 11 (2010), pp. 10–60. (Cit. on pp. 80, 81).

[174] M. Masud et al. "Facing the reality of data stream classification: coping with scarcity of labeled data". In: *Knowledge and Information Systems* (2011), pp. 1–32. (Cit. on p. 223).

[175] L. Mencía and J. Fürnkranz. "Pairwise learning of multilabel classifications with perceptrons". In: *IEEE International Joint Conference on Neural Networks (IJCNN-08),* vol. 0. 2008, pp. 995–1000. (Cit. on p. 220).

[176] B. Moghaddam, Y. Weiss, and S. Avidan. "Spectral Bounds for Sparse PCA: Exact and Greedy Algorithms". In: *Advances in Neural Information Processing Systems 20*. 2006, pp. 915–922. (Cit. on pp. xxii, 58, 80, 85, 91, 92, 97).

[177] R. J. Muirhead. *Aspects of multivariate statistical theory*. New York: John Wiley & Sons Inc., 1982. (Cit. on p. 136).

[178] D. Needell and J. A. Tropp. "CoSaMP: Iterative signal recovery from incomplete and inaccurate samples". In: *Applied and Computational Harmonic Analysis* 26 (2008), pp. 301–321. (Cit. on p. 56).

[179] A. Nemirovski. *Lecture Notes: Introduction to Linear Optimization*. 2010. (Cit. on p. 105).

[180] S. Nene, S. K. Nayar, and H. Murase. *Columbia Object Image Library (COIL-20)*. Tech. rep. 1996. (Cit. on pp. 85, 88).

[181] Y. Nesterov. "Smooth minimization of non-smooth functions". In: *Mathematical Programming* 103.1 (2005), pp. 127–152. ISSN: 0025-5610. (Cit. on p. 57).

[182] A. Y. Ng, M. I. Jordan, and Y. Weiss. "On spectral clustering: Analysis and an algorithm". In: *NIPS '01: Advances in Neural Information Processing Systems 14*. Vol. 2. 2001, pp. 849–856. (Cit. on pp. 55, 236).

[183] N. H. Nguyen, T. T. Do, and T. D. Tran. "A fast and efficient algorithm for low-rank approximation of a matrix". In: *STOC '09: The 41st annual ACM symposium on Theory of computing*. 2009, pp. 215–224. (Cit. on pp. 4, 142).

[184] E. Osuna, R. Freund, and F. Girosi. *Support Vector Machines: Training and Applications*. Tech. rep. Massachusetts Institute of Technology, 1997. (Cit. on pp. 263, 287).

[185] M. Y. Park and T. Hastie. "L1-regularization pathalgorithm for generalized linear models". In: *Journal of the Royal Statistical Society, Series B* 69.4 (2007), pp. 659–677. (Cit. on pp. 6, 84).

[186] K. B. Petersen and M. S. Pedersen. *The Matrix Cookbook*. Version 20081110. 2008. URL: http://www2.imm.dtu.dk/pubdb/p.php?3274. (Cit. on p. 74).

[187] J. Petterson and T. Caetano. "Reverse Multi-Label Learning". In: *NIPS*. 2010. (Cit. on pp. 187, 200).

[188] P. J. Phillips et al. "The FERET evaluation methodology for face-recognition algorithms". In: *Pattern Analysis and Machine Intelligence, IEEE Transactions on* 22.10 (2000), pp. 1090–1104. (Cit. on pp. 38, 85, 86, 141).

[189] S. J. Prince. *Computer vision: models, learning and inference*. Cambridge University Press, 2011. (Cit. on p. 192).

[190] M. Raginsky and S. Lazebnik. "Locality-sensitive binary codes from shift-invariant kernels". In: *The 23rd Annual Conference on Neural Information Processing Systems (NIPS 2009)*. 2009. (Cit. on p. 233).

[191] B. Rao et al. "Subset selection in noise based on diversity measure minimization". In: *IEEE Transactions on Signal Processing* 51 (2003), pp. 760–770. (Cit. on p. 57).

