COLLABORATIVE FILTERING AND MULTI-LABEL CLASSIFICATION WITH MATRIX FACTORIZATION

A thesis submitted during 2018 to the University of Hyderabad in partial fulfillment of the award of a Ph.D. degree in Computer Science

by

VIKAS KUMAR

SCHOOL OF COMPUTER & INFORMATION SCIENCES

UNIVERSITY OF HYDERABAD

(P.O.) CENTRAL UNIVERSITY

HYDERABAD - 500 046, INDIA

July 30, 2018

CERTIFICATE

This is to certify that the thesis entitled **"Collaborative Filtering and Multi-Label Classification with Matrix Factorization"** submitted by **Vikas Kumar** bearing **Reg. No. 14MCPC07** in partial fulfillment of the requirements for the award of **Doctor of Philosophy** in **Computer Science** is a bonafide work carried out by him under our supervision and guidance.

This thesis is free from plagiarism and has not been submitted previously in part or in full to this or any other university or institution for the award of any degree or diploma. The student has the following publications before submission of the thesis for adjudication and has produced evidence for the same.

1. "Collaborative Filtering Using Multiple Binary Maximum Margin Matrix Factorizations." *Information Sciences* 380 (2017): 1-11.

2. "Proximal Maximum Margin Matrix Factorization for Collaborative Filtering." *Pattern Recognition Letters* 86 (2017): 62-67.

3. "Multi-label Classification Using Hierarchical Embedding." *Expert Systems with Applications* 91 (2018): 263-269.

Further, the student has passed the following courses towards fulfillment of course-work requirement for Ph.D.

Course Code	Name	Credits	Pass/Fail
CS 801	Data Structures and Algorithms	4	Pass
CS 802	Operating Systems and Programming	4	Pass
AI 810	Metaheuristic Techniques	4	Pass
AI 852	Learning & Reasoning	4	Pass

Prof. Arun K Pujari
Supervisor

Prof. Vineet Padmanabhan
Supervisor

Prof. K. Narayana Murthy
Dean, School of Computer and Information Sciences

DECLARATION

I, **Vikas Kumar**, hereby declare that this thesis entitled **"Collaborative Filtering and Multi-Label Classification with Matrix Factorization"** submitted by me under the guidance and supervision of **Prof. Arun K Pujari** and **Prof. Vineet Padmanabhan** is a bonafide research work. I also declare that it has not been submitted previously in part or in full to this university or any other university or institution for the award of any degree or diploma.

Date:

Name: **Vikas Kumar**

Signature of the Student:

Reg. No. **14MCPC07**

Dedicated to my parents and family members, for everything what I am today and above all, devoted to The Almighty God!

ACKNOWLEDGEMENTS

The accomplishment of doctoral thesis and degree is a result of collective support and blessings of various people during my journey of Ph.D. I feel overwhelm and pleasure to acknowledge each one for the kind of contribution provided to me throughout my life.

The foremost is my advisor, **Prof. Arun K Pujari** who has not only taught me science but the facts behind and beyond science. Sir has endowed me with a sacred vision to pursue productive full time research in computer science. The importance of bearing an optimistic attitude and crystal clear concept of the subject is a boon conferred by sir towards shaping my research questions during my years of Ph.D. I am extremely obliged to sir for making himself always available for rigorous scientific discussions in spite of his busy schedule. Those insightful inputs by sir are the substantial asset to build the carrier further. I wish to acknowledge my sir for being the guiding light in my dark days of research. It is also a privilege to express my sincere regards to **Mrs. Subha Lakhmi Pujari** for her care, encouragement and responsible assistance.

I sincerely gratify my other advisor, **Prof. Vineet Padmanabhan** for being both amicable and friendly supervisor with whom I was able to share the days of being a research scholar. The periodical scientific counselling and supervision when I was at the initial phase of my Ph.D. carrier is accountably assisted by sir. I genuinely acknowledge sir for taking out his valuable time for undertaking detailed analysis of the given research problem and giving me precise comments. I thank **Mrs. Mrinalini Vineet** for her truthful blessings and generous wishes to accomplish success in my research.

I would like to thank my DRC members, **Prof. Arun Agarwal** and **Prof. Chakravarthy Bhagvati**, for their valuable suggestions. I extend admiration to my

ABSTRACT

Machine learning techniques for Recommendation System and Classification has become a prime focus of research to tackle the problem of information overload. Recommender Systems are software tools that aim at making informed decisions about the services that a user may like. Recommender Systems can be broadly classified into two categories namely, *content-based filtering* and *collaborative filtering*. In content-based filtering, users and items are represented using a set of features (profile) and an item is recommended by finding the similarity between the user and item profile. On the other hand in collaborative filtering, the user-item association is obtained based on the preferences of the user given so far and the preference information of other users.

Classification technique deals with the categorization of a data object into one of the several predefined classes. The majority of the methods for supervised machine learning proceeds from a formal setting in which data objects (instances) are represented in the form of feature vectors wherein each object is associated with a unique class label from a set of disjoint class labels. Depending on the total number of disjoint classes, a learning task is categorized as binary classification or multi-class classification. In the multi-label classification problem, unlike the traditional multi-class classification setting, each instance can be simultaneously associated with a subset of labels.

In both recommendation and classification problem, the initial assumption is that the input-data is in the form of matrices which are inherently low rank. The key technical challenge involved in designing new algorithms for recommendation and classification is dependent on how well one can handle the huge and sparse matrices which usually has thousands to millions of rows and which are usually noisy. Recent years have witnessed extensive applications of low-rank linear factor model for exploiting the complex

relationships that exist in such data matrices. The goal is to learn a low-dimensional embedding where the data object can be represented with a small number of features. Matrix factorization methods have attracted significant attention for learning the low-rank latent factors. The focus of thesis is on the development of novel techniques for collaborative filtering and multi-label classification.

In maximum margin matrix factorization scheme of collaborative filtering, ratings matrix with multiple discrete values is treated by specially extending hinge loss function to suit multiple levels. We view this process as analogous to extending two-class classifier to a unified multi-class classifier. Alternatively, multi-class classifier can be built by arranging multiple two- class classifiers in a hierarchical manner. We investigate this aspect for collaborative filtering and propose a novel method of constructing a hierarchical bi-level maximum margin matrix factorization to handle matrix completion of ordinal rating matrix. The advantages of the proposed method over other matrix factorization based collaborative filtering methods are given by detailed experimental analysis. We also observe that there could be several possible alternative criteria to formulate the factorization problem of discrete ordinal rating matrix, other than the maximum margin criterion. Taking the cue from the alternative formulation of support vector machines, a novel loss function is derived by considering *proximity* as an alternative criterion instead of margin maximization criterion for matrix factorization framework. We validate our hypothesis by conducting experiments on real and synthetic datasets.

We extended the concept of matrix factorization for yet another important problem of machine learning namely multi-label classification which deals with the classification of data with multiple labels. We propose a novel piecewise-linear embedding method with a low-rank constraint on parametrization to capture nonlinear intrinsic relationships that exist in the original feature and label space. Extensive comparative studies to validate the effectiveness of the proposed method against the state-of-the-art multi-label learning approaches is given through detailed experimental analysis. We also study the embedding of labels together with the group information with an objective to build an efficient multi-label classifier. We assume the existence of a low-dimensional space onto which the feature vectors and label vectors can be embedded. We ensure that labels belonging to the same group share the same sparsity pattern in their low-rank representations. We perform comparative analysis which manifests the superiority of our proposed method over state-of-art algorithms for multi-label learning.

TABLE OF CONTENTS

LIST OF TABLES

LIST OF FIGURES

ABBREVIATIONS

CF	Collaborative Filtering
FE	Feature Space Embedding
GroPLE	Group Preserving Label Embedding for Multi-Label Classification
HMF	Hierarchical Matrix Factorization
LE	Label Space Embedding
MF	Matrix Factorization
MLC	Multi-label Classification
MLC-HMF	Multi-label Classification Using Hierarchical Embedding
MMMF	Maximum Margin Matrix Factorization
PMMMF	Proximal Maximum Margin Matrix Factorization

CHAPTER 1

Introduction

With the advent of technology and high usage of modern equipment/ devices, large amounts of data are being generated world over. We now stand at the brink of data-driven transformation where the data can be harnessed to derive meaningful insights that can help organizations for better functioning as well as assist the user in his/ her decision making process. Examples include (a) suggesting the right choice of products to a user or identifying the most appropriate customers for a product (b) assisting an organization in classifying the users for better serviceability etc.

The vast amount of accumulated data is a crucial competitive asset and can be tailored to satisfy an individual's/ organization's needs. However, the crucial challenge here is in the processing of the data. The amount of data is enormous and therefore requires a large amount of time to explore all of them. For example, if one has to purchase an item, he/ she needs to process all the information to select which items meet their needs. To avoid the information overload, we need some assistance in the form of recommendation or classification in our day to day life. For instance, when we are purchasing an electronic gadget, we usually rely on the suggestions of friends who shares similar tastes. Similarly, when we are planning to invest in equity funds, we need some assistance to differentiate between different equity based on their features before we decide to invest. In the recent past, machine learning techniques are most commonly used to understand the nature of data so as to help in reducing the burden on individuals in the decision making process. Machine learning techniques for *Recommendation* and *Classification* has become a prime focus of research to tackle the problem of information overload.

Recommendation (Recommender) Systems are software tools that aim at making informed decisions about the services that a user may like (102; 64; 81). Given a list of users, items and user-item interactions (ratings), Recommender System predicts the score/ affinity of item j for user i and thereby helps in understanding the user behaviour which in turn can be used to make personalized recommendations. Examples include classic recommendation tasks such as recommending books, movies, music etc., as well as new web-based applications such as predicting preferred news articles, websites etc. Recommender Systems can be broadly classified into two categories namely, content-based filtering and collaborative filtering(81). In content-based filtering, users and items are represented using a set of features (profile) and an item is recommended by finding the similarity between the user and the item profile. In the case of collaborative filtering the user-item association is obtained based on the preferences of the user given so far and the preference information of other users.

Classification is a supervised learning technique which deals with the categorization of a data object into one of the several predefined classes. Majority of the methods for supervised machine learning proceeds from a formal setting in which the data objects (instances) are represented in the form of feature vectors wherein each object is associated with a unique class label from a set of disjoint class labels L, $|L| > 1$. Depending on the total number of disjoint classes in L, a learning task is categorized as binary classification (when $|L| = 2$) or multi-class classification (when $|L| > 2$) (95; 115). In this thesis, we are focusing on a special class of classification problem called multi-label classification. Unlike the traditional classification setting, in multi-label classification problem, each instance can be simultaneously associated with a subset of labels. For example, in image classification, an image can be simultaneously tagged with a subset of labels such as *natural*, *valley*, *mountain* etc. Similarly, in document classification, a document can simultaneously belong to *Computer Science* and *Physics*. The multi-label classification task aims at building a model that can automatically tag a new example with the most relevant subset of class labels.

In this thesis we focus on developing novel techniques for collaborative filtering and multi-label classification. It should be kept in mind that in both recommendation and classification problem the data is actually organized in matrix form. For example, in collaborative filtering, users preferences on items can be represented as a matrix, whose rows represent users, columns represent items, and each element of the matrix represent

the preference of a user for an item. Similarly, in multi-label classification problem, the set of data objects and their corresponding label vectors can be represented as a matrix. The data object can be represented as a row of the feature matrix and the associated label vector can be represented as the corresponding row of label matrix. The key technical challenge involved in designing new algorithms for recommendation and classification is dependent on how well one can handle the huge and sparse matrices which usually has thousands and millions of rows and which are usually noisy.

Recent years have witnessed extensive applications of low-rank linear factor models for exploiting the complex relationships existing in such data matrices. The goal is to learn a low-dimensional embedding where the data object can be represented with a small number of features. For instance, in the case of collaborative filtering, the idea is to learn low-dimensional latent factors for every user and item. Similarly, in multi-label classification the goal of low-rank factor model learning is to embed the feature and label vector to a low dimensional space so that the intrinsic relationship in the original space can be captured. Matrix factorization (MF) methods have attracted significant attention in the areas of computer vision (13; 14), pattern recognition (124; 143), image processing (30; 37), information retrieval (57; 111) and signal processing (121; 125) for learning low-rank latent factor models. The objective of MF is to learn low-rank latent factor matrices U and V so as to simultaneously approximate the observed entries under some loss measure. Here, the interpretation of factor matrices U and V are application dependent. We present a brief discussion on matrix factorization approach for collaborative filtering and multi-label classification.

Matrix factorization is just one way of doing collaborative filtering (CF) wherein it is possible to discover the latent features underlying the interactions between two different kinds of entities (user/ item). CF approaches also assume that a user's preference on an item is determined by a small number of factors and how each of those factors applies to the user and the item (low dimensional linear factor model). The intuition behind using matrix factorization is that there should be some latent features that determine how a user rates an item. It is possible that two users would highly rate a movie if it belongs to a particular genre like action/ comedy. Discovering these latent features definitely helps in predicting the rating with respect to a certain user and a certain item because the features associated with the user should match with the features associated with the item. It is also the case that in trying to discover the different features, an

assumption is made that the number of features would be smaller than the number of users and the number of items. The idea is that it is not reasonable to assume that each user is associated with a unique feature in which case it would be absurd to make any recommendations because each of these users would not be interested in the items rated by other users.

The discussion above shows that CF can be formulated as a MF problem in the sense that, in a d-factor model, given a rating matrix $Y \in \mathbb{R}^{N \times M}$ the idea is to find two low-rank matrices $U \in \mathbb{R}^{N \times d}$ and $V \in \mathbb{R}^{M \times d}$ such that $Y \approx UV^T$ where d is a parameter. Each entry $y_{ij} \in \{0, 1, 2, \ldots, R\}$ in Y, defines the preference given by ith user for jth item. $y_{ij} = 0$, indicates that the user has not given any preference for jth item and R is the total level of ratings. The goal is to predict the preference for all items for which the user's preference is unobserved. The ith row of U represents the latent factor of the ith user and the jth row of V represents the latent feature of the jth item. The prediction for y_{ij} can be achieved by the linear combination of the ith row of U with the jth row of V i.e.

$$y_{ij} = \sum_{p=1}^{d} U_{ip} V_{jp} = U_i V_j^T \tag{1.1}$$

The outcome of matrix factorization method is a low-rank, dense matrix X which is an approximation of the given sparse rating matrix Y. Here X is computed as $X = UV^T$.

Multi-label classification can be seen as a generalization of *single label* classification where an instance is associated with a unique class label from a set of disjoint labels L. Formally, given N training examples in the form of a pair of feature matrix X and label matrix Y where each example $x_i \in \mathbb{R}^D, 1 \leq i \leq N$, is a row of X and its associated labels $y_i \in \{-1, +1\}^L$ is the corresponding row of Y, the task of multi-label classification is to learn a parameterization $h : \mathbb{R}^D \to \{-1, +1\}^L$ that maps each instance to a set of labels. Multi-label classification has applications in many areas such as machine learning (1; 135; 95; 147), computer vision (79; 123; 12; 9) and data mining (87; 154; 115; 103). Multi-label classification is nowadays applied to massive data sets of considerable size and under such conditions, time- and space-efficient implementations of learning algorithms is of major concern. For example, there are millions of articles available on Wikipedia each tagged with a set of labels and in most of the cases, the author of an article associates very few labels which they know from a broad set of relevant labels.

To cope with the challenge of exponential-sized output space, exploiting intrinsic information in feature and label spaces has been the major thrust of research in recent years and use of parametrization and embedding have been the prime focus. Researchers have studied several aspects of embedding which include label embedding, feature embedding, dimensionality reduction and feature selection. These approaches differ from one another in their capability to capture other intrinsic properties such as label correlation and local invariance. To glean out the potentially hidden dynamics that exists in the original space, most of the embedding based approaches focus on learning a low-dimensional representation for the original feature (label) vector. There are two strategies of embedding for exploiting inter-label correlation; (1) Feature Space Embedding (FE); and (2) Label Space Embedding (LE). FE aims to design a projection function which can map the instance in the original feature space to an embedded space and then a mapping is learnt from the embedded space to the label space. The second approach is to transform the label vectors to an embedded space, followed by the association between feature vectors and embedded label space for classification purpose.

In recent years, matrix factorization based approach which aims at determining two matrices U and V is frequently used to achieve the FE and LE. In FE, the goal is to transform each D-dimensional feature vector (a row of matrix X) from the original feature space to a L-dimensional label vector (corresponding row in Y) and usually the mapping is achieved through a linear transformation matrix $W \in \mathbb{R}^{D \times L}$. The MF based approach assumes that the label matrix Y is of low-rank due to the presence of similar labels and thereby models the inter-label correlation implicitly using low-rank constraints on the transformation matrix W. The transformation matrix W is approximated using the product of two latent factor matrices U and V. The matrix U acts as a transformation matrix which transforms the data from the original feature space to an embedded space and the matrix V can be interpreted as the decoding matrix from the embedded space to the label space.

In LE, the goal is to transform each L-dimensional label vector (a row of matrix Y) from the original label space to a d-dimensional embedded vector. Thereafter, a predictive model is trained from the original feature space to the embedded space. With proper decoding process that maps the projected data back to the original label space, the task of multi-label prediction is achieved. The LE approach based on MF aims at approximating the label matrix Y as a product of two matrices U and V with the

assumption that the label matrix exhibits a low-rank structure. The ith row of matrix U can be viewed as the embedded representation of the label vector associated with the ith instance and the matrix V can be viewed as the decoding matrix from the embedded space to the label space.

1.1 Contributions of the Thesis

The major contributions of the thesis are as follows. In maximum margin matrix factorization scheme (a variant of basic matrix factorisation method), ratings matrix with multiple discrete values is treated by specially extending *hinge loss function* to suit multiple levels. We view this process as analogous to extending two-class classifier to a unified multi-class classifier. Alternatively, multi-class classifier can be built by arranging multiple two- class classifiers in a hierarchical manner. We investigate this aspect for collaborative filtering and propose a novel method of constructing a hierarchical bi-level maximum margin matrix factorization to handle matrix completion of ordinal rating matrix.

We observe that there could be several possible alternative criteria to formulate the factorization problem of discrete ordinal rating matrix, other than the maximum margin criterion. Taking a cue from the alternative formulation of support vector machines, a novel loss function is derived by considering *proximity* as an alternative criterion instead of margin maximization criterion for matrix factorization framework.

We extended the concept of matrix factorization for yet another important problem of machine learning namely multi-label classification which deals with the classification of data with multiple labels. We propose a novel piecewise-linear embedding method with a low-rank constraint on the parametrization to capture nonlinear intrinsic relationships that exist in the original feature and label space.

We study the embedding of labels together with the group information with an objective to build an efficient multi-label classifier. We assume the existence of a low-dimensional space onto which the feature vectors and label vectors can be embedded. We ensure that labels belonging to the same group share the same sparsity pattern in their low-rank representations.

1.2   Structure of the Thesis

The thesis is organized as follows. In Chapter 2, we start our discussion with an introductory discussion on matrix factorization and the associated optimization problem formulation. Thereafter we discuss some common loss functions used in matrix factorization to measure the deviation between the observed data and the corresponding approximation. We also discuss several ways of norm regularization which is needed to avoid overfitting in matrix factorization models. In the later part of the chapter we discuss at length the application of matrix of factorization techniques in collaborative filtering and multi-label classification.

Chapter 3 starts with a discussion on bi-level maximum margin matrix factorization (MMMF) which we subsequently use in our proposed algorithm. We carry out a deep investigation of the well known maximum margin matrix factorization technique for discrete ordinal rating matrix. This investigation led us to propose a novel and efficient algorithm called HMF (*Hierarchical Matrix Factorization*) for constructing a hierarchical bi-level maximum margin matrix factorization method to handle matrix completion of ordinal rating matrix. The advantages of HMF over other matrix factorization based collaborative filtering methods are given by detailed experimental analysis at the end of the chapter.

Chapter 4 introduces a novel method termed as PMMMF (*Proximal Maximum Margin Matrix Factorization*) for factorization of matrix with discrete ordinal ratings. Our work is motivated by the notion of Proximal SVMs (PSVMs) (74; 31) for binary classification where two *parallel* planes are generated, one for each class, unlike the standard SVMs (118; 10; 84). Taking the cue from here, we make an attempt to introduce a new loss function based on the *proximity* criterion instead of margin maximization criterion in the context of matrix factorization. We validate our hypothesis by conducting experiments on real and synthetic datasets.

Chapter 5 extended the concept of matrix factorization for yet another important problem in machine learning namely multi-label classification. We visualize matrix factorization as a kind of low-dimensional embedding of the data which can be practically relevant when a matrix is viewed as a transformation of data from one space to the other. At the beginning of the chapter we discuss briefly about the traditional approach

of multi-label classification and establish a bridge between multi-label classification and matrix factorization. We present a novel multi-label classification method, called MLC-HMF (*Multi-label Classification using Hierarchical Embedding*), which learns piecewise-linear embedding with a low-rank constraint on parametrization to capture nonlinear intrinsic relationships that exist in the original feature and label space. Extensive comparative studies to validate the effectiveness of the proposed method against the state-of-the-art multi-label learning approaches is discussed in the experimental section.

In Chapter 6, we study the embedding of labels together with group information with an objective to build an efficient multi-label classifier. We assume the existence of a low-dimensional space onto which the feature vectors and label vectors can be embedded. In order to achieve this, we address three sub-problems namely; (1) Identification of groups of labels; (2) Embedding of label vectors to a low rank-space so that the sparsity characteristic of individual groups remains invariant; and (3) Determining a linear mapping that embeds the feature vectors onto the same set of points, as in stage 2, in the low-rank space. At the end, we perform comparative analysis which manifests the superiority of our proposed method over state-of-art algorithms for multi-label learning.

We conclude the thesis with a discussion on future directions in Chapter 7.

1.3 Publications of the Thesis

1. Vikas Kumar, Arun K Pujari, Sandeep Kumar Sahu, Venkateswara Rao Kagita, Vineet Padmanabhan. "Collaborative Filtering Using Multiple Binary Maximum Margin Matrix Factorizations." Information Sciences 380 (2017): 1-11.

2. Vikas Kumar, Arun K Pujari, Sandeep Kumar Sahu, Venkateswara Rao Kagita, Vineet Padmanabhan. "Proximal Maximum Margin Matrix Factorization for Collaborative Filtering." Pattern Recognition Letters 86 (2017): 62-67.