[192] J. Read, B. Pfahringer, and G. Holmes. "Multi-label Classification Using Ensembles of Pruned Sets". In: *ICDM*. Vol. 0. 2008, pp. 995–1000. (Cit. on p. 220).

[193] J. Read et al. "Classifier Chains for Multi-label Classification". In: *Machine Learning and Knowledge Discovery in Databases* (2009), pp. 254–269. (Cit. on pp. 187, 201, 219, 221, 222).

[194] J. Read. *MEKA Softwares*. 2010. URL: http://meka.sourceforge.net. (Cit. on p. 265).

[195] S. Roweis. "EM algorithms for PCA and SPCA". In: *NIPS*. 1998, pp. 626–632. (Cit. on pp. 127, 128, 144).

[196] S. T. Roweis and L. K. Saul. "Nonlinear Dimensionality Reduction by Locally Linear Embedding". In: *Science* 290.5500 (2000), pp. 2323–2326. (Cit. on pp. 4, 54, 55, 87, 88, 107, 183).

[197] F. S. Samaria, A. Harter, and O. A. Site. "Parameterisation of a Stochastic Model for Human Face Identification". In: *Proceedings of the Second IEEE Workshop on Applications of Computer Vision* (1994). (Cit. on pp. 85, 86).

[198] R. E. Schapire and Y. Singer. "Boostexter: a boosting-based system for text categorization". In: *Machine Learning* 39.2/3 (2000), pp. 135–168. (Cit. on p. 222).

[199] G. Shakhnarovich and B. Moghaddam. *Face Recognition in Subspaces*. 2004. (Cit. on p. 19).

[200] S. Shalev-Shwartz, A. Gonen, and O. Shamir. "Large-Scale Convex Minimization with a Low-Rank Constraint." In: *ICML*. 2011. (Cit. on pp. 8, 168–170, 174).

[201] H. Shen and J. Z. Huang. "Sparse principal component analysis via regularized low rank matrix approximation". In: *Journal of Multivariate Analysis* 99.6 (2008), pp. 1015–1034. (Cit. on pp. xxii, 57, 80, 85, 91, 92, 97).

[202] S. Si, D. Tao, and B. Geng. "Bregman Divergence-Based Regularization for Transfer Subspace Learning". In: *IEEE Transactions on Knowledge and Data Engineering* 22.7 (2010), pp. 929–942. (Cit. on p. 263).

[203] C. G. M. Snoek et al. "The challenge problem for automated detection of 101 semantic concepts in multimedia". In: *MULTIMEDIA '06: Proceedings of the 14th annual ACM international conference on Multimedia*. New York, NY, USA: ACM, 2006, pp. 421–430. (Cit. on p. 265).

[204] N. Srebro, J. Rennie, and T. Jaakkola. "Maximum-Margin Matrix Factorization". In: *NIPS*. 2005. (Cit. on pp. 8, 174).

[205] L. Sun, S. Ji, and J. Ye. "A least squares formulation for canonical correlation analysis". In: *ICML '08: Proceedings of the 25th international*

conference on Machine learning. Helsinki, Finland: ACM, 2008, pp. 1024–1031. (Cit. on p. 24).

[206] L. Sun et al. "Efficient Recovery of Jointly Sparse Vectors". In: *Advances in Neural Information Processing Systems 23*. 2009. (Cit. on pp. 6, 54).

[207] D. Tao et al. "General Tensor Discriminant Analysis and Gabor Features for Gait Recognition". In: *IEEE Trans. Pattern Anal. Mach. Intell.* 29.10 (2007), pp. 1700–1715. (Cit. on pp. 3, 17).

[208] D. Tao et al. "Geometric Mean for Subspace Selection". In: *IEEE Transactions on Pattern Analysis and Machine Intelligence* 31.2 (2009), pp. 260–274. (Cit. on pp. 4, 283).

[209] D. Tao et al. "Supervised tensor learning". In: *Knowl. Inf. Syst.* 13.1 (2007), pp. 1–42. (Cit. on pp. 3, 17).

[210] J. B. Tenenbaum. "A Global Geometric Framework for Nonlinear Dimensionality Reduction". In: *Science* 290.5500 (2000), pp. 2319–2323. (Cit. on pp. 4, 54, 55, 85, 87, 107, 183).

[211] R. Tibshirani. "Regression Shrinkage and Selection via the Lasso". In: *Journal of the Royal Statistical Society (Series B)* 58 (1996), pp. 267–288. (Cit. on pp. 6, 54, 56).

[212] N. Trendafilov, I. T. Jolliffe, and M. Uddin. "A modified principal component technique based on the LASSO". In: *Journal of Computational and Graphical Statistics* 12 (2003), 531–C547. (Cit. on pp. 85, 92).

[213] K. Trohidis et al. “Multilabel Classification of Music into Emotions”. In: *Proc. 9th International Conference on Music Information Retrieval (ISMIR 2008), Philadelphia, PA, USA, 2008*. Philadephia, PA, USA, 2008. (Cit. on p. 265).

[214] J. A. Tropp and A. C. Gilbert. “Signal recovery from random measurements via Orthogonal Matching Pursuit”. In: *IEEE Transactions on Information Theory* 53 (2007), pp. 4655–4666. (Cit. on p. 56).

[215] P. Tseng and S. Yun. “A coordinate gradient descent method for nonsmooth separable minimization”. In: *Mathematical Programming* 117.1-2 (2009), pp. 387–423. (Cit. on p. 57).

[216] G. Tsoumakas and I. Katakis. “Multi-Label Classification: An Overview”. In: *International Journal of Data Warehousing and Mining* 3.3 (2007), pp. 1–13. (Cit. on pp. xxiv, 187, 200, 209–211, 217, 220, 256).

[217] G. Tsoumakas, I. Katakis, and I. Vlahavas. “Effective and Efficient Multilabel Classification in Domains with Large Number of Labels”. In: *ECML/PKDD Workshop on Mining Multidimensional Data*. 2008. (Cit. on pp. 187, 201, 219, 220, 257).

[218] G. Tsoumakas, I. Katakis, and I. Vlahavas. “Mining Multi-label Data”. In: *Data Mining and Knowledge Discovery Handbook* (2010). (Cit. on pp. 187, 209, 220).

[219] G. Tsoumakas and I. Vlahavas. “Random k-Labelsets: An Ensemble Method for Multilabel Classification”. In: *ECML*. 2007, pp. 406–417. (Cit. on pp. 187, 201, 220).

[220] G. Tsoumakas, M.-L. Zhang, and Z.-H. Zhou. "Learning from multi-label data". In: *ECML/PKDD*. 2009. (Cit. on pp. 187, 200).

[221] G. Tsoumakas. *Mulan: A Java Library for Multi-Label Learning*. 2010. URL: http://mulan.sourceforge.net/datasets.html. (Cit. on p. 265).

[222] M. A. Turk and A. P. Pentland. "Face recognition using eigenfaces". In: *Computer Vision and Pattern Recognition, 1991. Proceedings CVPR '91., IEEE Computer Society Conference on*. 1991, pp. 586–591. (Cit. on p. 43).

[223] N. Ueda and K. Saito. "Parametric mixture models for multi-labeled text". In: *Advances in Neural Information Processing Systems 15, Neural Information Processing Systems, NIPS 2002*. 2002. (Cit. on p. 265).

[224] V. N. Vapnik. *The nature of statistical learning theory*. Springer-Verlag New York, Inc., 1995. ISBN: 0387945598. (Cit. on p. 234).

[225] S. A. Vavasis. "On the complexity of nonnegative matrix factorization". In: *SIAM Joural on Optimization* 20.3 (2009), pp. 1364–1377. (Cit. on p. 104).