3. Vikas Kumar, Arun K Pujari, Vineet Padmanabhan, Sandeep Kumar Sahu, Venkateswara Rao Kagita. "Multi-label Classification Using Hierarchical Embedding." Expert Systems with Applications 91 (2018): 263-269.

4. Vikas Kumar, Arun K Pujari, Vineet Padmanabhan, Venkateswara Rao Kagita. "Group Preserving Label Embedding for Multi-Label Classification." Pattern Recognition, Under Review.

CHAPTER 2

Foundational Concepts

In majority of data-analysis tasks, the datasets are naturally organized in matrix form. For example, in collaborative filtering, users preferences on items can be represented as a matrix, whose rows represent users, columns represent items, and each element of the matrix represent the preference of a user for an item. In a clustering problem, a row of the matrix represents data object (instance) and the columns represent associated features (attributes). In image inpainting problem, an image can be represented by a matrix where each entry of the matrix corresponds to pixel values. Similarly, in document analysis, a set of document can be represented as a matrix wherein the rows represent terms and the columns represent documents. Each entry represents the frequency of the associated terms in a particular document.

Recent advances in many data-analysis tasks have been focused on finding a meaningful (simplified) representation of the original data matrix. A simplified representation typically helps in better understanding the structure, relationship within the data or attributes and retrieving the hidden information that exists in the original data matrix. Matrix Factorization (MF) methods have attracted significant attention for finding the latent (hidden) structure from the original data matrix. Given an original matrix $Y \in \mathbb{R}^{N \times M}$, MF aims at determining two matrices $U \in \mathbb{R}^{N \times d}$ and $V \in \mathbb{R}^{M \times d}$ such that $Y \approx UV^T$ where the inner dimension d is called the *numerical rank* of the matrix. The numerical rank is much smaller than M and N, and hence, factorization allows the matrix to be stored inexpensively. Here, the interpretation of factor matrices U and V are application dependent. For example, in collaborative filtering, given that Y is a $N \times M$ size user-item rating matrix, MF aims at determining the latent factor represen-

tation of users and items. The ith row of U represents the latent factor of the ith user and the jth row of V represents the latent feature of the jth item (65; 97; 86; 62; 101). In a clustering problem with M data points (columns of Y) and d cluster, the jth column of U can be interpreted as the representative of the jth cluster and ith column of V^T can be visualized as the reconstruction weights of the ith data point from those representatives (25; 134; 78; 122).

The MF field is too vast to cover and there are many research directions one can pursue starting from theoretical aspects to application-oriented aspects. In this thesis, we focus our discussion on two special applications of matrix factorization, namely collaborative filtering (65; 97; 86; 62; 101) and multi-label classification (12; 130; 3; 53). The purpose of this chapter is to introduce the basic matrix factorization framework and concepts which will be used throughout the thesis (Section 2.1). In Section 2.2, we discuss some of the common loss functions used in MF to measure the deviation between the observed data and corresponding approximation. We briefly discuss the different ways to avoid overfitting in MF model in Section 2.3. In Section 2.4 and 2.5, we explore how matrix factorisation can be applied to interesting applications of machine learning namely collaborative filtering and multi-label classification, respectively.

2.1   Matrix Factorization

Matrix factorization is a key technique employed for many real-world applications wherein the objective is to select U and V among all the matrices that fit the given data and the one with the lowest rank is preferred. In most applications, the task that needs to be performed is not just to compute any factorization, but also to enforce additional constraints on the factors U and V. The matrix factorization problem can be formally defined as follows:

Definition 1 (Matrix Factorization). *Given a data matrix $Y \in \mathbb{R}^{N \times M}$, matrix factorization aims at determining two matrices $U \in \mathbb{R}^{N \times d}$ and $V \in \mathbb{R}^{M \times d}$ such that $Y \approx UV^T$ where the inner dimension d is called the numerical rank of the matrix. The numerical rank is much smaller than M and N, and hence, factorization allows the matrix to be stored inexpensively. $X = UV^T$ is called as low-rank approximation of Y.*

A common formulation for this problem is a regularized loss problem which can be given as follows.

$$\min_{U,V} \ell(Y, U, V) + \lambda R(U, V) \qquad (2.1)$$

where Y is the data matrix, $X = UV^T$ is its low-rank approximation, $\ell(\cdot)$ is a loss function that measures how well X approximates Y, and $R(\cdot)$ is a regularization function that promotes various desired properties in X (low-rank, sparsity, group-sparsity, etc.). When $\ell(\cdot)$ and $R(\cdot)$ are convex functions of X (equivalently, of U and V) the above problem can be solved efficiently. The loss function $\ell(Y, U, V)$ can further be decmposed into the sum of pairwise discrepancy between the observed entries and their corresponding prediction, that is, $\ell(Y, U, V) = \sum_{i=1}^{N} \sum_{j=1}^{M} \ell(y_{ij}, U_i, V_j)$, where y_{ij} is the ijth entry of the matrix Y. There have been umpteen number of proposals for factorizing a matrix and these differ, by and large, among themselves in defining the loss function and the regularization function. The structure of the latent factors depends on the types of loss (cost) functions and the constraints imposed on the factor matrices. In the following subsections, we present a brief literature review of the major loss functions and regularizations.

2.2 Loss Function

Loss function $\ell : Y \times Y \rightarrow \mathbb{R}$ in MF model is used to measure the discrepancy between the observed entry and the corresponding prediction. Based on the types of observed data, the loss function in MF can be roughly grouped into three categories: (1) Loss function for Binary data; (2) Loss function for Discrete ordinal data; and (3) Loss function for Real-valued data.

2.2.1 Binary Loss Function:

For a binary data matrix $Y = [y_{ij}]$ where the entries are restricted to take only two values $y_{ij} \in \{+1, -1\}$, we review some of the important loss functions such as *zero-one loss*, *hinge loss*, *smooth hinge loss*, *modified least square* and *logistic loss* (98) in the following subsections.

Zero-One Loss : The zero-one loss is a standard loss function to measure the penalty

for an incorrect prediction which takes only two values zero or one. For a given observed entry y_{ij} and the corresponding prediction $U_iV_j^T$, zero-one loss is defined as

$$\ell(z) = \begin{cases} 0 & \text{if } z \geq 0; \\ 1 & \text{otherwise,} \end{cases} \qquad (2.2)$$

where $z = y_{ij}(U_iV_j^T)$.

Hinge Loss: There are two major drawbacks with zero-one loss (1) Minimizing objective function involving zero-one loss is difficult to optimize as the function is non-convex; (2) It is insensitive to the magnitude of prediction whereas in general, when the entries are binary, the magnitude of score $y_{ij}(U_iV_j^T)$ represent the confidence in our prediction. The hinge loss is a convex upper bound on zero-one error (34) and sensitive to the magnitude of prediction. In the case of classification with large confidence (margin), hinge loss is the most preferred loss function and is defined as

$$\ell(z) = \begin{cases} 0 & \text{if } z \geq 1; \\ 1 - z & \text{otherwise.} \end{cases} \qquad (2.3)$$

Smooth Hinge Loss: Hinge loss, $h(z)$ is non-differentiable at $z = 1$ and very sensitive to outliers (97). An alternative of hinge loss is proposed in (98) called smooth hinge and is defined as

$$\ell(z) = \begin{cases} 0 & \text{if } z \geq 1; \\ \frac{1}{2}(1 - z)^2 & \text{if } 0 < z < 1; \\ \frac{1}{2} - z & \text{otherwise.} \end{cases} \qquad (2.4)$$

Smooth Hinge shares many properties with hinge loss and is insensitive to outliers. A detailed discussion about hinge and smooth hinge loss is given in Chapter 3.

Modified Square Loss: In (150), the hinge function is replaced by a smoother quadratic function to make the derivative smooth. The modified square loss is defined as

$$\ell(z) = \begin{cases} 0 & \text{if } z \geq 1; \\ (1 - z)^2 & \text{otherwise.} \end{cases} \qquad (2.5)$$

Logistic Loss: The logistic loss function is strictly convex function which enjoys prop-

erties similar to that of the hinge loss (83). The logistic loss is defined as

$$\ell(z) = \log(1 + \exp^{-z}). \tag{2.6}$$

2.2.2 Discrete Ordinal Loss Function

For a discrete ordinal matrix $Y = [y_{ij}]$ where entries are no more *like* and *dislike* but can be any value from a discrete range, that is, $y_{ij} \in \{1, 2, \ldots, R\}$, there is need to extend binary class loss function to multi-class loss function. There are two major approaches for extending loss function for binary class classification to multi-class classification. The first approach is to directly solve a multi-class problem by modifying the binary class objective function by adding a constraint to it for every class as suggested in (98) (140) . The second approach is to decompose the multi-class classification problem into a series of independent binary class classification problems as given in (11).

To extend binary loss function to multi-class setting, most of the approaches define a set of threshold values $\theta_1 < \theta_1 < \cdots < R - 1$ such that the real line is divided into R regions. The region defined by threshold θ_{q-1} and θ_q corresponds to rating q (98). For simplicity of notation, we assume $\theta_0 = -\infty$ and $\theta_R = +\infty$. There are different approaches for constructing a loss function based on a set of threshold values such as immediate-threshold and all-threshold (98). For each observed entry and corresponding prediction pair $(y_{ij}, U_i V_j^T)$, the immediate-threshold based approach calculates the loss as the sum of immediate-threshold violation.

$$\ell(y_{ij}, U_i V_j^T) = \ell(U_i V_j^T - \theta_{i, y_{ij}-1}) \; + \; \ell(\theta_{i, y_{ij}} - U_i V_j^T) \tag{2.7}$$

On the other hand, all-threshold loss is calculated as the sum of loss for all threshold which is the cost of crossing multiple rating-boundaries and is defined as

$$\ell(U_i V_j^T, y_{ij}) = \sum_{r=1}^{y_{ij}-1} \ell(U_i V_j^T - \theta_{i,r}) + \sum_{r=y_{ij}}^{R-1} \ell(\theta_{i,r} - U_i V_j^T). \tag{2.8}$$

2.2.3 Real-valued Loss Function

For a real-valued data matrix $Y = [y_{ij}]$ where the entries are in real-valued domain $\mathbb{R}$, we review some of the important loss functions such as square-loss (90), KL-divergence (70), β-divergence (17) and Itakura-Saito divergence (29) loss. These are given in the following subsections.

Squared Loss: The square loss is the most common loss function used for real-valued prediction. The penalty for misclassification is calculated as the square distance between the observed entry and the corresponding prediction. The square loss is defined as

$$\ell(y_{ij}, \hat{y}_{ij}) = (y_{ij} - \hat{y}_{ij})^2 \tag{2.9}$$

where $\hat{y}_{ij} = U_i V_j^T$.

Kullback-Leibler Divergence Loss: The KL-divergence loss also know as I-divergence loss is the measure of information loss when $U_i V_j^T$ is used as an approximate to y_{ij}. The KL-divergence loss is defined as

$$\ell(y_{ij}, \hat{y}_{ij}) = y_{ij} \ln \frac{y_{ij}}{\hat{y}_{ij}} - y_{ij} + \hat{y}_{ij}. \tag{2.10}$$

Itakura-Saito Divergence: Itakura and Saito (29) is obtained from the the maximum likelihood (ML) estimation. The loss function is defined as

$$\ell(y_{ij}, \hat{y}_{ij}) = \frac{y_{ij}}{\hat{y}_{ij}} - y_{ij} \ln \frac{y_{ij}}{\hat{y}_{ij}} - 1. \tag{2.11}$$

β-Divergence Loss: Cichocki et al. (17) proposed a generalized family of loss function called β-divergence defined as

$$\ell(y_{ij}, U_i, V_j) = \begin{cases} \frac{y_{ij}^{\beta}}{\beta(\beta-1)} + \frac{\hat{y}_{ij}^{\beta}}{\beta} - \frac{y_{ij}\hat{y}_{ij}^{\beta-1}}{\beta-1} & \text{if } \beta \in \mathbb{R} \setminus \{0, 1\}; \\ y_{ij} \ln \frac{y_{ij}}{\hat{y}_{ij}} - y_{ij} + \hat{y}_{ij} & \text{if } \beta = 1; \\ \frac{y_{ij}}{\hat{y}_{ij}} - y_{ij} \ln \frac{y_{ij}}{\hat{y}_{ij}} - 1 & \text{if } \beta = 0, \end{cases} \tag{2.12}$$

where $\beta \geq 0$ is a generalization parameter. At limitng case $\beta = 0$ and $\beta = 1$, the β-divergence corresponds to Itakura-Saito divergence and Kullback-Leibler divergence,

respectively. The squared loss can be obtained as a special case at $\beta = 2$.

2.3 Regularization

Most of the machine learning algorithms suffer from the problem of overfitting where the model fits the training data too well but have poor generalization capability for new data. Thus, a proper technique should be adopted to avoid overfitting of the training data. In MF community, to make the model unbiased i.e., to avoid the model to fit only with a particular dataset, many researchers have tried with different regularization terms along with some loss function (63; 91; 110; 65). Regularization is not only used to prevent overfitting but also to achieve different structural representation of the latent factors. Several methods based on norm regularization has been proposed in the literature which includes ℓ_1, ℓ_2 and $\ell_{2,1}$.

ℓ_1 **Norm:** The ℓ_1 norm, also know as sparsity inducing norm, is used to produce sparse latent factors and thus avoids overfitting by retaining only the useful factors. In effect, this implies that most units take values close to zero while only few take significantly non-zero values. For a a given matrix A, the ℓ_1 norm is defined as

$$\|A\|_1 = \sum_{ij} |a_{ij}|. \tag{2.13}$$

ℓ_2 **Norm:** The most popular and widely investigated regularization term used in MF model is ℓ_2 norm which is also known as Frobenius norm. For a given matrix A, the ℓ_2 norm is defined as

$$\|A\|_F = \sqrt{\sum_{ij} a_{ij}^2} \tag{2.14}$$

where $\| \cdot \|_2$ is the ℓ_2 norm of a matrix. Minimizing the objective function containing the ℓ_2 norm as regularization term gives two benefits 1) It avoids overfitting by penalizing the large latent factor values. 2) Approximating the target matrix with low-rank factor matrix is a typical non-convex optimization problem and in fact, the ℓ_2 norm has also been suggested as a convex surrogate for the rank in control applications (28; 97).

$\ell_{2,1}$ **Norm:** The $\ell_{2,1}$ norm also known as group sparsity norm is used to induce a sparse

representation at the level of groups. For a given matrix A, the $\ell_{2,1}$ norm is defined as

$$\|A\|_{2,1} = \sum_{i=1}^{n} \sqrt{\sum_{j=1}^{m} A_{ij}^2}. \tag{2.15}$$

2.4 Collaborative filtering with Matrix Factorization

In collaborative filtering, the goal is to infer user preferences for items based on his/ her previously given preference and a large collection of preferences of other users. Given a partially observed $N \times M$ user-item rating matrix Y with N number of users and M number of items the goal is to predict unobserved preference of users for items. The collaborative filtering problem can be formally defined as follows:

Definition 2 (Collaborative Filtering). *Let Y be a $N \times M$ size user-item rating matrix and Ω be the set of observed entries. For each $(i,j) \in \Omega$, the entry $y_{ij} \in \{1, 2, \ldots, R\}$ defines the preference of ith user for jth item. For each $(i,j) \notin \Omega$, $y_{ij} = 0$ indicates that preference of ith user for jth item is not available (unsampled entry). Given a partially observed rating matrix $Y \in \mathbb{R}^{N \times M}$, the goal is to predict y_{ij} for $(i,j) \notin \Omega$.*

Matrix factorization is a key technique employed for completion of user-item rating matrix wherein the objective is to learn low-rank (or low-norm) latent factors U (for users) and V (for items) so as to simultaneously approximate the observed entries under some loss measure and predict the unobserved entries. There are various ways of doing so. It is shown in (68) that to approximate Y, the entries in the factor matrix U and V need to be non-negative so that only additive combination factors are allowed. The basic idea is to learn factor matrices U and V in such a way that, the squared sum distance between the observed entry and corresponding prediction is minimized. The optimization problem is formulated as

$$\min_{U \geq 0, V \geq 0} J(U, V) = \sum_{(i,j) \in \Omega} (y_{ij} - U_i V_j^T)^2. \tag{2.16}$$

Singular value decomposition (SVD) is used in (102) to learn the factor matrices U and V. The key technical challenge when SVD is applied to sparse matrices is that it suffers from severe overfitting. When SVD factorization is applied on sparse data, error

function needs to be modified so as to consider only the observed ratings by setting the non-observed entries to zero. This minor modification results in a non-convex optimization problem. Instead of minimizing the rank of a matrix, maximum margin matrix factorization (MMMF) (107) proposed by srebro et al. aims at minimizing the Froebenius norms of U and V, resulting in convex optimization problems. It is shown that MMMF can be formulated as a semi-definite programming (SDP) problem and solved using standard SDP solvers. However, current SDP solvers can only handle MMMF problems on matrices of dimensionality up to a few hundred. Hence, a direct gradient based optimization method for MMMF is proposed in (97) to make fast collaborative prediction. The detailed discussion about MMMF is given in Chapter 3. To further improve the performance of MMMF, in (23), MMMF is casted using ensemble methods which includes bagging and random weight seeding. MMMF was further extended in (126) by introducing offset terms, item dependent regularization and a graph kernel on the recommender graph. In (132), a noparametric Bayesian-style MMMF was proposed that utilizes nonparametric techniques to resolve the unknown number of latent factors in MMMF model (126)(97)(23). A probabilistic interpretation of MMMF was presented in (133) model through data augmentation.

The proposal of MMMF hinges heavily on extended hinge loss function. Research on different loss functions and their extension to handle multiple classes has not attracted much attention of researchers though there are some important proposals (80). MMMF has become a very popular research topic since its publication and several extensions have been proposed (23; 126; 132; 133). There has also been some research on matrix factorization on binary or bi-level preferences (119). But many view binary preference as a special case of matrix factorization with discrete ratings. In (152) the rating matrix is decomposed hierarchically by grouping similar users and items together, and each sub-matrix is factorized locally. To the best of our knowledge, there is no research on hierarchical MMMF.

2.5 Multi-label Classification with Matrix Factorization

In machine learning and statistics, the classification problem is concerned with the assignment of a class (category) to a data object (instance) from a given set of discrete

classes. For example, classifying a document into one of the several known categories such as *sports*, *crime*, *business*, *politics* etc. In a traditional classification problem, data objects are represented in the form of feature vectors, each associated with a unique class label from a set of disjoint class labels L, $|L| > 1$. Depending on the total number of disjoint classes in L, a learning task is categorized as *binary* classification (when $|L| = 2$) or *multi-class* classification (when $|L| > 2$) (105). However, in many real-word classification tasks, data object can be simultaneously associated with one or more than one class in L. For example, a document can simultaneously belong to more than one class such as *politics* and *business*. The objective of multi-label classification (MLC) is to build a classifier that can automatically tag an example with the most relevant subset of labels. This problem can be seen as a generalization of the *single label* classification where an instance is associated with a unique class label from a set of disjoint labels L. The multi-label classification problem can be formally defined as follows:

Definition 3 (Multi-label Classification). *Given N training examples in the form of a pair of feature matrix X and label matrix Y where each example $x_i \in \mathbb{R}^D, 1 \le i \le N$, is a row of X and its associated labels $y_i \in \{-1, 1\}^L$ is the corresponding row of Y. The $+1$ entry at the jth coordinate of vector y_i indicates the presence of label j in data point x_i. The task of multi-label classification is to learn a parametrization $h : \mathbb{R}^D \to \{-1, 1\}^L$ that maps each instance (or, a feature vector) to a set of labels (a label vector).*

MLC is a major research area in the machine learning community and finds application in several domains such as computer vision (12; 9), data mining (115; 103) and text classification (147; 103). Due to the exponential size of the output space, exploiting intrinsic information in the feature and the label space has been the major thrust of research in recent years and the use of parametrization and embedding techniques have been the prime focus in MLC. The embedding based approach assumes that there exists a low-dimensional space onto which the given set of feature vectors and/ or label vectors can be embedded. The embedding strategies can be grouped into two categories namely; (1) Feature space embedding; and (2) Label space embedding. Feature space embedding aims to design a projection function which can map the instance in the original feature space to the label space. On the other hand, the label space embedding approach transform the label vectors to an embedded space via linear or local

non-linear embeddings, followed by the association between feature vectors and embedded label space for classification purpose. With proper decoding process that maps the projected data back to the original label space, the task of multi-label prediction is achieved (41; 94; 109). We present a brief review of the FE and LE approaches for multi-label classification. The detailed discussion is given in Chapter 5.

Given N training examples in the form of a pair of feature matrix X and label matrix Y where each example $x_i \in \mathbb{R}^D, 1 \leq i \leq N$, is a row of X and its associated labels $y_i \in \{-1, +1\}^L$ is the corresponding row of Y, the goal of FE is to learn a transformation matrix $W \in \mathbb{R}^{D \times L}$ which maps instances feature space to label space. This approach requires $D \times L$ parameter to model the classification problem, which will be expensive when D and L are large (136). In (136) a generic empirical risk minimization (ERM) framework is used with low-rank constraint on linear parametrization $W = UV^T$, where $U \in \mathbb{R}^{D \times d}$ and $V \in \mathbb{R}^{L \times d}$ are of rank $d \ll D$. The problem can be restated as follows.

$$\min_{U,V} \ell(Y, XUV^T) + \frac{\lambda}{2}(\|U\|_F^2 + \|V\|_F^2) \tag{2.17}$$

where $\| \cdot \|_F$ is Frobenius norm. The formulation in Eq. (2.17) can capture the intrinsic information of both feature and label space. It can also be seen as a joint learning framework in which dimensionality reduction and multi-label classification are performed simultaneously (51; 137).

The matrix factorization (MF) based approach for LE aims at determining two matrices $U \in \mathbb{R}^{N \times d}$ and $V \in \mathbb{R}^{d \times L}$ (5; 131). The matrix U can be viewed as the *basis matrix*, while the matrix V can be treated as the *coefficient matrix* and a common formulation is the following optimization problem.

$$\min_{U,V} \ell(Y, U, V) + \lambda R(U, V) \tag{2.18}$$

where $\ell(\cdot)$ is a loss function that measures how well UV approximates Y, $R(\cdot)$ is a regularization function that promotes various desired properties in U and V (sparsity, group-sparsity, etc.) and the constant $\lambda \geq 0$ is the regularization parameter which controls the extent of regularization. In (76), a MF based approach is used to learn the label encoding and decoding matrix simultaneously. The problem is formulated as

follows.

$$\min_{U,V} \|Y - UV\|_F^2 + \alpha \Psi(X, U) \tag{2.19}$$

where $U \in \mathbb{R}^{N \times d}$ is the code matrix, $V \in \mathbb{R}^{d \times L}$ is the decoding matrix, $\Psi(X, U)$ is used to make U feature-aware by considering correlations between X and U as side information and the constant $\alpha \geq 0$ is the trade-off parameter.

CHAPTER 3

Collaborative Filtering Using Hierarchical Matrix Factorizations

In the previous chapter, we discussed the basic matrix factorization model and the associated formulation of matrix factorization as an optimization problem. We have also discussed at length the application of matrix of factorization techniques in collaborative filtering and multi-label classification. In this chapter, we elaborate on those basics and propose a matrix factorization based collaborative filtering approach.

3.1   Introduction

In this chapter, we describe the proposed method, termed as *HMF (Hierarchical Matrix Factorization)*. HMF is a novel method for constructing a hierarchical bi-level maximum margin matrix factorization to handle matrix completion of ordinal rating matrix. The proposed method draws motivation from research on multi-class classification. There are two major approaches of extending two-class classifiers to multi-class classifiers. The first approach explicitly reformulates the classifier, resulting in a unified multiclass optimization problem (*embedded technique*). The second approach (*combinational technique*) is to decompose a multiclass problem into multiple, independently trained, binary classification problems and to combine them appropriately so as to form a multiclass classifier. In maximum margin matrix factorization (MMMF) (97), the authors adopted *embedded* approach by extending bi-level hinge loss to multi-level cases.

Combinational techniques have been very popular and successful as they all exhibit some relative advantages over embedded techniques and this prompts us to question whether some sort of combinational approach can be employed in the context of MMMF. There are several approaches in combinational techniques and these are *One-Against-All* (OAA) (40; 99; 128), *One-Against-One* (OAO) (22; 40) etc. HMF falls into the category of OAA approach with a difference. The OAA approach of classification employs one binary classifier for each class against all other classes. In the present context 'class' corresponds to the number of ordinal ranks. Interestingly, since the ranks are ordered, in the present case, OAA strategy is used by taking advantage of the ordering of 'classes'. Unlike the traditional OAA strategy, (which means that one bi-level matrix factorization to be used for rank q as one label (say, -1) and all other ranks as the other label (say, $+1$)), here we employ for each rank q, all ranks below q as -1 and all ranks above q as $+1$.

MMMF and HMF, like any other matrix factorization methods, use latent factor model approach. The latent factor model is concerned with learning from limited observations, latent factor vector U_i for each user and latent factor vector V_j for each item, such that the dot product of U_i and V_j gives the ranking of user i for item j. In MMMF, besides learning the latent factors, the set of thresholds also needs to be learned. It is assumed that there are $R - 1$ thresholds $\theta_{i,r}$ for each user i (R is the number of ordinal values or number of ranks). Thus, the rating of user i for item j is decided by comparing the dot product of U_i and V_j with each $\theta_{i,r}$. Thus, the fundamental assumption of MMMF is that the latent factor vectors determine the properties of users and items and the threshold values capture the characteristics of rating. HMF differs from MMMF on this principle. The underlying principle of HMF is that the latent factors of users and items are going to be different, if the users' *likes* or *dislikes* cutoff thresholds are different. The latent factors are different according to situations. For instance, the latent factors when ranks above q are identified as *likes* is different from situations wherein the ranks above $q + 1$ are identified as *likes*. Thus if we have R ratings then, there will be $R - 1$ pairs of latent factors (U^q, V^q). Unlike the process of learning single pair of latent factors, U and V, HMF learns several U's and V's in this process without any additional computational overheads. The process is proved to be a more accurate matrix completion process. There has not been any attempt in this direction and we believe that our present study will open up new areas of research in future.

The rest of the chapter is organized as follows. In Section 3.2, we describe the bi-level MMMF which we use subsequently in our algorithm. Section 3.3 summarizes the well-known existing MMMF method. We introduce our proposed method of Hierarchical Matrix Factorization (HMF) in Section 3.4. The advantages of HMF over other matrix factorization based collaborative filtering methods are given by detailed experimental analysis in Section 3.6. Section 3.7 concludes the chapter.

3.2 Bi-level MMMF

In this section we describe Maximum Margin Matrix Factorization for a bi-level rating matrix. The matrix completion of bi-level matrix is concerned with the following problem.

Problem $P_{\pm 1}(Y)$: Let Y be a $N \times M$ partially observed user-item rating matrix and Ω is the set of observed entries. For each $(i, j) \in \Omega$, the entry $y_{ij} \in \{-1, +1\}$ defines the preference of ith user for jth item with $+1$ for *likes* and -1 for *dislikes*. For each $(i, j) \notin \Omega$, the entry $y_{ij} = 0$ indicates that the preference of ith user for jth item is not available. Given a partially observed rating matrix $Y \in \mathbb{R}^{N \times M}$, the goal is to infer y_{ij} for $(i, j) \notin \Omega$.

Matrix factorization is one of the major techniques employed for any matrix completion problem. In this line, the above problem can be rephrased using latent factors. Given a partially observed rating matrix $Y \in \mathbb{R}^{N \times M}$ and the observed preference set Ω, matrix factorization aims at determining two low-rank (or, low-norm) matrices $U \in \mathbb{R}^{N \times d}$ and $V \in \mathbb{R}^{M \times d}$ such that $Y \approx UV^T$. The row vectors U_i, $1 \leq i \leq N$ and V_j, $1 \leq j \leq M$ are the d-dimensional representations of the users and the items, respectively. A common formulation of $P_{\pm 1}(Y)$ is to find U and V as solution of the following optimization problem.

$$\min_{U,V} J(U, V) = \sum_{(i,j) \in \Omega} \ell(y_{ij}, U_i, V_j) + \lambda R(U, V) \tag{3.1}$$

where $\ell(\cdot)$ is a loss function that measures how well $(U_i . V_j^T)$ approximates y_{ij}, and $R(\cdot)$ is a regularization function. The idea of the above formulation is to alleviate the problem of outliers through a robust loss function and at the same time to avoid overfitting and

to make the optimization function smooth with the use of regularization function.

A number of approaches can be used to optimize the objective function 3.1. Gradient Descent method and its variants start with random U and V and iteratively update U and V using the equations 3.2 and 3.3, respectively.

$$U_{ip}^{t+1} = U_{ip}^t - c \frac{\partial J}{\partial U_{ip}^t} \tag{3.2}$$

$$V_{jq}^{t+1} = V_{jq}^t - c \frac{\partial J}{\partial V_{jq}^t} \tag{3.3}$$

where c is the step length parameter and suffixes t and $(t + 1)$ indicate current values and updated values.

The step-wise description of the process is given as Pseudo-code in Algorithm 1.

Algorithm 1: $A_{\pm 1}$ (Y, d, λ)

---

input : Bi-level Rating Matrix: Y, Number of Latent Factors: d, Regularization Parameter: λ

output: Factor Matrices: U and V

$t \leftarrow 0$;
Initialize: U^t, V^t;
while *Stopping criteria met* **do**
    Calculate $\frac{\partial J}{\partial U_{ip}^t}$ and $\frac{\partial J}{\partial V_{jq}^t}$ at U^t and V^t, respectively;
    $t \leftarrow t + 1$;
    $U_{ip}^{t+1} \leftarrow U_{ip}^t - c \frac{\partial J}{\partial U_{ip}^t}$;
    $V_{jq}^{t+1} \leftarrow V_{jq}^t - c \frac{\partial J}{\partial V_{jq}^t}$;
end
Let U and V are the factor matrices obtained after convergence;
return U and V;

Once U and V are computed by Algorithm 1, the matrix completion process is accomplished from the factor matrices as follows.

$$\hat{y}_{ij} = \begin{cases} -1, & \text{if } (i, j) \notin \Omega \wedge U_i V_j^T < \theta; \\ +1, & \text{if } (i, j) \notin \Omega \wedge U_i V_j^T \geq \theta; \\ y_{ij}, & \text{if } (i, j) \in \Omega, \end{cases} \tag{3.4}$$

where θ is the user-specified threshold value.

The latent factor V_j of each item j can be viewed as a point in d-dimensional space and the latent factor U_i of user i can be viewed as a decision hyperplane in this space. The objective of bi-level matrix factorization is to learn the embeddings of these points and hyperplanes in $\mathbb{R}^d$ such that each hyperplane (corresponding to a user) separates (or, equivalently, classifies) the items by *likes* and *dislikes* of a user. Let us consider the following partially observed bi-level matrix Y (Table 3.1) for illustration. The unobserved entries are indicated by 0 entries.

Table 3.1: Bi-level matrix Y

0	1	0	0	1	0	-1
-1	0	1	1	1	0	-1
0	1	-1	0	0	1	-1
-1	1	-1	1	-1	1	0
-1	0	-1	1	1	0	0
0	-1	0	1	0	1	1
1	1	0	-1	0	0	0

Let us assume that the following U and V are the latent factor matrices with $d = 2$.

Table 3.2: Latent factor matrices corresponding to Y.

-0.63	-0.50	-0.37	0.94
-0.69	-0.96	-0.70	-1.02
0.27	-1.09	-1.06	0.42
0.63	-0.84	0.55	-0.97
-0.02	-1.03	-1.04	-0.36
1.19	-0.03	0.67	-0.58
-1.11	0.21	0.65	0.81
U		V	

The same is depicted graphically in Figure 3.1. $V_1, V_2, \ldots, V_7$ are depicted as points. The hyperplanes $U_1, U_2, \ldots, U_7$ with threshold 0.1 are depicted as lines. An arrow is shown to indicate the *positive* side of each line. In other words, if a point falls this side, the preference is *like* $(+1)$ and the other side preference is *dislikes* (-1). Entry y_{34} is predicted based on the position of V_4 with respect to U_3. V_4 falls to the *positive* side of line corresponding to U_3 and hence, the entry y_{34} is predicted as $+1$. The latent factor U and V are obtained by a learning process making use of the generic algorithm (Algorithm 1) with the observed entries as the training set. The objective of this learning process is to minimize the loss due to discrepancy between computed entries and the observed entries. In the example (Figure 3.1) point V_3 and V_5 are in the *positive* side of U_2 and V_7 in the *negative* side of U_2. These points are located at different distance.
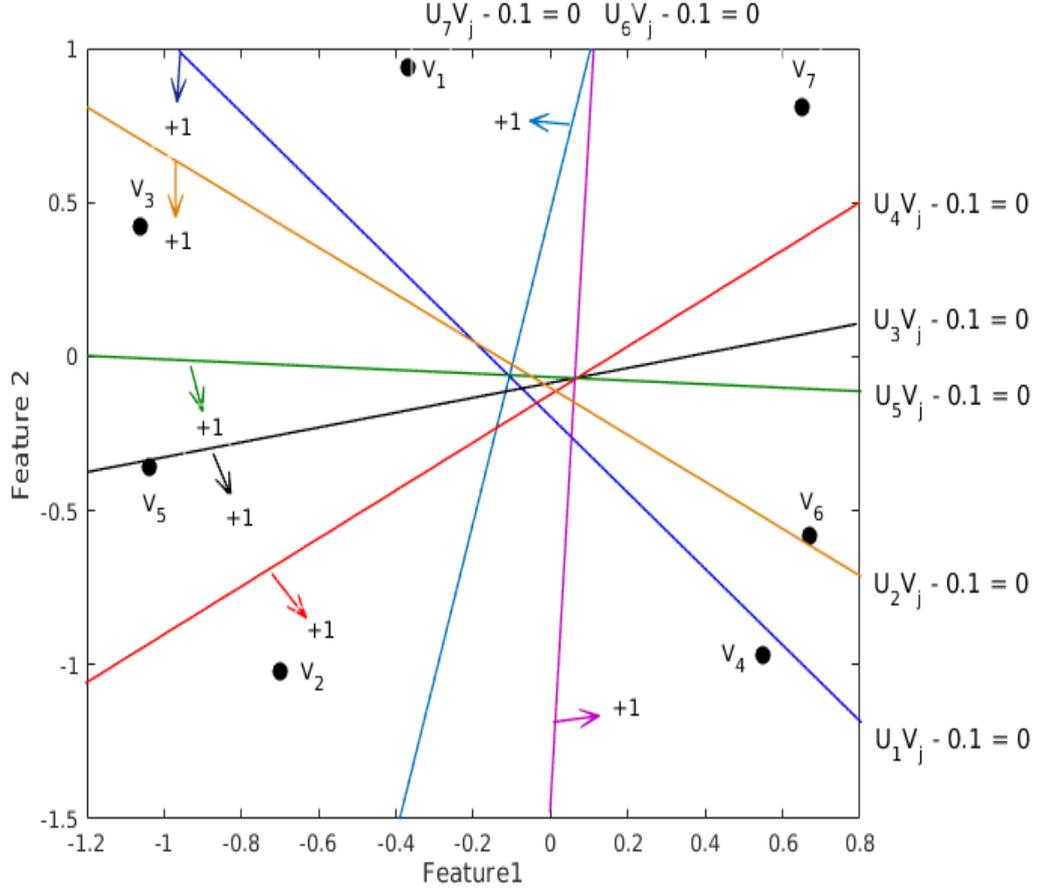
Figure 3.1: Geometrical interpretation of bi-level matrix factorization.

There are many matrix factorization algorithms for bi-level matrices which adopts the generic algorithm describe in Algorithm 1. These algorithms are designed based on the specification of the loss function and the regularization function. We adopt here a maximum-margin formulation as the guiding principle.

Loss functions in matrix factorization models are used to measure the discrepancy between the observed entry and the corresponding prediction. Furthermore, especially when predicting discrete values such as ratings, loss functions other then sum-squared loss are often more appropriate (97). The trade-off between generalization loss and empirical loss has been prevailing since the advent of support vector machine (SVM). Maximum margin approach aims at providing higher generalization ability and avoiding overfitting. In this context, hinge loss function is the most preferred loss function and is defined as follows.

$$
h(z) = \begin{cases} 0, & \text{if } z \geq 1; \\ 1 - z, & \text{otherwise}, \end{cases}
\tag{3.5}
$$

where $z = y_{ij}(U_i V_j^T)$. The hinge loss is illustrated in Figure 3.2.

26
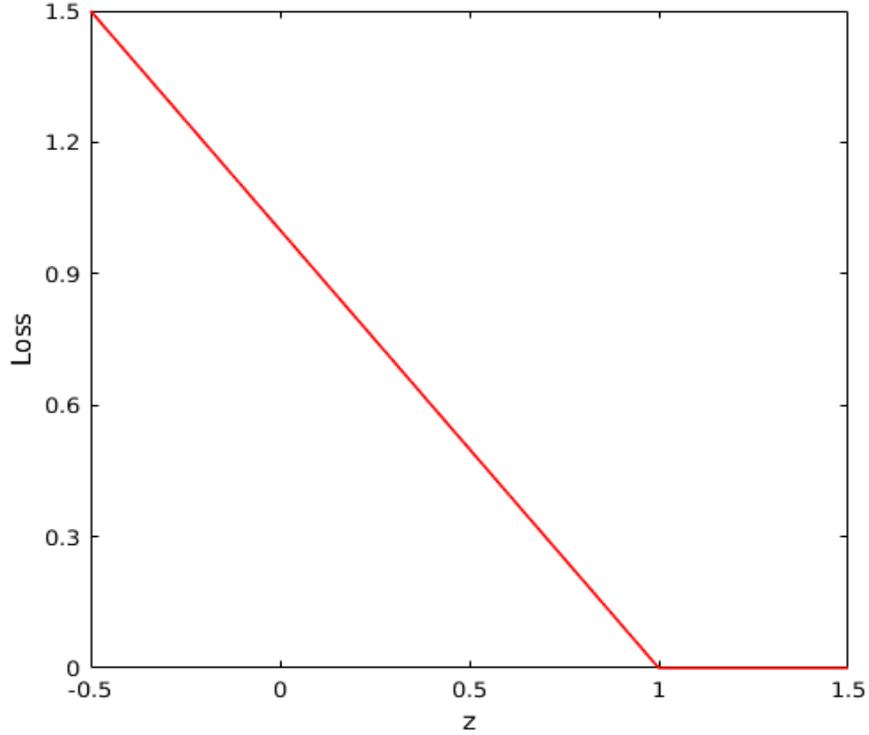
Figure 3.2: Loss function values for the hinge

The following real-valued prediction matrix is obtained from the latent factor matrices U and V (Table 3.2) corresponding to the matrix Y (Table 3.1).

Table 3.3: Real-valued prediction corresponding to bi-level matrix Y.

-0.24	0.95	0.46	0.14	0.84	-0.13	-0.81
-0.65	1.46	0.33	0.55	1.06	0.09	-1.23
-1.12	0.92	-0.74	1.21	0.11	0.81	-0.71
-1.02	0.42	-1.02	1.16	-0.35	0.91	-0.27
-0.96	1.06	-0.41	0.99	0.39	0.58	-0.85
-0.47	-0.80	-1.27	0.68	-1.23	0.81	0.75
0.61	0.56	1.26	-0.81	1.08	-0.87	-0.55

The hinge loss function values corresponding to the observed entries in Y (Table 3.1) and the real-valued prediction (Table 3.3) is shown in Table 3.4.

Table 3.4: Hinge loss corresponding to observed entries in Y.

0	0.05	0	0	0.16	0	0
0.35	0	0.67	0.45	0	0	0.35
0	0.08	0.26	0	0	0.19	0
0	0.58	0	0	0.65	0.09	0
0.04	0	0.59	0.01	0.61	0	0.04
0	0.20	0	0.32	0	0.19	0
0.39	0.44	0	0.19	0	0	0.39

Hinge loss, $h(z)$ is non-differentiable at $z = 1$ and is very sensitive to outliers as mentioned in (97). Therefore an alternative called *smooth hinge loss* is proposed in (98) and can be defined as,

$$h(z) = \begin{cases} 0, & \text{if } z \geq 1; \\ \frac{1}{2}(1 - z)^2, & \text{if } 0 < z < 1; \\ \frac{1}{2} - z, & \text{otherwise.} \end{cases} \quad (3.6)$$

The smooth hinge loss is illustrated in Figure 3.3.
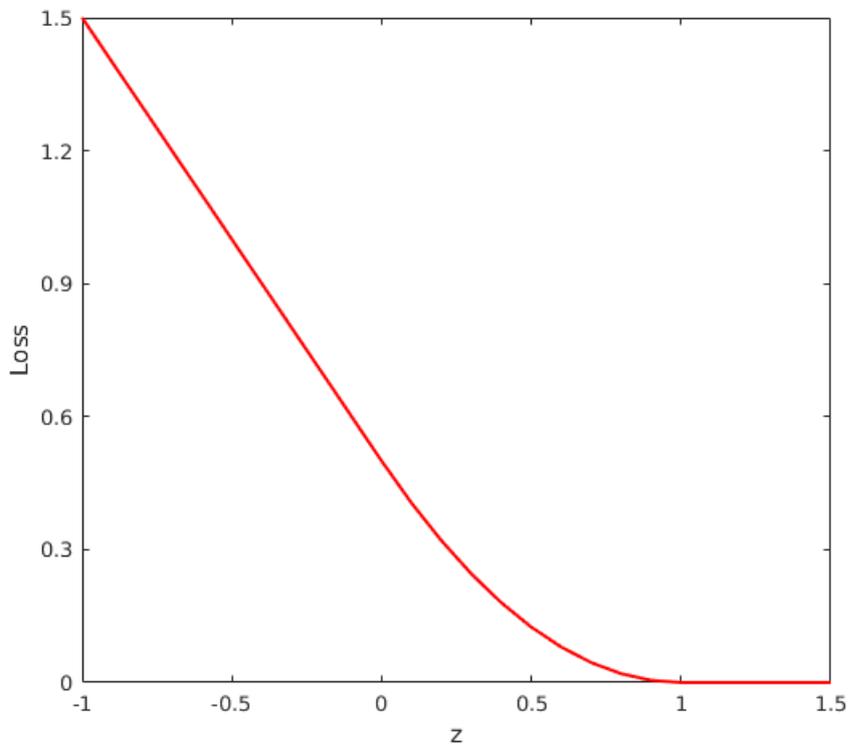
Figure 3.3: Loss function values for the smooth hinge

The smooth hinge loss function values corresponding to the observed entries in Y (Table 3.1) and the real-valued prediction (Table 3.3) is shown in Table 3.5.

Table 3.5: Smooth hinge loss corresponding to observed entries in Y.

0	0	0	0	0.01	0	0
0.06	0	0.23	0.10	0	0	0.06
0	0	0.03	0	0	0.02	0
0	0.17	0	0	0.21	0	0
0	0	0.17	0	0.19	0	0
0	0.02	0	0.05	0	0.02	0
0.08	0.10	0	0.02	0	0	0.08

28

Figure 3.2 and 3.3 show the loss function values for the hinge and smooth hinge loss, respectively. It can be seen that the smooth hinge loss shares important similarities to the hinge loss and has a smooth transition between a zero slope and a constant negative slope. Table 3.4 and 3.5 show the hinge loss and smooth hinge loss function values corresponding to the observed entries in Y (Table 3.1). It can also be seen that the smooth hinge is less sensitive to the outliers as compared to the hinge loss function. For example, the rating given by user 4 for item 2 is positive and the same reflect in the embedding (Figure 3.1). The loss incurred by hinge and smooth hinge are 0.58 and 0.17, respectively. Even though the point is embedded with margin the hinge loss function gives more reward to the model for the increase in objective value.

We reformulate $P_{\pm1}(Y)$ problem for the bi-level rating matrix as the following optimization problem.

$$\min_{U,V} J(U,V) = \sum_{(i,j)\in\Omega} h\big(y_{ij}(U_i V_j^T)\big) + \frac{\lambda}{2}(\|U\|_F^2 + \|V\|_F^2) \qquad (3.7)$$

where $\|.\|_F$ is the Frobenius norm which is the same as defined in Chapter 2, $\lambda > 0$ is the regularization parameter and $h(\cdot)$ is the smooth hinge loss function as defined previously.