[226] S. S. Vempala. *The Random Projection Method*. Vol. 65. DIMACS Series in Discrete Mathematics and Theoretical Computer Science. American Mathematical Society, 2004. (Cit. on pp. 107, 226, 233).

[227] F. Wang et al. "Semi-supervised metric learning by maximizing constraint margin". In: *CIKM '08: Proceeding of the 17th ACM conference on Information and knowledge management*. Napa Valley, California, USA: ACM, 2008, pp. 1457–1458. (Cit. on p. 5).

[228] Z. Wen, W. Yin, and Y. Zhang. "Solving a low-rank factorization model for matrix completion by a nonlinear successive over-relaxation algorithm". In: *Mathematical Programming Computation* 4.4 (2012), pp. 333–361. (Cit. on p. 165).

[229] D. M. Witten, R. Tibshirani, and T. Hastie. "A penalized matrix decomposition, with applications to sparse principal components and canonical correlation analysis". In: *Biostatistics* 10.3 (2009), pp. 515–534. (Cit. on pp. 58, 80, 85, 91).

[230] J. Wright et al. "Robust Face Recognition via Sparse Representation". In: *IEEE Transactions on Pattern Analysis and Machine Intelligence* 31.2 (2009), pp. 210–227. (Cit. on p. 6).

[231] S. Wu, O. Oreifej, and M. Shah. "Action recognition in videos acquired by a moving camera using motion decomposition of Lagrangian particle trajectories". In: *International Conference on Computer Vision*. 2011, pp. 1419–1426. (Cit. on pp. 186, 188).

[232] Y. Xin and T. Jaakkola. "Primal-Dual methods for sparse constrained matrix completion". In: *Journal of Machine Learning Research - Proceedings Track* 22 (2012), pp. 1323–1331. (Cit. on p. 107).

[233] L. Xiong, X. Chen, and J. Schneider. "Direct Robust Matrix Factorization for Anomaly Detection". In: *The 11th International Conference on Data Mining (ICDM)*. 2010. (Cit. on pp. 166, 185).

[234] W. Xu, X. Liu, and Y. Gong. "Document clustering based on non-negative matrix factorization". In: *SIGIR '03: ACM SIGIR conference on Re-*

search and development in informaion retrieval. 2003, pp. 267–273. (Cit. on p. 90).

[235] S. Yan et al. "Graph Embedding and Extensions: A General Framework for Dimensionality Reduction". In: *IEEE Transactions on Pattern Analysis and Machine Intelligence* 29.1 (2007), pp. 40–51. (Cit. on p. 5).

[236] C. Yang, R. Duraiswami, and L. S. Davis. "Fast multiple object tracking via a hierarchical particle filter". In: *International Conference on Computer Vision*. 2005, pp. 212–219. (Cit. on p. 186).

[237] F. Yates. "Contingency tables involving small numbers and the χ^2 test". In: *Journal of the Royal Statistical Society* 1 (1934), pp. 217–235. (Cit. on p. 264).

[238] J. Ye. "Generalized Low Rank Approximations of Matrices". In: *Machine Learning (Springer)* 61.1 (2005), pp. 167–191. (Cit. on pp. 3, 103, 142, 183).

[239] J. Ye. "Least squares linear discriminant analysis". In: *ICML '07: Proceedings of the 24th international conference on Machine learning*. Corvalis, Oregon: ACM, 2007, pp. 1087–1093. (Cit. on p. 24).

[240] W. Yin et al. "Bregman iterative algorithms for l1-minimization with applications to compressed sensing". In: *SIAM Journal on Imaging Sciences* 1.1 (2008), pp. 143–168. (Cit. on p. 57).

[241] H.-F. Yu et al. "Scalable Coordinate Descent Approaches to Parallel Matrix Factorization for Recommender Systems". In: *IEEE International Conference on Data Mining (ICDM)*. 2012. (Cit. on p. 105).