The gradients of the variables to be optimized are determined as follows. The gradient with respect to each element of U is calculated as

$$
\begin{aligned}
\frac{\partial J}{\partial U_{ip}} &= \sum_{j|(i,j)\in\Omega} \frac{\partial h\big(y_{ij}(U_i V_j^T)\big)}{\partial U_{ip}} + \frac{\lambda}{2}\left(\frac{\partial \|U\|_F^2}{\partial U_{ip}} + \frac{\partial \|V\|_F^2}{\partial U_{ip}}\right) \\
&= \sum_{j|(i,j)\in\Omega} y_{ij} h'\big(y_{ij}(U_i V_j^T)\big)\frac{\partial(U_i V_j^T)}{\partial U_{ip}} + \lambda U_{ip} \\
&= \sum_{j|(i,j)\in\Omega} y_{ij} h'\big(y_{ij}(U_i V_j^T)\big) V_{jp} + \lambda U_{ip} \qquad (3.8)
\end{aligned}
$$

Similarly, the gradient with respect to each element of V is calculated as follows.

$$\frac{\partial J}{\partial V_{jq}} = \sum_{i|(i,j)\in\Omega} y_{ij} h'(y_{ij}(U_i V_j^T))U_{iq} + \lambda V_{jq} \qquad (3.9)$$

where $h'(z)$ is defined as follows.

$$h'(z) = \begin{cases} 0, & \text{if } z \geq 1; \\ z - 1, & \text{if } 0 < z < 1; \\ -1, & \text{otherwise.} \end{cases} \tag{3.10}$$

Algorithm 2 outlines the main flow of the bi-level maximum margin matrix factorization (BMMMF).

Algorithm 2: BMMMF (Y, d, λ)

input : Bi-level Rating Matrix: Y, Number of Latent Factors: d, Regularization Parameter: λ

output: Factor Matrices: U and V

$t \leftarrow 0$;
Initialize: U^t, V^t;
while *Stopping criteria met* **do**
$\quad \frac{\partial J}{\partial U_{ip}^t} \leftarrow \sum_{j|(i,j)\in\Omega} y_{ij} h'(y_{ij}(U_i^t V_j^{t^T})) V_{jp}^t + \lambda U_{ip}^t$;
$\quad \frac{\partial J}{\partial V_{jq}^t} \leftarrow \sum_{i|(i,j)\in\Omega} y_{ij} h'(y_{ij}(U_i^t V_j^{t^T})) U_{iq}^t + \lambda V_{jq}^t$;
$\quad t \leftarrow t + 1$;
$\quad U_{ip}^{t+1} \leftarrow U_{ip}^t - c\frac{\partial J}{\partial U_{ip}^t}$;
$\quad V_{jq}^{t+1} \leftarrow V_{jq}^t - c\frac{\partial J}{\partial V_{jq}^t}$;
end
Let U and V are the factor matrices obtained after convergence;
return U and V;

We also show the behaviour of J (Equation 3.7). We plot the value of J obtained in every iteration for the same Y (Table 3.1) starting from different initial points. It can be seen from Figure 3.4 that J is having asymptotic convergence.

3.3   Multi-level MMMF

As discussed in the previous chapter, MMMF (107) and subsequently, Fast MMMF (97) are proposed primarily for collaborative filtering with ordinal rating matrix when user-preferences are not in the form of like/ dislike but are values in a discrete range. The matrix completion of ordinal rating matrix is concerned with the following problem.

Problem P_{ord}(Y): Let Y be a $N \times M$ partially observed user-item rating matrix and Ω
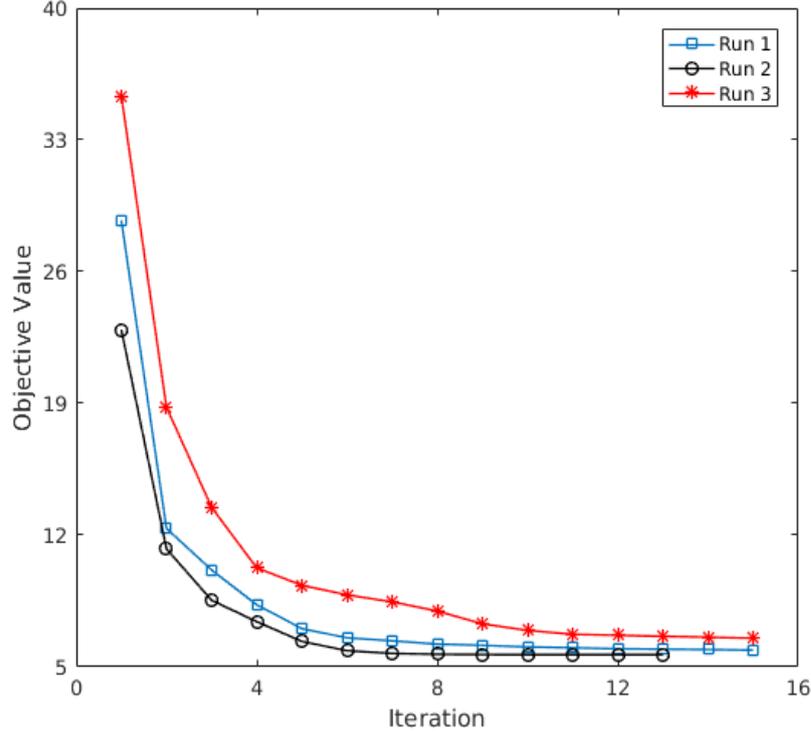
Figure 3.4: Convergence graph of BMMMF with different initial points

is the set of observed entries. For each $(i, j) \in \Omega$, the entry $y_{ij} \in \{1, 2, \ldots, R\}$ defines the preference of the ith user for the jth item. For each $(i, j) \notin \Omega$, $y_{ij} = 0$ indicates that the preference of ith user for jth item is not available. Given a partially observed rating matrix $Y \in \mathbb{R}^{N \times M}$, the goal is to predict y_{ij} for $(i, j) \notin \Omega$.

Need for multiple thresholds: Unlike $\text{P}_{\pm 1}(\text{Y})$, $\text{P}_{\text{ord}}(\text{Y})$ has domain of y_{ij} with more than two values. When the domain has two values, $\text{P}_{\text{ord}}(\text{Y})$ is equivalent to $\text{P}_{\pm 1}(\text{Y})$. Continuing our discussion on geometrical interpretation of $\text{P}_{\pm 1}(\text{Y})$, the *likes* $(+1)$ and *dislikes* (-1) of user i are separated with both sides of the hyperplane defined by U_i. The same concept when extended to $\text{P}_{\text{ord}}(\text{Y})$, it is necessary to define threshold values $\{\theta_1, \ldots, \theta_{R-1}\}$ such that the region between two parallel hyperplanes defined by the same U_i with different threshold θ_{q-1} and θ_q corresponds to the rating q. Thus, in $\text{P}_{\text{ord}}(\text{Y})$, in addition to learning the latent factor matrices U and V it is also needed to learn the thresholds θ's. There may be a debate on the number of θ's needed, but following the original proposal of MMMF (97), we follow $R - 1$ thresholds for each user and hence there are $n(R - 1)$ thresholds.

There are different approaches for constructing a loss function based on a set of thresholds such as immediate-threshold and all-threshold (98). For each observed entry

and the corresponding prediction pair $(y_{ij}, U_iV_j^T)$, the immediate-threshold based approach calculates the loss as sum of immediate-threshold violation which is $\ell(U_iV_j^T - \theta_{i,y_{ij}-1}) + \ell(\theta_{i,y_{ij}} - U_iV_j^T)$. On the other hand, all-threshold loss is calculated as the sum of loss for all thresholds which is the cost of crossing multiple rating-boundaries and is defined as follows.

$$\ell(U_iV_j^T, y_{ij}) = \sum_{r=1}^{y_{ij}-1} \ell(U_iV_j^T - \theta_{i,r}) + \sum_{r=y_{ij}}^{R-1} \ell(\theta_{i,r} - U_iV_j^T) \tag{3.11}$$

Using the $R-1$ thresholds θ's, MMMF extended the hinge loss function meant for binary classification to ordinal settings. The difference between immediate-threshold and all-threshold hinge is illustrated with the help of the following example. Let us consider the partially observed ordinal rating matrix Y with $R=5$, the learnt factor matrices U, V and the set of thresholds θ's for each user (Table 3.6).

Table 3.6: Runnning example.

1	2	4	3	5	2	1	4	3	0
0	4	5	3	2	4	1	1	3	2
4	3	5	5	2	0	2	3	1	4
5	0	1	2	3	1	4	3	5	2
2	5	3	4	2	4	0	3	1	1
0	1	5	3	1	5	3	2	4	4
4	2	5	1	4	3	2	5	0	3
2	2	3	1	3	4	5	0	5	4
5	4	5	2	3	1	4	0	1	2
3	0	4	5	1	2	1	5	3	4

Y

0.22	-0.12
0.90	0.05
1.45	0.31
-1.52	0.04
0.94	1.06
1.09	-1.45
0.10	0.14
-0.60	-1.65
-0.24	0.71
0.48	0.59

U

-0.79	1.01
0.10	1.21
1.51	-0.32
0.76	0.63
-0.53	0.24
1.45	-0.79
-0.98	-0.74
-0.17	0.72
-0.72	-1.32
0.39	-0.63

V

-0.61	-0.18	0.51	1.21
-0.74	0.09	0.69	1.41
-1.42	-0.65	0.28	0.91
-1.35	-0.25	0.74	0.96
-0.50	0.25	0.77	1.44
-1.08	-0.60	0.56	1.51
-1.10	-0.24	0.34	0.62
-1.41	-0.62	0.13	0.99
-0.73	-0.24	0.02	0.83
-0.73	-0.51	0.09	0.90

θ's

The following real-valued prediction matrix is obtained from the above latent factor matrices U and V corresponding to the matrix Y.

Table 3.7: Real-valued prediction corresponding to ordinal rating matrix Y.

-0.30	-0.12	0.37	0.09	-0.15	0.41	-0.13	-0.12	0.00	0.16
-0.66	0.15	1.34	0.72	-0.47	1.27	-0.92	-0.12	-0.71	0.32
-0.83	0.52	2.09	1.30	-0.69	1.86	-1.65	-0.02	-1.45	0.37
1.24	-0.10	-2.31	-1.13	0.82	-2.24	1.46	0.29	1.04	-0.62
0.33	1.38	1.08	1.38	-0.24	0.53	-1.71	0.60	-2.08	-0.30
-2.33	-1.65	2.11	-0.09	-0.93	2.73	0.00	-1.23	1.13	1.34
0.06	0.18	0.11	0.16	-0.02	0.03	-0.20	0.08	-0.26	-0.05
-1.19	-2.06	-0.38	-1.50	-0.08	0.43	1.81	-1.09	2.61	0.81
0.91	0.84	-0.59	0.26	0.30	-0.91	-0.29	0.55	-0.76	-0.54
0.22	0.76	0.54	0.74	-0.11	0.23	-0.91	0.34	-1.12	-0.18

One can see that the entry y_{13} is observed as 4 and the corresponding real-valued prediction is 0.37. When immediate-threshold hinge is the loss measure, the overall loss is calculated as follows.

$$
\begin{aligned}
h(y_{13}, U_1 V_3^T) &= h(U_1 V_3^T - \theta_{i,3}) + h(\theta_{i,4} - U_1 V_3^T) \\
&= h(0.37 - 0.51) + h(1.21 - 0.37) \\
&= h(-0.14) + h(0.84) \\
&= 0.65
\end{aligned}
$$

where $h(\cdot)$ is the smooth hinge loss function as defined previously. For the same example, the overall loss with all-threshold hinge function is calculated as follows.

$$
\begin{aligned}
h(y_{13}, U_1 V_3^T) &= h(U_1 V_3^T - \theta_{i,1}) + h(U_1 V_3^T - \theta_{i,2}) + h(U_1 V_3^T - \theta_{i,3}) + h(\theta_{i,4} - U_1 V_3^T) \\
&= h(0.37 + 0.61) + h(0.37 + 0.18)) + h(0.37 - 0.51) + h(1.21 - 0.37) \\
&= h(0.98) + h(0.55) + h(-0.14) + h(0.84) \\
&= 0.75
\end{aligned}
$$

Continuing our discussion on geometrical interpretation, immediate-threshold loss tries to embed the point V_3 into the region defined by the parallel hyperplanes $(U_1, \theta_{i,3})$ and $(U_1, \theta_{1,4})$ which basically mean that $U_1 V_3^T - \theta_{1,3} > 0$ and $U_1 V_3^T - \theta_{1,4} < 0$. The all-threshold hinge function not only tries to embed the points rated as r into the region defined by (U_i, θ_{r-1}) and (U_i, θ_r) but also consider the position of the points with respect to other hyperplanes. It is also desirable that every point V_j rated by user i should satisfy the condition $U_i V_j^T - \theta_{i,r-1} > 0$ for $r < y_{ij}$ and $U_i V_j^T - \theta_{i,r} < 0$ for $r \geq y_{ij}$.

In MMMF (97), each hyperplane acts as a maximum-margin separator which is ensured by considering smooth hinge as the loss function (all-threshold hinge). The resulting optimization problem for $\text{P}_{\text{ord}}(\text{Y})$ is

$$
\min_{U,V} J(U, V, \theta) \sum_{(i,j) \in \Omega} \left(\sum_{r=1}^{y_{ij}-1} h(U_i V_j^T - \theta_{i,r}) + \sum_{r=y_{ij}}^{R-1} h(\theta_{i,r} - U_i V_j^T) \right) + \frac{\lambda}{2} (\|U\|_F^2 + \|V\|_F^2)
$$

$$(3.12)$$

where $\|.\|_F$ is the Frobenius norm, $\lambda > 0$ is the regularization parameter, Ω is the set of observed entries, $h(z)$ is the smooth hinge loss as defined previously and $\theta_{i,r}$ is the

threshold for rank r of user i. The equation given above can be rewritten as follows.

$$\min_{U,V} J(U,V,\theta) \sum_{r=1}^{R-1} \sum_{(i,j)\in\Omega} h\big(T_{ij}^r(\theta_{i,r} - U_iV_j^T)\big) + \frac{\lambda}{2}(\|U\|_F^2 + \|V\|_F^2) \qquad (3.13)$$

where T is defined as

$$T_{ij}^r = \begin{cases} +1, & \text{if r} \geq y_{ij}; \\ -1, & \text{if r} < y_{ij}. \end{cases}$$

The gradients of the variables to be optimized are determined as follows. The gradient with respect to each element of U is calculated as follows.

$$\begin{aligned} \frac{\partial J}{\partial U_{ip}} &= \sum_{r=1}^{R-1} \sum_{j|(i,j)\in\Omega} \frac{\partial h\big(T_{ij}^r(\theta_{i,r} - U_iV_j^T)\big)}{\partial U_{ip}} + \frac{\lambda}{2}\left(\frac{\partial\|U\|_F^2}{\partial U_{ip}} + \frac{\partial\|V\|_F^2}{\partial U_{ip}}\right) \\ &= \sum_{r=1}^{R-1} \sum_{j|(i,j)\in\Omega} T_{ij}^r h'\big(T_{ij}^r(\theta_{i,r} - U_iV_j^T)\big)\frac{\partial(\theta_{i,r} - U_iV_j^T)}{\partial U_{ip}} + \lambda U_{ip} \\ &= \lambda U_{ip} - \sum_{r=1}^{R-1} \sum_{j|(i,j)\in\Omega} T_{ij}^r h'\big(T_{ij}^r(\theta_{i,r} - U_iV_j^T)\big)V_{jp} \qquad (3.14) \end{aligned}$$

where $h'(\cdot)$ is the same as defined previously. Similarly, the gradients with respect to each element of V is calculated as

$$\frac{\partial J}{\partial V_{jq}} = \lambda V_{jq} - \sum_{r=1}^{R-1} \sum_{i|(i,j)\in\Omega} T_{ij}^r h'\big(T_{ij}^r(\theta_{i,r} - U_iV_j^T)\big)U_{iq} \qquad (3.15)$$

and the gradient with respect to each $\theta_{i,r}$ is determined as follows.

$$\begin{aligned} \frac{\partial J}{\partial \theta_{i,r}} &= \sum_{j|(i,j)\in\Omega} \frac{\partial h\big(T_{ij}^r(\theta_{i,r} - U_iV_j^T)\big)}{\partial\theta_{i,r}} + \frac{\lambda}{2}\left(\frac{\partial\|U\|_F^2}{\partial\theta_{i,r}} + \frac{\partial\|V\|_F^2}{\partial\theta_{i,r}}\right) \\ &= \sum_{j|(i,j)\in\Omega} T_{ij}^r h'\big(T_{ij}^r(\theta_{i,r} - U_iV_j^T)\big)\frac{\partial(\theta_{i,r} - U_iV_j^T)}{\partial\theta_{i,r}} \\ &= \sum_{j|(i,j)\in\Omega} T_{ij}^r h'\big(T_{ij}^r(\theta_{i,r} - U_iV_j^T)\big) \qquad (3.16) \end{aligned}$$

Once U, V and θ's are computed, the matrix completion process is accomplished as
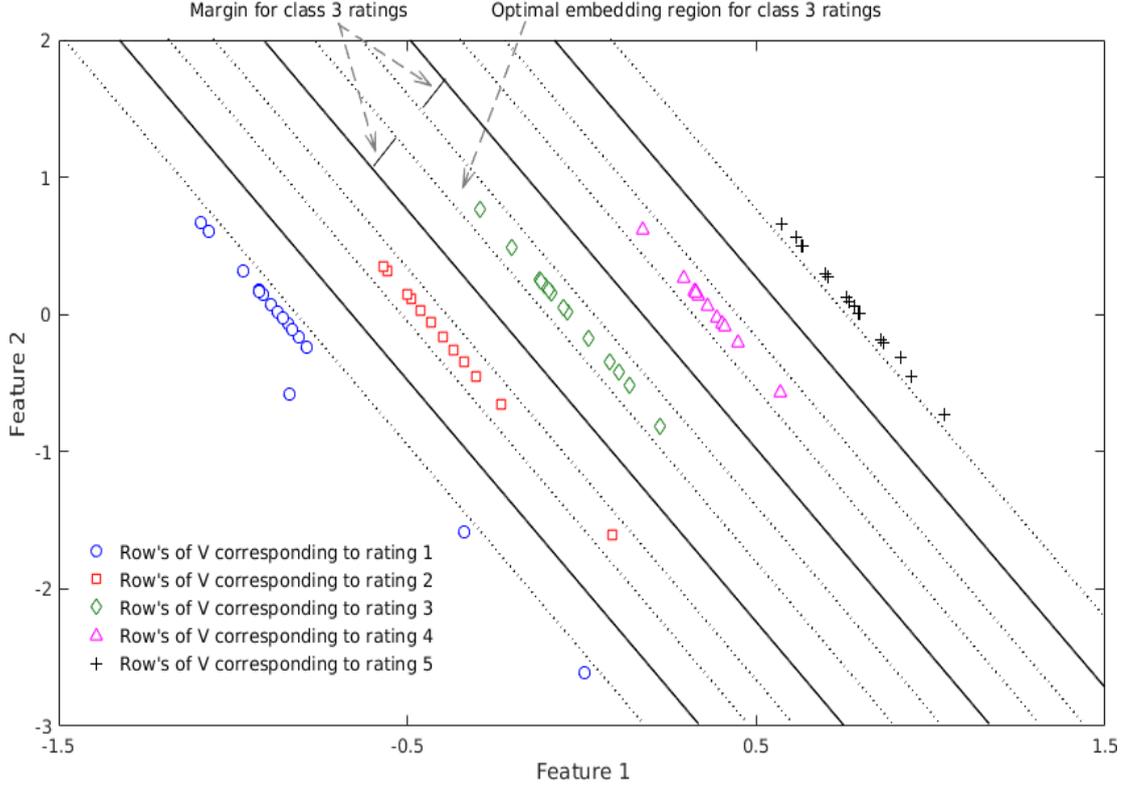
Figure 3.5: Classification by MMMF for the ith user

follows.

$$
\hat{y}_{ij} = \begin{cases} r, & \text{if } (i,j) \notin \Omega \wedge (\theta_{i,r} \leq x_{ij} \leq \theta_{i,r+1}) \wedge (0 \leq r \leq R-1); \\ y_{ij}, & \text{if } (i,j) \in \Omega, \end{cases}
$$

where, $\hat{y}_{ij}$ is the prediction for item j by user i. For simplicity of notation, we assume $\theta_{i,0} = -\infty$ and $\theta_{i,R} = +\infty$ for each user i.

The objective of MMMF is to learn the embedding of items as points, users as hyperplanes and ratings as thresholds such that the points fall as correctly as possible into regions with the margin of ratings defined by the decision hyperplane $(U_j, \theta_{j,r})$. We illustrate this concept by taking a synthetic dataset of size 5×1000 with 10% of observed entries. The number of ordinal ratings is 5 and $d = 2$. Figure 3.5 gives the decision hyperplane for a user and embedding of points corresponding to the items. Since $R = 5$, there are 4 hyperplanes that subdivide the entire space into 5 regions corresponding to 5 ratings. The data so chosen is balanced in the sense that the number of items for each rating is uniform. From the figure, it is clear that the embedding is learnt perfectly and the points corresponding to the items fall in the region of respective ratings with the margin. The margin is also shown in the figure by drawing parallel lines artifically.
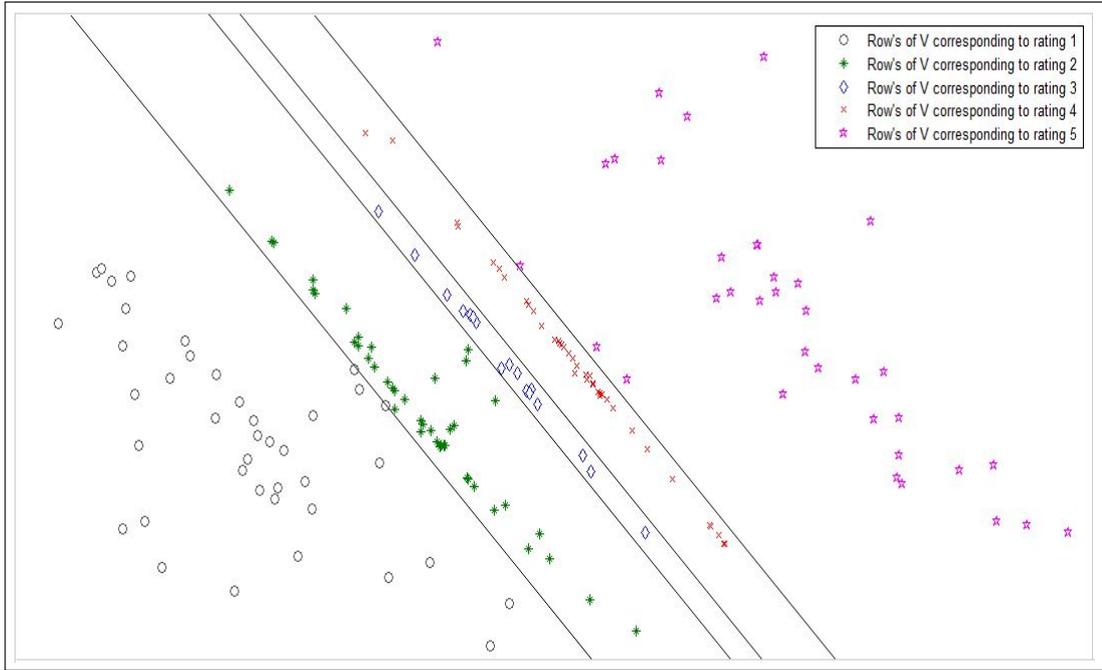
Figure 3.6: Classification by MMMF for ith user

Thus, one can see that the embedding is achieved with true sense of maximum-margin. However, this is not so for all cases. It is observed through several experiments that the margin for the parallel lines is not the same and these hyperplanes separate classes with unequal margin with bias toward classes with sparse training samples. When the data is balanced, as in the previous case (Figure 3.5), the unequal margin phenomenon is not evident. In another synthetic example of size 5×1000 with 20% of observed entries, $R = 5$, $d = 2$ and where the data is not balanced the inequalities in the margin size is evident. This is depicted in Figure 3.6. In this example the observed entries for rating $2, 3$ and 4 are substantially less than those for ratings 1 and 5. The observed entries for rating 3 is substantially less than the observed entries for ratings 2 and 4. When the points are embedded in two dimensional plane the parallel separating line corresponding to different thresholds for a given user have different margins and this is depicted in Figure 3.6. One can see that the hyperplanes tend to come closer towards the ratings with sparse samples. The hyperplane separating regions 4 and region 5 is not giving sufficient margin for rating 4 but overfits rating 5. The hyperplane separating regions 3 and 4 have unequal margin on both sides. So as a conclusion, it is observed that the proposed MMMF (97) is not truly handling margin maximization for predictions of ratings.

We also show the behaviour of J defined in Equation 3.13. We plot the value of

J obtained in every iteration for the same Y (Table 3.6) starting from different initial points. It can be seen from Figure 3.7 that J is having asymptotic convergence.
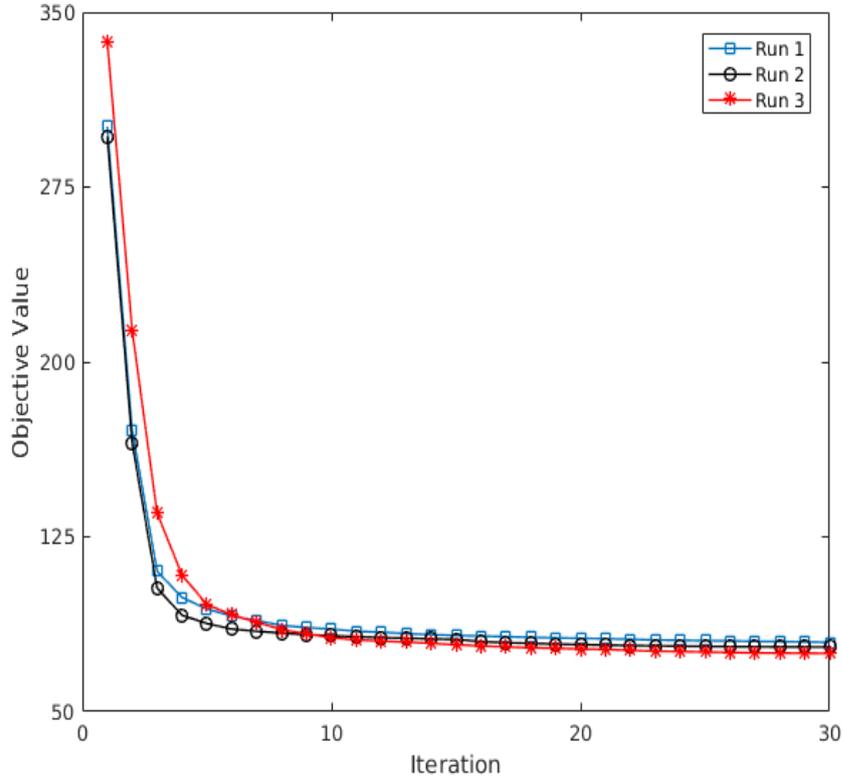
Figure 3.7: Convergence graph of MMMF with different initial points

For ordinal ratings with discrete values such as 1-5 stars, it is necessary to think of a sort of multiclass separation such that items are classified based on ratings which take on values from a finite discrete set. MMMF (97) follows the principle of single machine extension and handles ordinal rating matrix factorization by proposing a multi-class extension of the hinge loss function.

3.4   HMF- The Proposed Method

In this section, we describe the proposed method, termed as *HMF (Hierarchical Matrix Factorization)*. As discussed in Section 3.1, the proposed method draws motivation from research on multi-class classification. There are two major approaches for addressing multi-class classification problems namely *embedded techniques* (129; 71; 20; 8) and *combinational technique* (40; 99; 2). The first approach explicitly formulate the multi-class problem directly. The second approach (*combinational technique*) is to de-

compose the multi-class problem to multiple independent binary classification problems. In (MMMF) (97), an *embedded* approach is proposed to address the $P_{ord}(Y)$ problem. The authors has extended hinge loss defined for binary-level to multi-level cases. As discussed in Section 3.3, the extension of hinge-loss to multi-level does not preserve the property of maximum margin and also impose the parallel constraints on decision hyperplane. Our goal is to examine a solution to the $P_{ord}(Y)$ problem by constructing a hierarchy of bi-level MMMF using the principle of combinational approach. We believe that by following the principle of combinational approach the maximum margin advantage of bi-level matrix factorization can be retained and the parallel constraints on the decision hyperplane can also be relaxed.

As discussed previously, HMF is a stage-wise matrix completion technique that makes use of several bi-level MMMFs in a hierarchical fashion. At every stage q of HMF, the original problem $P_{ord}(Y)$ is converted to $P_{\pm 1}(Y)$ and solved using the method described in Section 3.2. The output at stage q is the learnt latent factor matrices U^q and V^q. The so computed latent factors U^q and V^q are used to predict the entries of the matrix of a specific ordinal value q. In other words, at Stage q , HMF attempts to complete only those unobserved entries of the rating matrix where the rating q is predicted. Thus for a R -level ordinal rating matrix, we use $R-1$ stages. The method of conversion from $P_{ord}(Y)$ to $P_{\pm 1}(Y)$ at stage q is discussed below.

Conversion of $P_{ord}(Y)$ to $P_{\pm 1}(Y)$: At every stage q of HMF, a BMMMF is employed and for this purpose, the $P_{ord}(Y)$ is converted to $P_{\pm 1}(Y)$ problem as follows.

$$
y_{ij}^q = \begin{cases} -1, & \text{if } (i,j) \in \Omega \wedge y_{ij} \leq q; \\ +1, & \text{if } (i,j) \in \Omega \wedge y_{ij} > q; \\ 0, & \text{if } (i,j) \notin \Omega. \end{cases} \tag{3.17}
$$

This conversion states that any rating above q is treated as $likes$ and below or equal to q is treated as $dislike$.

The outcome of BMMMF at stage q are the factor matrices U^q and V^q approximating Y^q. Let $\hat{Y}^q$ be the bi-level prediction matrix obtained from the factor matrices U^q and V^q and Ω^q is the index set corresponding to the -1 entries in $\hat{Y}^q$. For simplicity of notation, we assume Ω^0 is empty. The entries in ordinal rating prediction matrix $\hat{Y}$ are

predicted as q based on the following rule.

$$\hat{y}_{ij} = \begin{cases} q & \text{if } (i,j) \notin (\Omega^0 \cup \Omega^1 \cdots \cup \Omega^{q-1}) \wedge (\Omega^q = -1) \\ y_{ij} & \text{if } (i,j) \in \Omega \end{cases} \tag{3.18}$$

At the last stage, stage $R - 1$, after employing the above rule to predict entries with values $R - 1$, we complete the matrix by assigning R for all remaining unobserved entries.

Algorithm 3 outlines the main flow of the proposed method HMF for the ordinal rating matrix completion problem.

Algorithm 3: HMF $(Y, R, d, \theta, \lambda)$

---

input : Ordinal Rating Matrix: Y, Maximum Rating: R, Number of Latent
Factors: d, Threshold: θ, Regularization Parameters: $\{\lambda_1, \ldots, \lambda_{R-1}\}$
output: $\hat{Y}$

Initialize: $\hat{Y} \leftarrow 0$;
for $q = 1 \text{ to } R - 1$ **do**
   $Y^q \leftarrow 0$;
   for $(i,j) \in \Omega$ **do**
      if $y_{ij} \leq q$ **then**
         $y_{ij}^q = -1$;
      else
         $y_{ij}^q = +1$;
      end
   end
   $(U^q, V^q) \leftarrow \text{BMMMF}(Y^q, d, \lambda_q)$;
   for *ever user-item pair* (i,j) **do**
      if $U_i^q V_j^{q^T} < \theta$ **then**
         $\hat{y}_{ij}^q = \text{-1}$;
      else
         $\hat{y}_{ij}^q = \text{+1}$;
      end
   end
   for *ever user-item pair* (i,j) **do**
      if $\left((i,j) \notin \Omega \wedge \hat{y}_{ij}^q = -1\right) \vee y_{ij} = q$ **then**
         $\hat{y}_{ij} \leftarrow q$
      end
   end
end
$\hat{y}_{ij} \leftarrow R, \text{ if } \hat{y}_{ij} = 0, \forall (i,j) \notin \Omega$;
return $\hat{Y}$;

We illustrate the working of HMF using a toy example. Let us consider the follow-

ing partially observed ordinal rating matrix Y of size 5×7 with 5 levels.

Table 3.8: Ordinal rating matrix Y

3	0	0	5	2	0	0
5	4	0	1	5	3	4
1	0	4	0	3	1	0
5	4	0	0	0	0	1
0	3	2	0	5	2	0

For notational convenience, we use ij to denote (i, j). The set of observed entries $\Omega = \{11, 14, 15, 21, 22, 24, 25, 26, 27, 31, 33, 35, 36, 41, 42, 47, 52, 53, 55, 56\}$. For each of the 4 stages, the original 5-level rating matrix Y is converted to a bi-level (± 1) matrix. Taking $d = 2$, we employed bi-level MMMF to get factor matrix pairs (U^q, V^q) for each stage q independently. At stage 1, the bi-level matrix Y^1 obtained from Y, the learnt latent factor matrices U^1 and V^1 and the corresponding bi-level prediction $\hat{Y}^1$ is shown in Table 3.9.

Table 3.9: Runnning example.

1	0	0	1	1	0	0
1	1	0	-1	1	1	1
-1	0	1	0	1	-1	0
1	1	0	0	0	0	-1
0	1	1	0	1	1	0

Y^1

-0.48	-0.54
0.12	-1.09
0.98	-0.13
-0.77	-0.29
-0.01	-0.94

U^1

-0.70	-0.63
-0.36	-0.72
0.47	-0.53
-0.51	0.25
0.28	-0.80
-0.52	-0.66
0.58	-0.37

V^1

1	1	1	1	1	1	-1
1	1	1	-1	1	1	1
-1	-1	1	-1	1	-1	1
1	1	-1	1	1	1	-1
1	1	1	-1	1	1	1

$\hat{Y}^1$

At stage 1, the set of entries $\Omega^1 = \{17, 24, 31, 32, 34, 36, 43, 47, 54\}$ are predicted as -1. The set of entries $\Omega^1 \setminus \Omega$ i.e., $\{17, 32, 34, 43, 54\}$ and the set of entries where 1 are observed now contains rating 1 in the partially complete prediction matrix $\hat{Y}$ (Table 3.10).

Table 3.10: Partially complete matrix after stage 1

0	0	0	0	0	0	1
0	0	0	1	0	0	0
1	1	0	1	0	1	0
0	0	1	0	0	0	1
0	0	0	1	0	0	0

Similarly, at stage 2, the original matrix is mapped to bi-level matrix Y^2. The bi-level MMMF is employed on Y^2 to get the factor matrices U^2 and V^2. The factor matrices

and the corresponding bi-level prediction $\hat{Y}^2$ is shown in the Table 3.11.

Table 3.11: Runnning example.

1	0	0	1	-1	0	0
1	1	0	-1	1	1	1
-1	0	1	0	1	-1	0
1	1	0	0	0	0	-1
0	1	-1	0	1	-1	0

Y^2

-0.69	-0.47
-0.43	1.06
0.92	0.01
-0.79	0.05
0.24	0.86

U^2

-0.89	0.21
-0.43	0.64
0.42	-0.54
-0.21	-0.67
0.53	0.76
-0.76	0.01
0.35	0.56

V^2

1	-1	-1	1	-1	1	-1
1	1	-1	-1	1	1	1
-1	-1	1	-1	1	-1	1
1	1	-1	1	-1	1	-1
-1	1	-1	-1	1	-1	1

$\hat{Y}^2$

As discussed previously, at any stage q, the candidate set of entries for prediction will exclude the set of observed entries and the set of entries predicted before stage q. At stage 2, the set of entries $\Omega^2 = \{12, 13, 15, 17, 23, 24, 31, 32, 34, 36, 43, 45, 47, 51, 53, 54, 56\}$ are predicted as -1. The set of entries $\Omega^2 \setminus \{\Omega^1 \cup \Omega\}$ i.e., $\{12, 13, 23, 45, 51\}$ along with the entries where 2 are observed now contains rating 2 in the partially complete prediction matrix $\hat{Y}$ (Table 3.12).

Table 3.12: Partially complete matrix after stage 2

0	2	2	0	2	0	1
0	0	2	1	0	0	0
1	1	0	1	0	1	0
0	0	1	0	2	0	1
2	0	2	1	0	2	0

The bi-level mapping Y^3, the latent factor matrices U^3, V^3 and the corresponding bi-level prediction $\hat{Y}^3$ obtained at stage 3 is shown in able 3.13.

Table 3.13: Runnning example.

-1	0	0	1	-1	0	0
1	1	0	-1	1	-1	1
-1	0	1	0	-1	-1	0
1	1	0	0	0	0	-1
0	-1	-1	0	1	-1	0

Y^3

-0.42	0.70
0.26	-1.05
-0.80	0.44
0.84	0.08
-0.05	-0.81

U^3

0.72	-0.54
0.80	-0.12
-0.18	0.68
-0.26	0.62
0.26	-0.83
0.59	0.52
-0.39	-0.57

V^3

-1	-1	1	1	-1	1	-1
1	1	-1	-1	1	-1	1
-1	-1	1	1	-1	-1	1
1	1	-1	-1	1	1	-1
1	1	-1	-1	1	-1	1

$\hat{Y}^3$

It can be seen from Table 3.13 that the entries set $\Omega^3 = \{11, 12, 14, 15, 23, 25, 31, 32, 35, 36, 43, 44, 46, 53, 54, 56\}$ are predicted as -1. The set of entries $\Omega^3 \setminus \{\Omega^1 \cup \Omega^2 \cup \Omega\}$

i.e., $\{44\}$ and the set of entries where 3 are observed now is fixed to 3 in $\hat{Y}$ (Table 3.14).

Table 3.14: Partially complete matrix after stage 3

3	2	2	0	2	0	1
0	0	2	1	0	3	0
1	1	0	1	3	1	0
0	0	1	3	2	0	1
2	3	2	1	0	2	0

Similarly, at stage 4, the observed matrix Y is mapped to bi-level matrix Y^4. Table 3.15 shows the bi-level matrix Y^4, the factor matrix U^4, V^4 and the corresponding bi-level prediction matrix $\hat{Y}^4$ obtained at stage 4.

Table 3.15: Runnning example.

-1	0	0	1	-1	0	0
1	-1	0	-1	1	-1	-1
-1	0	-1	0	-1	-1	0
1	-1	0	0	0	0	-1
0	-1	-1	0	1	-1	0

Y^4

0.76	0.28
-0.76	-0.73
0.81	-0.58
-0.71	-0.43
-0.43	-0.85

U^4

-0.89	-0.15
0.54	0.55
-0.17	0.69
0.58	0.35
-0.86	-0.27
0.01	0.83
0.54	0.40

V^4

-1	1	1	1	-1	1	1
1	-1	-1	-1	1	-1	-1
-1	1	-1	1	-1	-1	1
1	-1	-1	-1	1	-1	-1
1	-1	-1	-1	1	-1	-1

$\hat{Y}^4$

The set of entries $\Omega^4 = \{11, 15, 22, 23, 24, 26, 27, 31, 33, 35, 36, 42, 43, 44, 46, 47, 52, 53, 54, 56, 57\}$ are predicted as $-$ at stage 4. The set of entries $\Omega^4 \setminus \{\Omega^1 \cup \Omega^2 \cup \Omega^3 \cup \Omega\}$ i.e., $\{46, 57\}$ and the set of entries where 4 are observed now contains rating 4 in the partially complete prediction matrix $\hat{Y}$ (Table 3.16).

Table 3.16: Partially complete matrix after stage 4

3	2	2	0	2	0	1
0	4	2	1	0	3	4
1	1	4	1	3	1	0
0	4	1	3	2	4	1
2	3	2	1	0	2	4

After completion of stage 4, the set of entries where 0 is present in $\hat{Y}^4$ are filled with rating 5. The resulting complete matrix is shown in Table 3.17.

Table 3.17: Complete matrix computed by HMF

3	2	2	5	2	5	1
5	4	2	1	5	3	4
1	1	4	1	3	1	5
5	4	1	3	2	4	1
2	3	2	1	5	2	4

3.5 Parallelization of HMF

The process described in Section 3.4 is sequential as the candidate entries for stage q are those which are completed till Stage $q - 1$. A minor modification makes the process suitable for distributed computing. Since every pair of (U, V) is used to predict only a non-overlapping subset of elements of Y, the computing need required at each stage q can be ported to a parallel or a distributed environment. In other words, the factorization for each Y^q can be accomplished in parallel on a multiprocessor system.
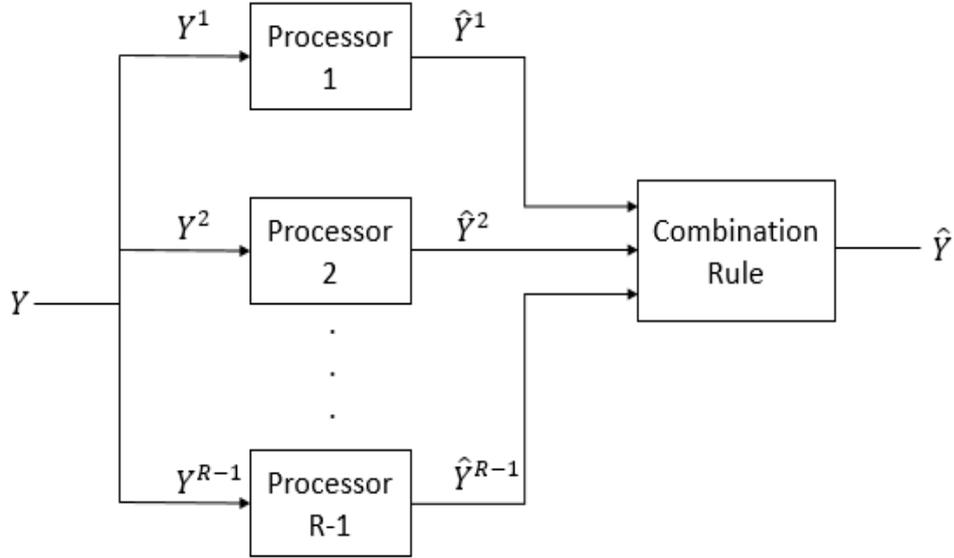
Figure 3.8: Block diagram of a parallel architecture for HMF

Figure 3.8 shows the block diagram of a parallel architecture for HMF. Each processor solves a $P_{\pm 1}(Y)$ problem defined at stage q. After that, the output of each processor i.e., $\hat{Y}^1, \hat{Y}^2, \ldots, \hat{Y}^{R-1}$ is combined to obtain the complete prediction matrix $\hat{Y}$. Algorithm 4 outlines the main flow of the parallel architecture for HMF.

Algorithm 4: PHMF $(Y, R, d, \theta, \lambda)$

---

input : Ordinal Rating Matrix: Y, Maximum Rating: R, Number of Latent
      Factors: d, Threshold: θ, Regularization Parameters: $\{\lambda_1, \ldots, \lambda_{R-1}\}$

output: $\hat{Y}$

Initialize: $\hat{Y} \leftarrow 0$;

for *each processor q in parallel* **do**
    $Y^q \leftarrow 0$;
    for $(i, j) \in \Omega$ **do**
        if $y_{ij} \leq q$ **then**
            $y_{ij}^q = -1$;
        else
            $y_{ij}^q = +1$;
        end
    end
    $\hat{Y}^q \leftarrow 0$;
    $(U^q, V^q) \leftarrow$ BMMMF (Y^q, d, λ_q);
    for *ever user-item pair* (i, j) **do**
        if $U_i^q V_j^{q^T} < \theta$ **then**
            $\hat{y}_{ij}^q = -1$;
        else
            $\hat{y}_{ij}^q = +1$;
        end
    end
end
for $q = 1$ *to* $R - 1$ **do**
    for *ever user-item pair* (i, j) **do**
        if $\big((i, j) \notin \Omega \wedge \hat{y}_{ij}^q = -1\big) \vee y_{ij} = q$ **then**
            $\hat{y}_{ij} \leftarrow q$
        end
    end
end
return $\hat{Y}$;

3.6 Experimental Analysis

In this section we analyze the performance of HMF by taking into account factors such as accuracy and efficiency. In the following sections, we describe the experimental setup including the data sets and relevant statistics, the experimental protocols, the competing algorithms, the evaluation metrics, the parameter setting and following this, we discuss the experimental results.

3.6.1 Data Sets

For a comprehensive performance evaluation, we conducted experiments on both *real* and *synthetic* data sets. We used three benchmark datasets for our experiments, namely *MovieLens 100K*, *MovieLens 1M* and *EachMovie*, which are the standard data sets used in the matrix factorization community. All of these datasets can be downloaded from *grouplens*[1]. The *MovieLens 100K* data set consists of $100,000$ ratings given by 943 users for 1682 movies. The *MovieLens 1M* dataset consists of $1,000,209$ ratings, given by $6,040$ users for $3,952$ movies, out of which $3,706$ are actually rated and every user has at least 20 ratings. There are five possible rating values, $\{1, 2 \ldots, 5\}$ in *MovieLens 100K* and *MovieLens 1M*. The *EachMovie* dataset consists of $2,811,983$ ratings, given by $72,916$ users for $1,628$ movies, out of which $1,623$ are actually rated and $36,656$ users has given at least 20 ratings. There are six possible rating values, $\{0, 0.2, \ldots, 1\}$ and we mapped them to $\{1, 2 \ldots, 6\}$. The detailed characteristics of these data sets are summarized in Table 3.18.