[242] Q. Yu, T. B. Dinh, and G. Medioni. "Online tracking and reacquistion using co-trained generative and discriminative trackers". In: *European Conference on Computer Vision*. 2008, pp. 678–691. (Cit. on p. 189).

[243] M. Yuan and Y. Lin. "Model selection and estimation in regression with grouped variables". In: *Journal of the Royal Statistical Society, Series B* 68 (2006), pp. 49–67. (Cit. on pp. 6, 75, 77, 187, 207).

[244] D. Zachariah et al. "Alternating Least-Squares for Low-Rank Matrix Reconstruction". In: *IEEE Signal Processing Letters* 19.4 (2012), pp. 231–234. (Cit. on pp. 164, 188).

[245] R. Zass and A. Shashua. "Nonnegative Sparse PCA". In: *In Neural Information Processing Systems*. 2007, pp. 1561–1568. (Cit. on p. 18).

[246] M. L. Zhang and Z. H. Zhou. "ML-KNN: A lazy learning approach to multi-label learning". In: *Pattern Recognition* 40.7 (2007), pp. 2038–2048. (Cit. on pp. xxiv, 209–211, 223, 263, 283).

[247] M. L. Zhang and Z. H. Zhou. "Multi-Label Neural Networks with Applications to Functional Genomics and Text Categorization". In: *IEEE Transactions on Knowledge and Data Engineering* 18.10 (2006), pp. 1338–1351. (Cit. on p. 222).

[248] M.-L. Zhang and Z.-J. Wang. "MIMLRBF: RBF neural networks for multi-instance multi-label learning". In: *Neurocomputing* 72.16-18 (2009), pp. 3951–3956. (Cit. on p. 223).

[249] T. Zhang, D. Tao, and J. Yang. "Discriminative Locality Alignment". In: *ECCV '08: Proceedings of the 10th European Conference on Computer*

Vision. Marseille, France: Springer-Verlag, 2008, pp. 725–738. (Cit. on pp. 5, 38).

[250] T. Zhang et al. "Patch Alignment for Dimensionality Reduction". In: *IEEE Transactions on Knowledge and Data Engineering* 21.9 (2009), pp. 1299–1313. (Cit. on pp. 5, 38, 183).

[251] X. Zhang, T. Graepel, and R. Herbrich. "Bayesian Online Learning for Multi-label and Multi-variate Performance Measures". In: *AISTATS*. 2010. (Cit. on p. 201).

[252] Y. Zhang and J. Schneider. "Multi-Label Output Codes using Canonical Correlation Analysis". In: *AISTATS*. 2011. (Cit. on p. 201).

[253] Y. Zhang and Z. H. Zhou. "Multi-label dimensionality reduction via dependence maximization". In: *AAAI'08: Proceedings of the 23rd national conference on Artificial intelligence*. Chicago, Illinois, 2008, pp. 1503–1505. (Cit. on pp. xxiv, 187, 202, 209–211, 221, 263, 283).

[254] Z. Zhang and H. Zha. "Principal Manifolds and Nonlinear Dimensionality Reduction via Tangent Space Alignment". In: *SIAM J. Sci. Comput.* 26.1 (2005), pp. 313–338. (Cit. on p. 4).

[255] P. Zhao, S. C. H. Hoi, and R. Jin. "Double Updating Online Learning". In: *Journal of Machine Learning Research* 12 (2011), pp. 1587–1615. (Cit. on p. 14).

[256] P. Zhao, G. Rocha, and B. Yu. "Grouped and hierarchical model selection through composite absolute penalties". In: *Annals of Statistics* 37.6A (2009), pp. 3468–3497. (Cit. on p. 77).

[257] T. Zhou, W. Bian, and D. Tao. "Divide-and-Conquer Anchoring for Near-separable Nonnegative Matrix Factorization and Completion in High Dimensions". In: *IEEE International Conference on Data Mining (ICDM)*. 2013. (Cit. on p. 10).