Table 3.18: Description of the experimental datasets

Data set	#Users	#Items	#Ratings	Sparsity	Rating-scale	Filtering
MovieLens 100K	943	1682	100,000	93.7%	5	20
MovieLens 1M	6040	3900	1,000,209	95.7%	5	20
EachMovie	72,916	1628	2,811,983	97.6%	6	20

Synthetic Data Generation: We generated the synthetic dataset of size 1000×1000 with ratings in the range 1-5. Synthetic data is generated by a novel method. The motivation for using *synthetic* dataset in the experiments is to see whether a method can successfully retrieve the factor matrices when the input matrix is known to be the exact product of the latent factor matrices. Since the rating matrix is discrete and cannot be the exact product of two real-valued matrices, we make use of a novel iterative technique to generate the rating matrices. Let Y_0 be a random rating matrix of size $N \times M$ and $d \leq min(n, m)$. We start with a random $N \times d$ factor matrix U. Then the latent factor matrix V is obtained as $V = ((U^T U)^{-1} U^T Y_0)^T$. We then compute $[UV^T]$ as Y_1, where $[\cdot]$ denotes rounding to the nearest integer[2] . Y_1, so computed, becomes new Y_0 and U

---

[1]http://grouplens.org/datasets/

[2]In case if nearest integer is exceeding (or below) the range of rating we set it to maximum (minimum) rating.

is computed as $U = Y_0(VV^T)^{-1}V$. This process is repeated iteratively, by alternatively updating U and V, until the matrices stbilize. The current Y_0 becomes the *synthetic* rating matrix.

3.6.2 Experimental Protocols

Our experimental set up is based on two different and popular experimental protocols that have been proposed in the literature for evaluating the empirical performance of collaborative filtering methods i.e., weak and strong generalization (85).

- Weak generalization: Testing based on weak generalization protocol is based on the notion that the *test set* is formed by randomly holding out one rating from each user's rating set and the rest of the known ratings are considered as part of the *training set*. A prediction model is trained using the data in the training set, and its performance is evaluated over the test set. Therefore weak generalization is a one stage process wherein one can measure the ability of a model to generalize to other items rated by the *same users* used for training the model.

- Strong generalization: It is a two stage process in which, initially, a subset of users is randomly selected and completely removed from the training set so as to form the test set. We can call this test set **G**. The initial prediction model (**M**) is trained (learned) with all available ratings from the bulk of the users in the training set (all users not in the test set **G**) . In the second stage (testing phase) one rating is randomly selected from each user in the set **G** to form the *held out* set. The remaining ratings for each user in the set **G** can be used to tweak the prediction model (**M**) during strong testing. The model is evaluated by predicting the held out ratings. The main idea behind strong generalization is to build a prediction model for a large set of initial users that can be generalized later for a small set of novel users (**G**).

3.6.3 Comparing Algorithms

We consider the following nine well-known state-of-the-art algorithms for comparison:

- **URP** (85): The user rating profile (URP) model is the generative version of the vector aspect model for rating profiles. The URP model sample the latent features for each user from a Dirichlet prior and then perform variational inference on the user profile. Finally, compute the distribution over rating values for a particular unrated item given a user profile.

- **Attitude** (85): The attitude model representsthe latent space description of a user as a vector of attitude expression levels. Given an attitude vector of user, the probability of rating value for each item is a product of attitude expression levels and preference parameters.

- **MMMF** (97): The *hinge loss* function defined for bi-level is extended to multi-level for collaborative filtering with ordinal rating matrix when user-preferences are not in the form of like/ dislike but values in a discrete range.

- **E-MMMF** (23): To overcome the problem of local minima and impact of outliers (e.g. abnormal or even malicious raters) and other noise in MMMF, E-MMMF investigates diffrent ensemble methods such as *multiple random weight seeds* and *baggings* with MMMF.

- **PMF** (86): Salakhutdinov et al. (86) proposed a probabilistic framework for matrix factorization where the factor variables are assumed to be marginally independent while rating variables are assumed to be conditionally independent given the factor variables.

- **BPMF** (100): Given the ratings, inferring the posterior distribution over the factors is intractable in PMF. BPMF presented a fully Bayesian treatment of Probabilistic Matrix Factorization by placing hyperpriors over the hyperparameters. BPMF uses Markov chain Monte Carlo (MCMC) method for approximate inference in this model.

- **GP-LVM** (67): Lawrence et al. (67), developed a non-linear extension to PMF by generating latent components via Gaussian process latent variable models.

- **iPMMMF and iBPMMMF** (132): In (132), a noparametric Bayesian-style MMMF was proposed that utilizes nonparametric techniques to resolve the unknown number of latent factors in MMMF model.

- **Gibbs MMMF and iPMMMF** (133): A probabilistic interpretation of MMMF model through data augmentation is presented in (133).

3.6.4 Evaluation Metrics

In order to evaluate the performance of the proposed method, we use two popular evaluation metrices: *Normalized Mean Absolute Error (NMAE)* (85) and *Frobenius-norm RelativeError (FRE)* (77). NMAE is defined as Mean Absolute Error (MAE) divided by 1.6 in the case of MovieLens data set and for EachMovie data set it is MAE divide by 1.944 (85). Given Y a $N \times M$ partially observed user-item rating matrix, Ω the observed entries set and $\hat{Y}$ be a recovered matrix, the MAE and FRE are defined as follows.

$$MAE = \frac{\sum_{(i,j) \in \Omega} |y_{ij} - \hat{y}_{ij}|}{|\Omega|} \qquad FRE = \sqrt{\frac{\sum_{(i,j) \in \Omega} (y_{ij} - \hat{y}_{ij})^2}{\sum_{(i,j) \in \Omega} y_{ij}^2}}$$

3.6.5 Parameter Setting

We first demonstrate the effect of regularization parameter λ. Determining λ value is one of the critical aspect for learning latent factors optimized over a loss function. To select the best regularization value for each dataset, we follow the same approach as defined in (23). We held out one rating from every user to form the validation set which
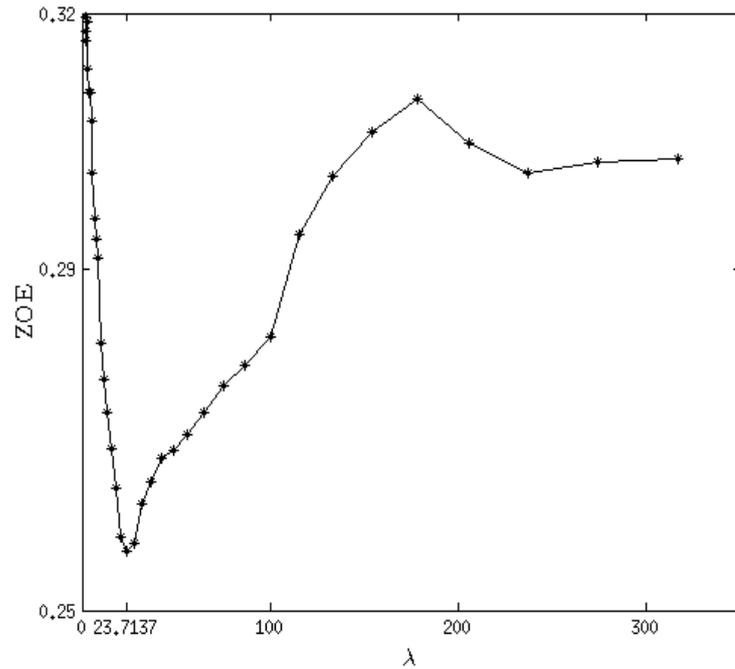
Figure 3.9: Validation score for different values of λ on *MovieLens* data set
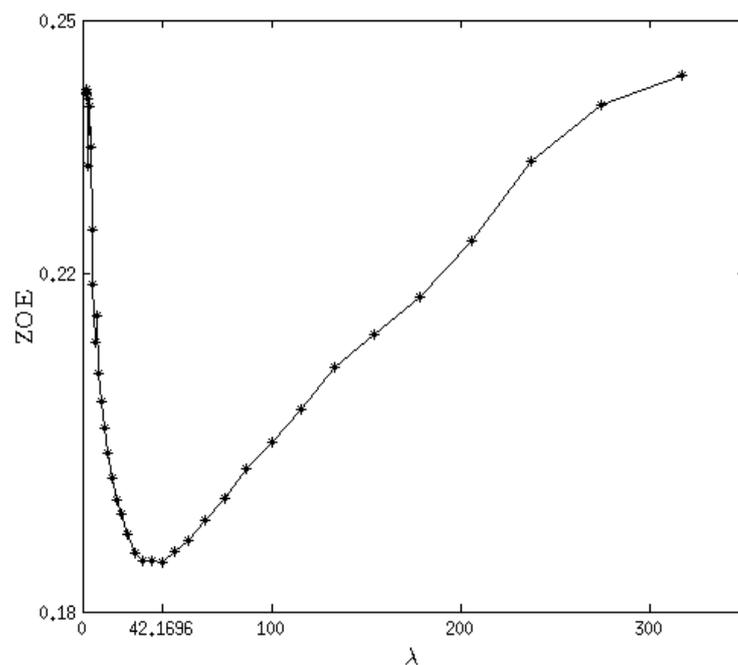
Figure 3.10: Validation score for different values of λ on *EachMovie* data set

is later used to evaluate the model performance. We repeat the selection process (train/ validation set) 3 times for every candidate λ and calculate Zero-One Error (ZOE) on validation set. Finally, we select λ corresponding to the smallest ZOE. The candidate values of λ are $\{10^{\frac{i}{16}}\}, \forall i \in \{4, 5, \ldots, 40\}$. We repeat the same procedure for every level and select the best λ for the experiments reported in Table 3.19. Figures 3.9 and 3.10 depict the variation of ZOE for different values of λ for two data sets for Stage 3. We carry out this exercise for each stage.

3.6.6 Results and Discussion

Comparative Analysis: Table 3.19 gives the comparative analysis of the proposed HMF against state-of-art algorithms on two different data sets with two different experimental setups. We use the same d value ($d = 100$) as reported in all the comparative methods to attain fair evaluation. We observe that our algorithm exhibits better accuracy in case of *EachMovie* data set than any of the algorithms in both strong and weak generalization. In the case of *MovieLens 1M* data set, our method outperforms other methods in weak generalization and is better than all but GP-LVM for strong generalization. The results reported here are average values of three runs.

Table 3.19: Average and standard deviation of NMAE of different models

Algorithms	MovieLens		EachMovie	
	weak	strong	weak	strong
URP	.4341 ± .0023	.4444 ± .0032	.4422 ± .0008	.4557 ± .0008
Attitude	.4320 ±.0055	.4375 ± .0028	.4520 ± .0016	.4550 ± .0023
MMMF	.4156 ± .0037	.4203 ± .0138	.4397 ± .0006	.4341 ± .0025
E-MMMF	.4029 ± .0027	.4071 ± .0093	.4287 ± .0020	.4295 ± .0030
PMF	.4332 ± .0033	.4413 ± .0074	.4466 ± .0016	.4579 ± .0016
BPMF	.4235 ± .0023	.4450 ± .0085	.4352 ± .0014	.4445 ± .0005
GP-LVM	.4026 ± .0020	**.3994 ± .0145**	.4179 ± .0018	.4134 ± .0049
iPMMMF & iBPMMMF	.4031 ± .0030	.4089 ± .0146	.4211 ± .0019	.4224 ± .0051
Gibbs MMMF & iPMMMF	.4037 ± .0005	.4040 ± .0055	.4134 ± .0017	.4142 ± .0059
HMF	**.4019 ± .0044**	.4032 ± .0022	**.4118 ± .0019**	**.4095 ± .0044**

As stated earlier, the main motivation of our algorithm is to examine *combinational* approach as an alternative to otherwise well-known *embedded* principle of MMMF. There is substantial improvement of performance (in terms of NMAE) of HMF from MMMF. This corroborates our claim that the proposed framework which is based on

a *combinational* approach has advantages over factorizations based on *embedded* approach (like MMMF).

Number of Latent Factors: Selecting an optimal (minimum) number of latent factors (d) with less compromise on accuracy is important for any matrix factorization technique. In the second set of experiments we analyse the impact of d on the proposed method versus MMMF. We use *MovieLens 100K* dataset for this study. We randomly select 80% data for training and rest 20% for testing. For every *d*, we tune the regularization parameter in both MMMF and HMF so that the NMAE on training set is in the range $[0.06, 0.08]$. We calculate the NMAE on the test set. The average NMAE of three runs are reported in Figure 3.11, where the abscissa depicts d in decreasing order. It can
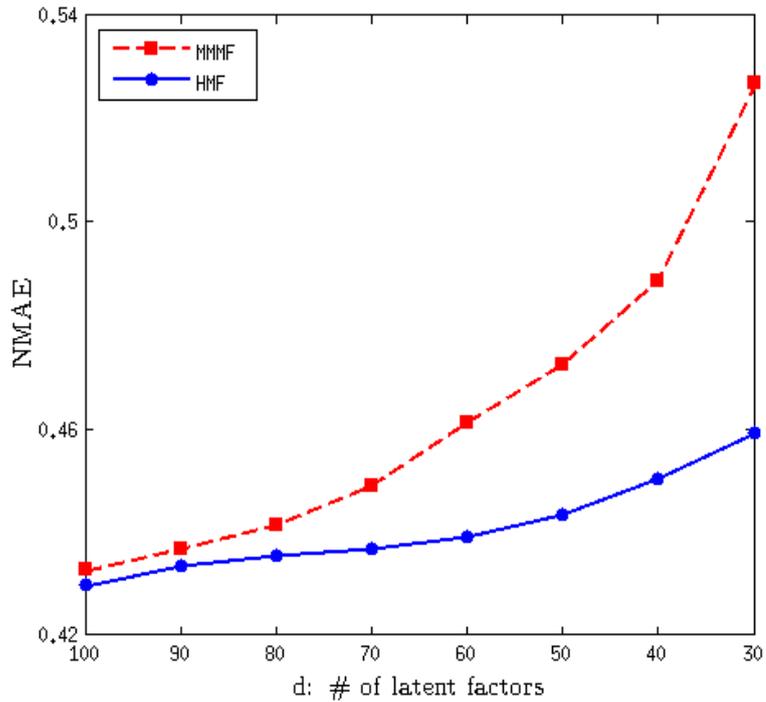
Figure 3.11: Testing error on changing d value

be seen from Figure 3.11 that when d decreases the performance of MMMF drastically deteriorates whereas HMF maintains a balanced performance. This observation indicates that the proposed method can be used for lower ranks of the latent factor matrices without compromising the accuracy of prediction.

Generalization Analysis: Traditionally, the concept of maximum margin was introduced with an objective of achieving good generalization error in contrast to empirical error. It is well-known that both cannot be achieved together and hence there is a trade-off between test-error and training-error. We analyse the interaction between test error

and the training error for HMF and MMMF on *MovieLens 100K* data set. We fix the value of d to 100 and plot test-error against training-error (Figure 3.12). It is observed that the trade-off of MMMF is much higher than that of the proposed method. When the training-error is reduced, test-error for MMMF becomes higher than that of our method. Similarly, for low test-error, training error of MMMF is higher than that of our method.
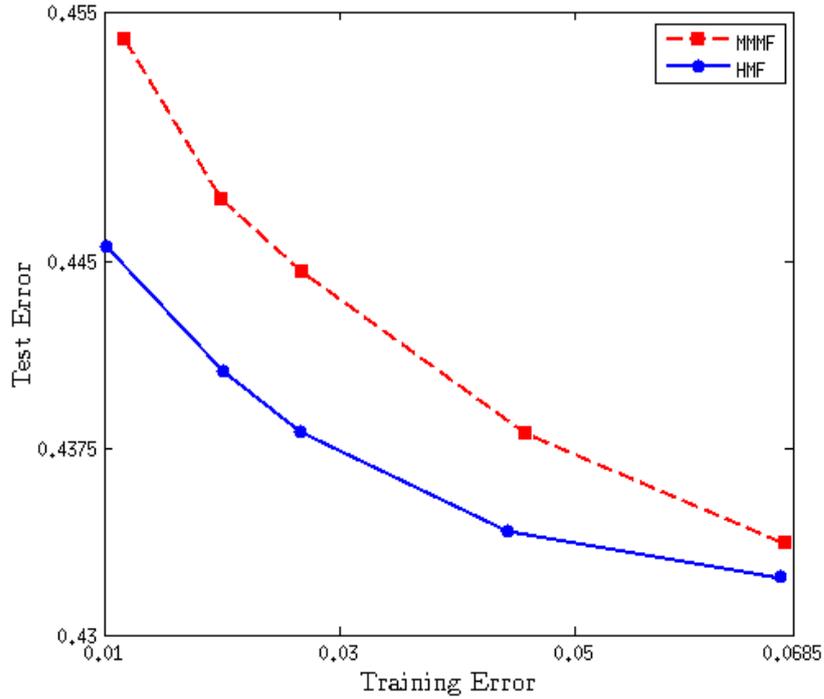
Figure 3.12: Training error vs Test error

Computation Time: Although HMF uses several stages of matrix factorization, it requires less computational time than MMMF. The optimization problem of MMMF has $Nd + Md + N(R - 1)$ variables whereas each stage of HMF is an optimization problem of $Nd + Md$ variables. Any Gradient Descent based algorithm requires updating these variables iteratively and hence the number of variables has a major influence on the computational time. We carry out experiments to compare the computational efficiency of HMF with MMMF for different values of *d* with fixed λ. For MMMF we use the same λ as reported in (23). Aggregated time of HMF for all $R - 1$ stages requires substantially less computational time than that of MMMF for *MovieLens 1M* (Figure 3.13) and *EachMovie* (Figure 3.14) datasets. It can be concluded that the proposed framework, HMF is both accurate and computationally efficient.
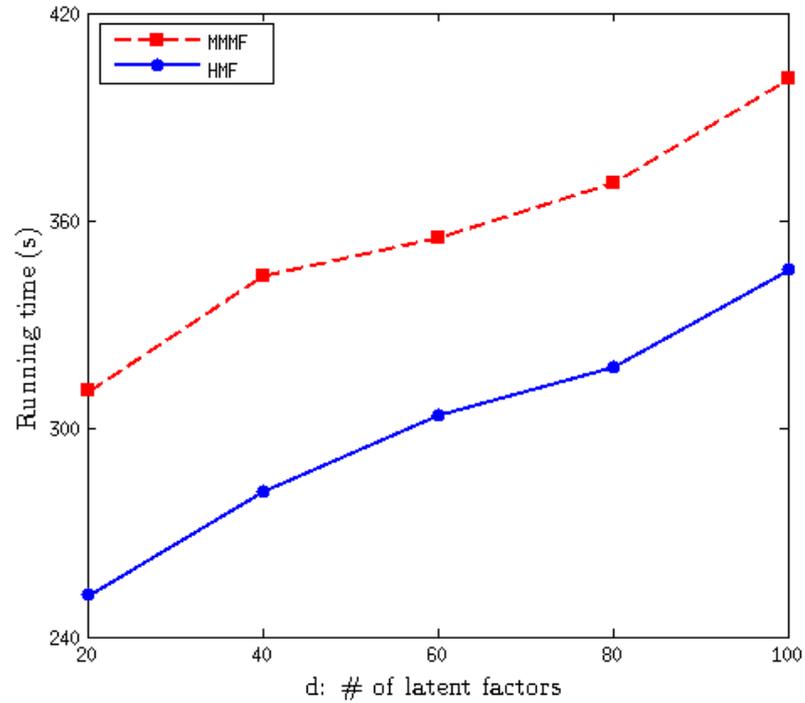
Figure 3.13: Running time on MovieLens dataset
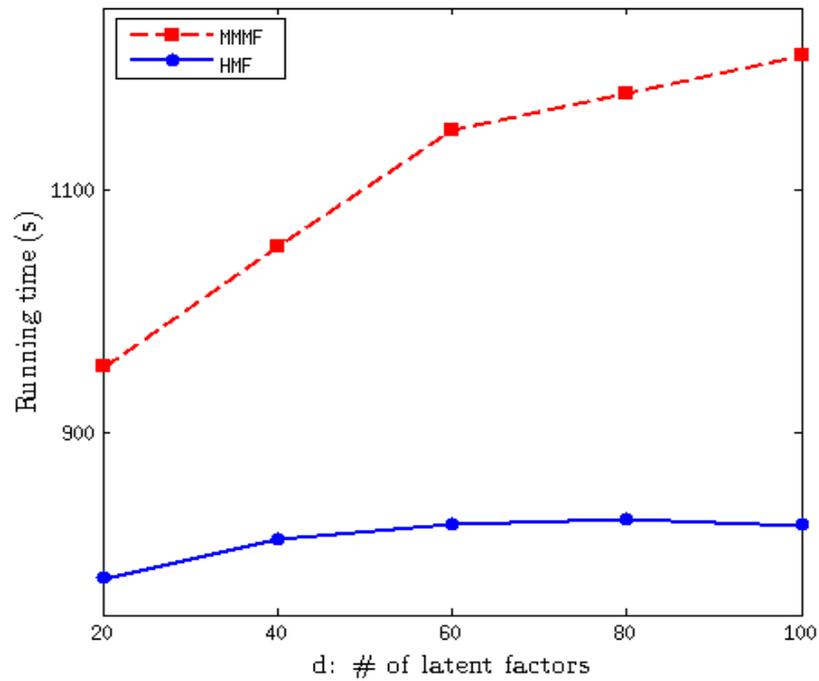
Figure 3.14: Running time on EachMovie dataset

Matrix Recovery: In this experiment we analyzed the effect of sample percentage and the number of latent factors that are necessary for acceptable recovery of the original matrix Y. A *synthetic* matrix Y of size 500×500 is generated using the procedure given in Section 3.6.1. We kept the observed entries percentage to 100 and the number of ordinal ratings to five $(1-5)$. Given Y, we train HMF and MMMF models with different
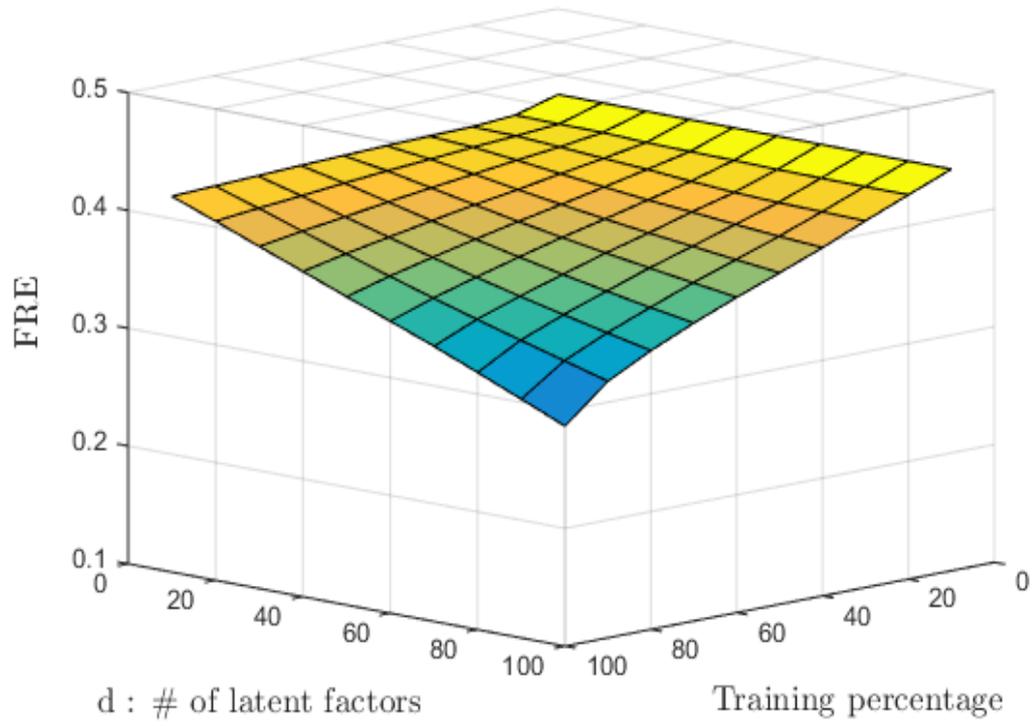
(a) MMMF
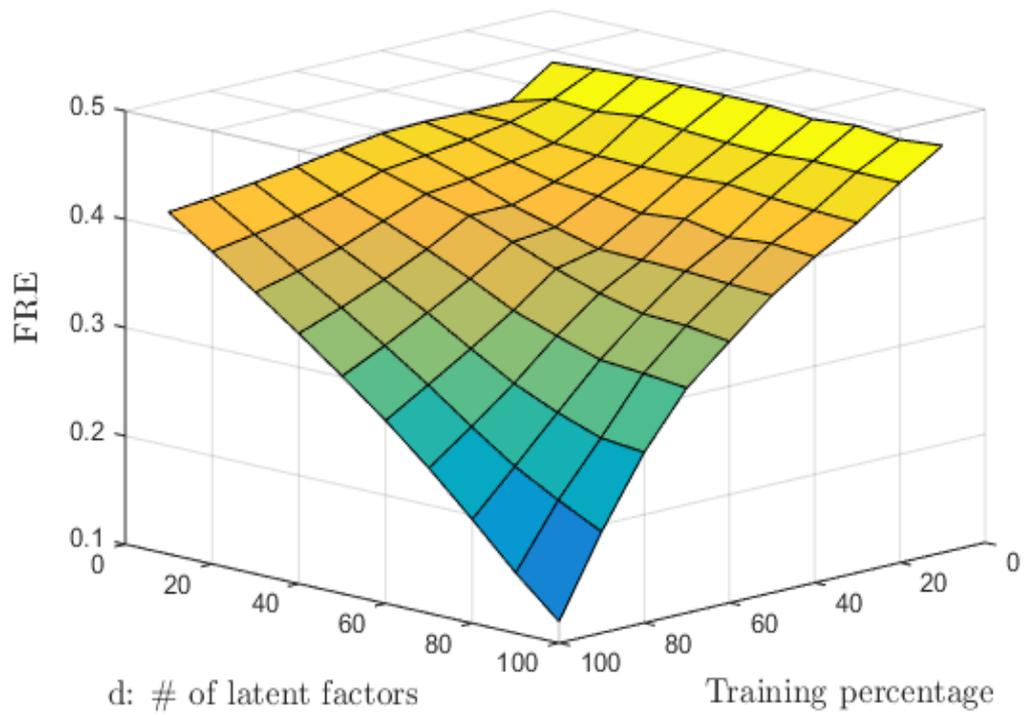
(b) HMF

Figure 3.15: Recovery of original rating matrix by MMMF and HMF

53

percentage of sampled entries (Ω) and for different values of the latent dimension d. For each (Ω, d)-pair, the regularization parameter λ in both HMF and MMMF are tuned from the candidate set $\{10^{\frac{i}{16}}\}, \forall i \in \{4, 5, \ldots, 40\}$. We repeat the selection process (train/ test set) three times for every (Ω, d)-pair and calculate the average FRE. The result is reported in Figure 3.15. It can be seen from Figure 3.15 that there is an increase in the performance of HMF as the observed entries percentage is increasing and for observed entries $\geq 40\%$, HMF is performing better than MMMF.

3.7 Conclusions

There have been large number of publications on MMMF but these profess the principle of *embedded technique*. The present chapter for the first time investigates matrix factorization of ordinal matrix in a *combinational technique* that uses multiple bi-level matrix factorization. We show several advantages of this approach. Our experimental results show that NMAE for our framework is better than any of these approaches including probabilistic approaches, except in the case of GP-LVM (MovieLens 1M) for strong generalization protocol. The proposed method exhibits better trade-off of generalization vs. empirical errors and it yields latent factors of lower rank. In the era of big data, when the data is voluminous the proposed method can easily be ported to a parallel or a distributed environment. In a multiprocessor system, the factorization for each q can be accomplished in parallel.

CHAPTER 4

Proximal Maximum Margin Matrix Factorization for Collaborative Filtering

In the previous chapter, we discussed the underlying principle of maximum margin matrix factorization (MMMF) and the motivation for finding an alternative solution. This led us to a new formulation of matrix factorization, namely, hierarchical matrix factorization which was discussed in the subsequent parts of Chapter 3. In the present chapter, we propose another alternative for matrix factorisation techniques based on maximum margin and propose a novel method called proximal maximum margin matrix factorization (PMMMF). The background information for this chapter is the same as discussed in Section 3.3.

4.1   Introduction

In MMMF (97), as shown in Section 3.2, the smooth hinge loss function is defined to ensure that the embedding of points and hyperplanes maintain specific margin to minimize generalization error. In the context of support vector machines (SVM) there have been proposals of alternative formulations different from margin maximization. There are proposals where the hyperplanes are non-parallel for different classes as proposed in twin SVM (50). Similarly, proximal SVM (31) attempts to classify based on

proximity of the data points to the decision hyperplane. Taking the que from such alternative formulation of SVMs this chapter explores the possibility of proximal as an alternative criterion instead of margin maximization criterion. In this sense, the idea is to look for embedding of row of U and row of V such that if an item j is rated as r by user i then the objective of the embedding is to ensure that the point V_j is closest to the rth parallel hyperplane of U_i. This criteria necessitates redefining the loss function. A new loss function is derived in this chapter and it is observed in this process that the threshold values can be computed in closed form avoiding any optimization process. The proposed loss function ensures that the embedded point is in the proximity of the desired hyperplanes and maintain a distance from other hyperplanes. In order to have the parallel hyperplanes corresponding to different ratings uniformally spaced, we have used a hinge loss function. Thus, our loss function is a combination of proximal loss and hinge loss function. There has not been any attempt to find alternative matrix factorization based on maximum margin for ordinal ratings. The present formulation is one of the first alternative formulation to MMMF.

The rest of the chapter is organized as follows. We introduce our proposed method, termed as Proximal Maximum Margin Matrix Factorization (PMMMF), in Section 4.2. Experimental analysis of the proposed method is reported in Section 4.3. We conclude the chapter with a discussion on future directions in Section 4.4.

4.2   PMMMF- The Proposed Method

In this section, a novel method of matrix factorization for discrete valued ratings is proposed. Given a partially observed rating matrix Y with R ratings and with Ω as the observed set, the aim is to determine two factor matrices U and V and a threshold set θ such that the predicted value $x_{ij} = U_i V_j^T$ is related to rating y_{ij} by proximity of x_{ij} to the corresponding threshold $\theta_{y_{ij}}$. We use, similar to MMMF, R number of thresholds for each user. But, unlike MMMF, the thresholds are computed in closed form and not considered as variables during the optimization step. Thus, the objective is to minimize

$$J = \sum_{r=1}^{R} \sum_{(i,j)\in\Omega\wedge(y_{ij}=r)} (U_i V_j^T - \theta_{i,r})^2 + \frac{\lambda}{2}(\|U\|_F^2 + \|V\|_F^2). \qquad (4.1)$$

At the minimizing point we have $\frac{\partial J}{\partial \theta_{i,r}} = 0$ which implies that

$$\theta_{i,r}^* = \frac{\sum_{(i,j)\in\Omega\wedge(y_{ij}=r)} U_i V_j^T}{|\Omega(i,r)|}$$

where $\Omega(i,r)$ is the index set of items rated as r by user i.

We shall use the above expression to determine the threshold values for fixed U and V. When $\Omega(i,r)$ is empty, we take the corresponding threshold $\theta_{i,r}^*$ as undefined. This assumption is reasonable because, in the absence of any training example of rank r by i, the corresponding threshold cannot be learned. Without loss of generality, we assume that $\Omega(i,r)$ is never empty and there is at least one item of rank r by user i. Our optimization iteration alternates between two steps . In one step, the factor matrices U and V are updated and then using the updated U and V the optimal value of θ^* is computed. We use the newly computed θ^* in the next iteration.

There can be several possible Us and Vs minimizing the objective function in Eq (4.1). In order to ensure that the hyperplanes are widely placed, a hinge loss component is added and the new objective is to minimize

$$J = \sum_{r=1}^{R} \sum_{(i,j)\in\Omega\wedge(y_{ij}=r)} (D_{ijr})^2 + \sum_{r=1}^{R} \sum_{(i,j)\in\Omega\wedge(y_{ij}\neq r)} h(T_{ij}^r(D_{ijr})) + \frac{\lambda}{2}(\|U\|_F^2 + \|V\|_F^2)$$

(4.2)

where $h(z)$ is the same as defined in Eq (3.6) , $D_{ijr} = (U_i V_j^T - \theta_{i,r}^*)$ and T_{ij}^r is defined as

$$T_{ij}^r = \begin{cases} +1 & \text{if } r < Y_{ij}; \\ -1 & \text{if } r > Y_{ij}. \end{cases}$$

(4.3)

The gradients are determined as follows.

$$\frac{\partial J}{\partial U_{ip}} = \lambda U_{ip} + \sum_{r=1}^{R} \sum_{(i,j)\in\Omega\wedge(y_{ij}=r)} 2(D_{ijr})(V_{jp} - \bar{V}_{ipr})$$

$$+ \sum_{r=1}^{R} \sum_{(i,j)\in\Omega\wedge(y_{ij}\neq r)} T_{ij}^r h'(T_{ij}^r(D_{ijr}))(V_{jp} - \bar{V}_{ipr})$$

(4.4)

$$\frac{\partial J}{\partial V_{jq}} = \lambda V_{jq} + \sum_{r=1}^{R} \sum_{(i,j)\in\Omega\wedge(y_{ij}=r)} 2(D_{ijr})U_{iq} + \sum_{r=1}^{R} \sum_{(i,j)\in\Omega\wedge(y_{ij}\neq r)} T_{ij}^{r} h'(T_{ij}^{r} D_{ijr}))U_{iq}$$
$$- \sum_{r=1}^{R} \sum_{(i,j)\in\Omega\wedge(y_{ij}=r)} \frac{\sum_{(i,t)\in\Omega\wedge y_{it}\neq r} T_{it}^{r} h'(T_{it}^{r}(D_{itr}))}{|\Omega(i,r)|} U_{iq} \quad (4.5)$$

where $\bar{V}_{ipr}$ is defined as

$$\bar{V}_{ipr} = \frac{\sum_{(i,t)\in\Omega(i,r)} V_{tp}}{|\Omega(i,r)|}.$$

Finally, the latent factor matrices U and V are iteratively updated using the following rule.

$$U_{ip}^{t+1} = U_{ip}^{t} - c\frac{\partial J}{\partial U_{ip}^{t}} \quad (4.6)$$

$$V_{jq}^{t+1} = V_{jq}^{t} - c\frac{\partial J}{\partial V_{jq}^{t}} \quad (4.7)$$

When predicting ordinal ratings, the introduced thresholds θ are very important since they underpin the large-margin principle of maximum-margin matrix factorization models. The matrix completion process is accomplished from the factor matrices U and V by the following rule.

$$\hat{Y}_{ij} = \begin{cases} r & \text{if } (\theta_{i,r}^{*} + \frac{n_{i,r}}{n_{i,r}+n_{i,r+1}}|\theta_{i,r+1}^{*} - \theta_{i,r}^{*}| < U_i V_j^T)\wedge \\ & (0 \leq r \leq R) \wedge (i,j) \notin \Omega; \\ y_{ij} & \text{if } (i,j) \in \Omega, \end{cases}$$

where $n_{i,r}$ is the number of items rated as r by the ith user. For simplicity of notation we assume $\theta_{i,0}^{*} = -\infty$, $\theta_{i,R+1}^{*} = +\infty$, $n_{i,0} = 0$ and $n_{i,R+1} = 0$.

The objective of PMMMF is to learn the embedding of items as points, users as hyperplanes and ratings as thresholds such that the embedding of points rated as r by user i is in the proximity of the decision hyperplane defined by $(U_i, \theta_{i,r})$. While embedding, PMMMF ensure that the hyperplanes are widely placed so that a large margin is created between two classes. In other words, the hyperplane defined by $(U_i, \theta_{i,r})$ is no longer a bounding plane as in MMMF, but can be seen as *proximal* plane, around which the items rated similar are clustered and placed as far as possible from other hyperplanes. We illustrate the concept of PMMMF by taking a synthetic data of size 5×1000 with
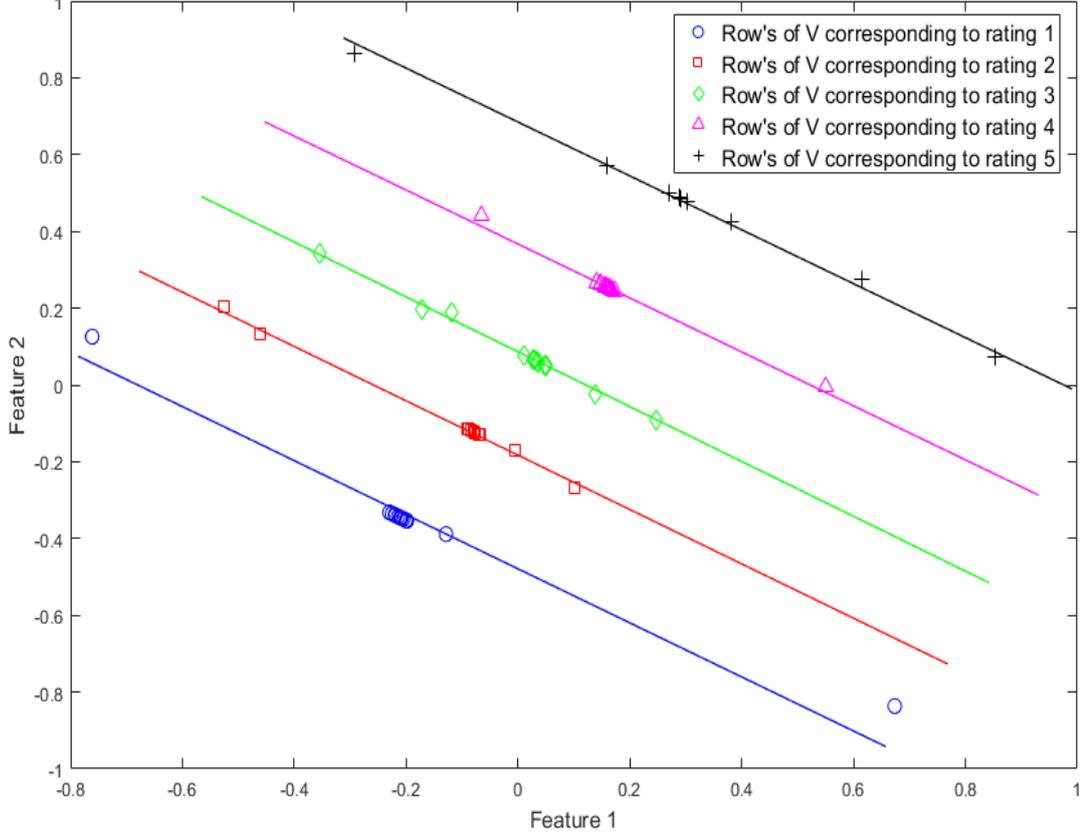
Figure 4.1: Classification by PMMMF for the ith user

10% of observed entries. The number of ordinal rating is 5 and $d = 2$. Figure 4.1 gives the decision hyperplane for a user and embedding of points corresponding to items. Since $R = 5$, there are 5 hyperplanes corresponding to 5 ratings. From Figure 4.1, it is clear that the embedding is learnt perfectly and the points rated similar fall in the *proximity* of the respective decsion hyperplane.

Complexity Analysis: We analyze the computational complexity of the proposed method. In every iteration of gradient descent, PMMMF requires computation of θ^*, $\bar{V}$ and UV^T with computation cost MN, MN and MNd, respectively. The computation cost required for calculation of gradients $\frac{\partial J}{\partial U}$ and $\frac{\partial J}{\partial V}$ are $2MNdR$ and $3MNdR$, respectively. Hence, the overall computation required in every gradient iteration is $(5MNdR + MNd + 2MN)$, that is, $\mathcal{O}(MNdR)$.

The optimization problem of PMMMF requires updation of $Nd + Md$ variables unlike MMMF where the optimization problem requires updation of $Nd+Md+N(R-1)$ variables in every iteration of gradient descent method. But on the other hand the updation step in PMMMF takes more time because of repeated computation of the *mean* for obtaining θ^*. Though both the methods are of $\mathcal{O}(MNdR)$. The dominating

component in PMMMF and MMMF are $5MNdR$ and $2MNdR$, respectively. In that sense, the computation cost of PMMMF is higher than that of MMMF. Reducing the computation cost of PMMMF is part of our future work.

4.3 Experiments

In the following sections, we describe the experimental setup including the data sets, the evaluation metrics, the competing algorithms and following this, we discuss the experimental results.

4.3.1 Data Sets

We carried out experiments on *real* and *synthetic* data sets. We use *MovieLens* $100K$ as the benchmark data set which is standard in the matrix factorization community. The detailed characteristics of *MovieLens-100K* is given in Subsection 3.6.1. The *MovieLens-100K* data set consists of $100,000$ ratings $(1-5)$ given by 943 users for 1682 movies. Out of 943 users, 693 users have used all possible ratings to express their preferences. We generated the synthetic dataset of size 1000×1000 with ratings in the range $1-5$. Detailed procedure related to generating the synthetic data is given in Subsection 3.6.1.

4.3.2 Evaluation Metrics

In order to evaluate the performance of the proposed method, we used two popular evaluation metrices: *Mean Absolute Error (MAE)* (85) and *Root Mean Square Error (RMSE)* (77). Given Y a $N \times M$ partially observed user-item rating matrix, Ω be the observed entries set and $\hat{Y}$ be a recovered matrix, the MAE and RMSE are defined as follows.

$$MAE = \frac{\sum_{(i,j) \in \Omega} |y_{ij} - \hat{y}_{ij}|}{|\Omega|} \qquad RMSE = \sqrt{\frac{\sum_{(i,j) \in \Omega} (y_{ij} - \hat{y}_{ij})^2}{|\Omega|}}$$

4.3.3 Comparing Algorithms

We consider the following two well-known state-of-the-art algorithms for comparison:

- **ALS** (65): In regularized least squares based matrix factorization (RMF), the goal is to minimize the squared sum distance between the observed entry and the corresponding prediction while overfitting is avoided through a regularized model. One of the several approaches to solve the RMF problem is the alternating minimization algorithms (ALS) where the latent factor matrices are updated in alternate steps.

- **MMMF** (97): The *hinge loss* function defined for bi-level is extended to multi-level for collaborative filtering with ordinal rating matrix when user-preferences are not in the form of like/ dislike but are values in a discrete range.

All methods are implemented in MATLAB with single computational thread on 4-core 3.40GHz Intel i7 CPU with 4GB RAM. As reported in (97), factor numbers higher than 50 yield similar performances. Hence, they choose $d = 100$ as a compromise between model capacity and computational complexity. Therefore, we also set the latent factor d to be 100 for all methods to attain fair evaluation. In our experiments, we randomly selected 80% of the observed ratings for training and used the remaining 20% as the test set. We report the average of the three prediction accuracies.