[258] T. Zhou and D. Tao. "Greedy bilateral sketch, completion & smoothing". In: *Journal of Machine Learning Research - Proceedings Track* 31 (2013), pp. 650–658. (Cit. on pp. 12, 15, 107, 122, 124).

[259] T. Zhou and D. Tao. "Multi-label Subspace Ensemble". In: *AISTATS*. 2012. (Cit. on pp. 7, 13, 165, 187).

[260] T. Zhou and D. Tao. "Shifted Subspaces Tracking on Sparse Outlier for Motion Segmentation". In: *IJCAI*. 2013. (Cit. on pp. 13, 186).

[261] T. Zhou and D. Tao. "1-bit Hamming Compressed Sensing". In: *ISIT '12: IEEE International Symposium on Information Theory*. 2012. (Cit. on pp. 6, 54).

[262] T. Zhou and D. Tao. "Backward-Forward Least Angle Shrinkage for Sparse Quadratic Optimization". In: *ICONIP*. 2010, pp. 388–396. (Cit. on p. 6).

[263] T. Zhou and D. Tao. "Bilateral Random Projections". In: *ISIT '12: IEEE International Symposium on Information Theory*. 2012. (Cit. on pp. 12, 54, 143, 186, 190).

[264] T. Zhou and D. Tao. "Double Shrinking Sparse Dimension Reduction". In: *IEEE Transactions on Image Processing* 22.1 (2013), pp. 244–257. (Cit. on p. 10).

[265] T. Zhou and D. Tao. "GoDec: Randomized Low-rank & Sparse Matrix Decomposition in Noisy Case". In: *ICML '11: International Conference on Machine Learning*. 2011. (Cit. on pp. 12, 165, 166, 203).

[266] T. Zhou and D. Tao. "Labelset anchored subspace ensemble (LASE) for multi-label annotation". In: *ICMR*. 2012, p. 42. (Cit. on p. 13).

[267] T. Zhou and D. Tao. "Manifold Elastic Net for Sparse Learning". In: *SMC*. 2009, pp. 3699–3704. (Cit. on p. 10).

[268] T. Zhou, D. Tao, and X. Wu. "Compressed labeling on distilled labelsets for multi-label learning". In: *Machine Learning (Springer)* (2012). (Cit. on pp. 13, 58).

[269] T. Zhou, D. Tao, and X. Wu. "Manifold elastic net: a unified framework for sparse dimension reduction". In: *Data Mining and Knowledge Discovery (Springer)* 22.3 (2011), pp. 340–371. (Cit. on pp. 10, 149, 183, 221).

[270] T. Zhou, D. Tao, and X. Wu. "NESVM: A Fast Gradient Method for Support Vector Machines". In: *ICDM '10: Proceedings of the 2010 IEEE International Conference on Data Mining*. 2010, pp. 679–688. (Cit. on pp. 15, 54, 273, 289).

[271] Z. Zhou et al. "Stable Principal Component Pursuit". In: *ISIT*. 2010. (Cit. on pp. 150, 166, 185, 189).

[272] J. Zhuang et al. "When recommendation meets mobile: contextual and personalized recommendation on the go". In: *Ubicomp*. 2011, pp. 153–162. (Cit. on pp. 8, 173).

[273] H. Zou, T. Hastie, and R. Tibshirani. "Sparse principal component analysis". In: *Journal of Computational and Graphical Statistics* 15.2 (2006), pp. 262–286. (Cit. on pp. xxii, 6, 18, 57, 80, 85, 91, 92, 97).

[274] H. Zou. "The Adaptive Lasso and Its Oracle Properties". In: *Journal of the American Statistical Association* 101 (2006), pp. 1418–1429. (Cit. on pp. 52, 75, 76).

[275] H. Zou and T. Hastie. "Regularization and variable selection via the Elastic Net". In: *Journal of the Royal Statistical Society B* 67 (2005), pp. 301–320. (Cit. on pp. 6, 75, 76).