4.3.4 Experimental Results

Generalization Analysis: The concept of maximum margin was introduced with an objective of achieving good generalization error in contrast to empirical error. In our first experiment, we analysed the trade-off between generalization-error and empirical-error on the *synthetic* dataset by varying the number of observed entries. We vary the regularization parameter λ in the range $\{10^{\frac{i}{16}}\}, \forall i \in \{1, 5, \ldots, 25\}$, for all the methods so that the MAE on training set decreases. Thereafter, we plot the generalization-error against the empirical-error. In all the graphs of Figure 4.2, the curve corresponding to PMMMF is lower than that of the other methods and the sufficient gap indicates that the rate of increase in empirical-error vs generalization-error is slower. In other words, our proposed approach, PMMMF, exhibits better trade-off than that of MMMF and ALS.
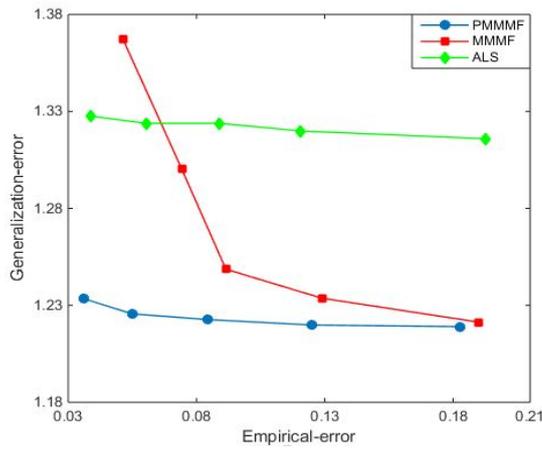
(a) Observed entries 5%   (b) Observed entries 10%
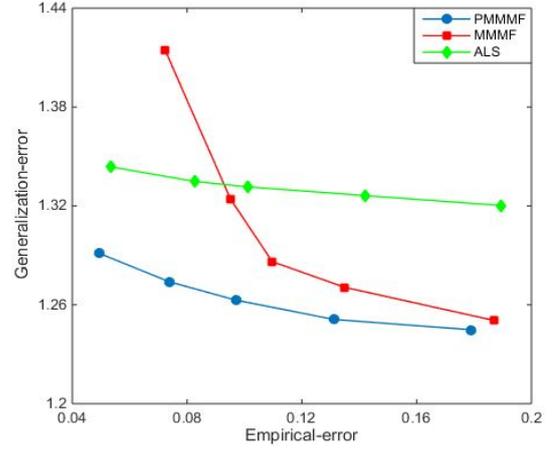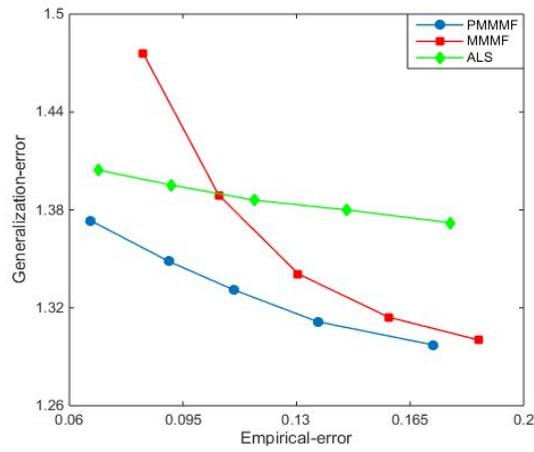
(c) Observed entries 15%   (d) Observed entries 20%
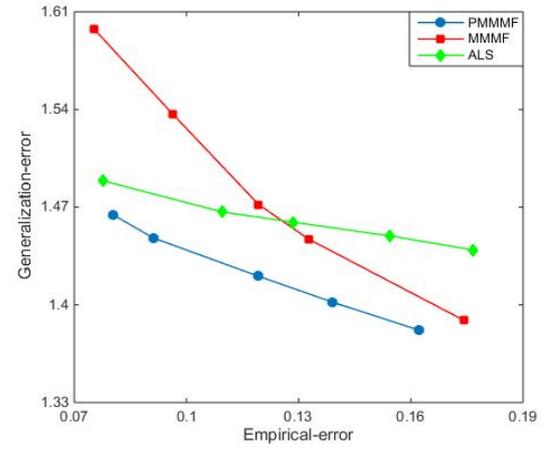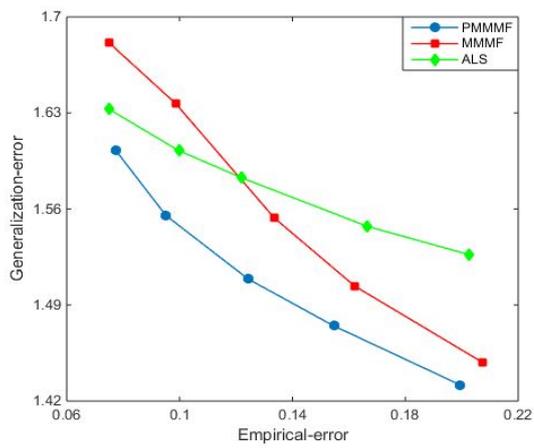
(e) Observed entries 25%   (f) Observed entries 30%
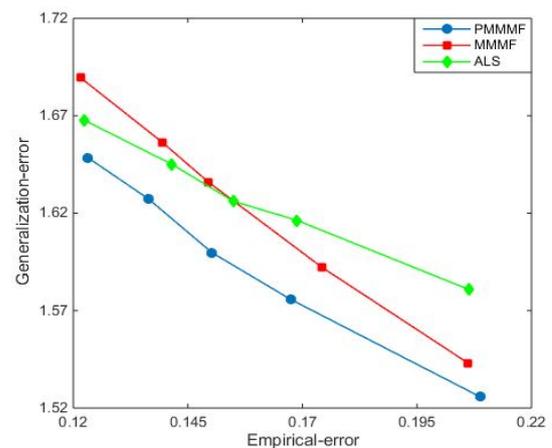
Figure 4.2: Figure (a), (b), (c), (d), (e) and (f) shows the trade-off between Generalization-error Vs. Empirical-error on synthetic dataset of size 1000×1000 with 5%, 10%, 15%, 20%, 25% and 30% observed entries respectively.

Comparative Analysis: In the second set of experiments, we compare the accuracy of the proposed PMMMF against MMMF and ALS on *MovieLens 100K* datasets. We tune the regularization parameter λ in all the methods so that MAE on the training set is in the range $[0.07, 0.09]$. From the results reported in Table 4.1, it can be seen that our proposed method, PMMMF, exhibits better accuracy than that of MMMF and ALS.

Table 4.1: Average and standard deviation of MAE and RMSE of different models

Algorithms	MAE	RMSE
MMMF	0.7355 ± 0.0070	1.0426 ± 0.0093
ALS	0.7899 ± 0.0039	1.0838 ± 0.0055
PMMMF	0.7138 ± 0.0045	1.0178 ± 0.0041

4.4 Conclusions and Discussion

This chapter introduced a novel concept of matrix factorization for multi-level ordinal rating matrix. Other than MMMF (97), there has not been any attempt for factorizing such matrices. In MMMF, the factorization is achieved by the latent factors which separate different ratings in terms of margins. Recently, proximal SVMs are proposed as an alternative to margin-based classifiers. Proximal SVMs are shown to be superior to traditional SVMs on many counts. Taking the cue from here, we make an attempt in this chapter to introduce *proximity* criterion in place of margin maximization criterion in the context of matrix factorization. The process is non-trivial and novel. Such improvisation yields superior performance. We hope that this will open up new vistas of research in collaborative filtering and matrix completion.

Like MMMF, PMMMF also assumes that the entries of the rating matrix are ordered. One can think of a more general problem of matrix factorization for matrices having discrete entries which are not ordered. We realize that there are important applications in computer vision and social networking of completion of such matrices. We propose to explore this problem in the future.

CHAPTER 5

Multi-label Classification Using Hierarchical Embedding

In the previous chapters, we discussed about matrix factorisation techniques for collaborative filtering. Essentially four techniques where examined therein, namely; (1) Binary rating matrix completion using bi-level matrix factorization; (2) Discrete ordinal rating matrix completion using maximum margin matrix factorization; (3) Hierarchical bi-level matrix factorization to handle matrix completion of discrete ordinal rating matrix and; (4) Proximal matrix factorization for rating matrix completion. In all the four methods one common concept that is discussed is the matrix factorization of the rating matrix. The underlying hypothesis is that, matrix factorization is a process of separation of the characteristics of users and items as factor matrices. In a general setting, one can view matrix factorization as a principle of separating or extracting the component latent factors. Another way of visualizing matrix factorization is as a kind of low-dimensional embedding of the data. This is practically relevant when a matrix is viewed as a transformation of data from one space to the other. Matrix factorization is essentially a composition of two transformations, out of which one is for embedding. Hence, matrix factorization can be used in several scenarios where we seek to capture the underlying low-dimensional structure of the data (142; 151; 21; 141). In this chapter, we explore the application of matrix factorization for yet another important problem of machine learning namely *multi-label classification*.

5.1 Introduction

In a traditional classification problem, data objects (instances) are represented in the form of feature vectors wherein each object is associated with a unique class label from a set of disjoint class labels L, $|L| > 1$. Depending on the total number of disjoint classes in L, a learning task is categorized as *binary* classification (when $|L| = 2$) or *multi-class* classification (when $|L| > 2$) (105). An example of binary classification problem is the email spam filtering problem where the goal is to classify an email into spam or non-spam whereas the classification of email into one of the predefined classes such as *primary*, *social*, *promotions*, *updates*, *forum* etc., falls into the category of multi-class classification. However, in many real-world classification tasks, the data object can simultaneously belong to one or more classes in L. For example, in text categorization, an article can belong to several predefined classes such as *politics*, *sport*, *software*, *business*, *entertainment*; In protein function prediction, a protein can be associated with a set of functional roles such as *metabolism*, *energy*, *cell fate*, *storage protein*, *localization*; In web mining, a web page can be classified as *news*, *academic*, *e-commerce*, *blog*, *forum* etc.; Similarly, in image classification, an image can be annotated with several classes such as *sea*, *sky*, *tree*, *mountain*, *valley*, and so on.

Multi-label learning is concerned with the classification of data with multiple class labels. The objective of multi-label classification is to build a classifier that can automatically tag an example with the most relevant subset of labels. This problem can be seen as a generalization of *single label* classification where an instance is associated with a unique class label from a set of disjoint labels L. The majority of the methods for supervised machine learning proceeds from a formal setting in which data objects (instances) are represented in the form of feature vectors. Thus, an instance x is represented as D dimensional real-valued feature vector $(x_1, \ldots, x_D) \in \mathbb{R}^D$. In multi-label classification, each training example $x_i, 1 \leq j \leq N$ is associated with a label vector $y_i \in \{-1, 1\}^L$. The $+1$ entry at the jth coordinate of vector y_i indicates the presence of label j in data point x_i. Given the pair, a feature matrix $X \in \mathbb{R}^{N \times D}$ and a label matrix $Y \in \{-1, 1\}^{N \times L}$, the task of multi-label classification is to learn a parameterization $h : \mathbb{R}^D \rightarrow \{-1, 1\}^L$ that maps each instance (or, a feature vector) to a set of labels (a label vector). The multi-label classification problem can be formally defined as follows:

Definition 4 (Multi-label Classification). *Given N training examples in the form of a pair of feature matrix X and label matrix Y where each example $x_i \in \mathbb{R}^D, 1 \leq i \leq N$, is a row of X and its associated labels $y_i \in \{-1, 1\}^L$ is the corresponding row of Y. The $+1$ entry at the jth coordinate of vector y_i indicates the presence of label j in data point x_i. The task of multi-label classification is to learn a parametrization $h : \mathbb{R}^D \to \{-1, 1\}^L$ that maps each instance (or, a feature vector) to a set of labels (a label vector).*

Multi-label classification has applications in many areas, such as machine learning (95; 147), computer vision (12; 9), and data mining (115; 103). Existing methods of multi-label classification can be broadly divided into two categories (105; 149) - methods based on *problem transformation* and methods based on *algorithm adaptation*. Former approach transforms the multi-label classification problem into single label classification problems so that existing single-label classification algorithms can be applied. During the last decade, a number of problem transformation techniques are proposed in the literature such as Binary Relevance (BR) (9), Calibrated Label Ranking (32), Classifier Chains (95), Random k-labelsets (115) to name a few. On the other hand, methods based on algorithm adaption extend or adapt the learning techniques to deal with multi-label data directly. Representative algorithms include AdaBoost.MH and AdaBoost.MR (103) which are two simple extensions of AdaBoost, ML-DT (18) adapting decision tree techniques, lazy learning techniques such as ML-kNN (148) and BR-kNN (106) to name a few.

In case of multi-label classification, one of the major scalability issues arises when we have extremely large feature space and label space. Most of the conventional algorithms of multi-label classification fail in such situations. To cope with the challenge of exponential-sized output space, modeling inter-label correlations has been the major thrust of research in the area of multi-label classification in recent years (72; 46; 7) and for this, use of parametrization and embedding have been the prime focus (72; 46; 12; 136; 43). There are two strategies of embedding for exploiting inter-label correlation: (1) Feature Space Embedding (FE); and (2) Label Space Embedding (LE). The first is to learn a projection function which can transform the data from original feature-space to an embedded space. In (43; 44), a direct transformation from original feature-space to label-space is suggested with the assumption that each class label is associated with a

sparse label specific feature. In (136; 12), the inter-label correlation is modelled implicitly using low-rank constraint on the transformation matrix. The debate is going on as to whether it is the low-rank embedding or the label-specific sparse transformation that models the label correlation accurately. The LE approach transforms the label vectors to an embedded space, followed by the association between feature vectors and embedded label space for classification purpose. With proper decoding process that maps the projected data back to the original label space, the task of multi-label prediction is achieved (136; 109; 76). It can be seen that both the approaches are essentially a process of parametrization to overcome the complexity of multi-label classification and most often it is proposed to adopt linear parametrization. Some researchers (61; 73) suggest a natural extension of their proposal of linear parametrization to nonlinear cases but no detailed study is undertaken in this direction. Moreover, all these approaches do not yield results beyond a particular level of accuracy for problems with large data and large number of labels.

Experimental and theoretical study of the recent approaches for multi-label classification reveals many important aspects of the problem. It is clear that a single linear embedding h may not take us very far in finding accurate multi-label classification. There are several reasons for this: the diversity of the training set, the correlation among labels, the feature-label relationship, and most importantly, the learning algorithm to determine the mapping h. Normally, h is determined by a process of nonlinear optimization. A research question that naturally arises is whether there can be a parametrization which is piecewise-linear.

In this chapter, we investigate this aspect and propose a novel method that generates optimal embeddings for subsets of training examples. The proposed method is novel in the sense that it judiciously selects a subset of training examples for training and then it assigns a suitable subset of the training set to an embedding. Using multiple embeddings and their assigned training sets, a new instance is classified and we show that the proposed method outperforms all major algorithms on all major benchmark datasets.

The rest of the chapter is organized as follows. In Section 5.2, we briefly review the earlier research on multi-label learning. Section 5.3 discuss about embedding approach for multi-label classification. The outline of the proposed method is described in

Section 5.4. We introduce our proposed method, termed as MLC-HMF in Section 5.5. Experimental analysis of the proposed method is reported in Section 5.6. Finally, Section 5.7 concludes and indicates several issues for future work.

5.2 Multi-label Classification Approaches

During the past decade, the problem of multi-label classification has been widely dealt with and different methods have been proposed which can be broadly classified into two categories (113) - methods based on problem transformation and method based on algorithm adaptation. In this section, we present a brief literature review about these two approaches for multi-label classification.

5.2.1 Problem Transformation Approach

A common approach to multi-label classification problem is to decompose the problem into one or more single-label (binary or multi-class) problems. There are several problem transformation techniques proposed in the literature which includes Binary Relevance (9), Calibrated Label ranking (32) and Classifier Chains (95). Here we review these fundamental methods.

Binary Relevance (BR) (9): BR is a popular problem transformation approach that decomposes the multi-label classification with L labels into L independent binary class classification problem. The BR approach learns L independent binary classifiers h_l : $\mathbb{R}^D \to \{\pm 1\}$, $1 \leq l \leq L$, where each binary classifier corresponds to a label in L. For each binary classifier h_l, the binary training-set D_l is obtained by transforming the original data set D using the following rule.

$$D_l = \{(x_i, \varphi(y_i, l)) | 1 \leq i \leq N\} \tag{5.1}$$

where

$$\varphi(y_i, l) = \begin{cases} +1, & \text{if } l\text{th label is in } y_i \\ -1, & \text{otherwise} \end{cases} \tag{5.2}$$

Once these data sets are ready, some binary classification algorithm is utilized to train

the binary classifier h_l corresponding to the prediction of binary association of the lth label. For any unseen instance $x_{new} \in \mathbb{R}^D$, the label vector y_{new} is predicted by combining the output of the L classifiers i.e., the union of labels predicted positively by L binary classifiers.

Table 5.1: Multi-label data set.

#	Features				Labels		
	a_1	a_2	a_3	a_4	l_1	l_2	l_3
x_1	-0.71	-0.33	0.40	0.80	1	-1	-1
x_2	0.01	-0.57	-0.34	0.91	-1	1	1
x_3	-0.08	-0.33	0.11	-0.31	1	1	1
x_4	0.10	-0.20	0.33	0.92	1	-1	1
x_5	0.08	0.09	- 0.41	0.60	-1	1	-1
	X				Y		

Let us consider the multi-label data set in Table 5.1 with five instances $x_1, \ldots, x_5$ and three target labels denoted as l_1, l_2 and l_3. Each instance is represented by a set of features $a_1, \ldots, a_4$. We will refer the feature vector of the kth instance by x_k in the subsequent discussion.

The BR method transforms the original multi-label training-set in Table 5.1 into three binary training-sets, one for each label. The transformed data set D_l for each classifier h_l is shown in Table 5.2. After that, some binary classification algorithm is utilized to train the binary classifier h_l on data set D_l.

Table 5.2: Training-set D_l for each classifier h_l.

#	l_1
x_1	1
x_2	-1
x_3	1
x_4	1
x_5	-1
D_1	

#	l_2
x_1	-1
x_2	1
x_3	1
x_4	-1
x_5	1
D_2	

#	l_3
x_1	-1
x_2	1
x_3	1
x_4	1
x_5	-1
D_3	

The unseen instance x_{new} is then fed into all the binary classifiers and the corresponding label vector prediction y_{new} is obtained by aggregating the output of the individual classifier. Table 5.3 illustrates the classification process using BR approach on unseen instance x_{new}.

Finally, the prediction of each binary classifier h_1, h_2 and h_3 is combined to obtain the label vector prediction $y_{new} = \{+1, -1, -1\}$.

Table 5.3: Binary Relevance example.

Classifiers	Prediction
$h_1 : x_{new} \to \{l_1, \neg l_1\} \to \{+1, -1\}$	$+1$
$h_2 : x_{new} \to \{l_2, \neg l_2\} \to \{+1, -1\}$	$+1$
$h_3 : x_{new} \to \{l_3, \neg l_3\} \to \{+1, -1\}$	-1

Calibrated Label ranking (CLR) (32): The CLR method transforms the original data sets into $\frac{L(L-1)}{2}$ binary label data set, one for each label pair (j, k), $1 \leq j < k \leq L$. Each transformed data set D_{jk} retains instances which are associated with only one label from the label pair (j, k). The data set D_{jk} is obtained using the following rule.

$$D_{jk} = \{(x_i, \varphi(y_i, j, k)) | \varphi(y_i, j) \neq \varphi(y_i, k), 1 \leq i \leq N\} \tag{5.3}$$

where

$$\varphi(y_i, j, k) = \begin{cases} +1, & \text{if } \varphi(y_i, j) = +1 \text{ and } \varphi(y_i, k) = -1 \\ -1, & \text{if } \varphi(y_i, j) = -1 \text{ and } \varphi(y_i, k) = +1 \end{cases} \tag{5.4}$$

A binary classifier $h_{jk} : \mathbb{R}^D \to \{\pm 1\}$ is then trained on each data set D_{jk}. Given an unseen instance x_{new}, firstly all $\frac{L(L-1)}{2}$ binary classifiers are invoked and the ranking of labels is obtained by counting the votes x_{new} receives for each label. Finally, the label vector y_{new} is then obtained from the ranked label list using some thresholding function.

For example in Table 5.1, the CLR approach transforms the original multi-label data set into $\frac{3(3-1)}{2}$ i.e., 3 binary label data sets, one for each label pair. The transformed data sets D_{jk}, $1 \leq j < k \leq 3$ for each label pair is shown in Table 5.4.

Table 5.4: Training-set D_{jk} for each classifier h_{jk}.

#	l_{12}
x_1	1
x_2	-1
x_4	1
x_5	-1

D_{12}

#	l_{13}
x_1	1
x_2	-1

D_{13}

#	l_{23}
x_4	1
x_5	-1

D_{23}

After that, a binary classifier h_{jk} is learnt on the transformed data D_{jk}, $1 \leq j < k \leq 3$. Given an unseen instance x_{new}, let it be the case that the classifiers l_{12}, l_{13} and l_{23} has predicted l_1, l_2 and l_2, respectively. After counting the votes x_{new} receives from each classifier, h_{jk} for each label, the inferred label ranking is $l_2 > l_1 > l_3$.

Classifier Chains (CC) (95): The BR approach trains L independent binary classifiers,

one for each label. However, its performance can be poor when strong inter-label correlations exist i.e., information of one label is helpful in inferring the information about related label. The CC approach model the inter-label correlations by transforming the multi-label classification problem into a chain of binary classification problems. Each binary classifier in the chain pass the label information to the subsequent classifiers in the chain. The chain involves same number of binary classifiers as in BR but extends the feature space of each binary classifier with all prior binary relevance predictions in the chain. Given L labels $\{1, 2, \ldots, L\}$, let $\pi : \{1, 2, \ldots, L\} \rightarrow \{1, 2, \ldots, L\}$ be the permutation function which specifies the ordering of binary classifier in the chain. Each classifier $h_{\pi(j)}$ is responsible for training and prediction of the binary association for jth label in the ordered chain. The binary training-set $D_{\pi(1)}$ for the first binary classifier $h_{\pi(1)}$ in the chain is obtained using Eq. (5.1). For subsequent classifiers $h_{\pi(j)}$ in the chain, the original training-set gets appended using the following rule.

$$D_j = \{([x_i, pre(h_{\pi(1)}), \ldots, pre(h_{\pi(j-1)})], \varphi(y_i, \pi(j))) | 1 \leq i \leq N\} \qquad (5.5)$$

where $pre(h_{\pi(l)})$ is the prediction of lth binary classifier for x_i and $[x_i, pre(h_{\pi(1)}), \ldots, pre(h_{\pi(j-1)})]$ concatenates vector x_i with prior binary predictions $pre(h_{\pi(1)}), \ldots, pre(h_{\pi(j-1)})$. For any unseen instance $x_{new} \in \mathbb{R}^D$, the classification process begins at $h_{\pi(1)}$ and propagates along the chain $h_{\pi(2)}, \ldots, h_{\pi(L)}$. The label vector y_{new} is predicted by combining the output of each classifier $h_{\pi(l)}$ in the chain.

Table 5.5: Training-set $D_{\pi(l)}$ for each classifier $h_{\pi(l)}$.

# $[x]$	$pred(h_1)$
x_1	1
x_2	-1
x_3	-1
x_4	1
x_5	-1

D_1

# $[x, pred(h_1)]$		$pred(h_2)$
x_1	1	-1
x_2	-1	1
x_3	-1	1
x_4	1	1
x_5	-1	-1

D_2

# $[x, pred(h_1), pred(h_2)]$			$pred(h_3)$
x_1	1	-1	-1
x_2	-1	1	1
x_3	-1	1	1
x_4	1	1	1
x_5	-1	-1	1

D_3

For the example in Table 5.1, let the training order of the classifier obtained by permutation function π is $h_1 \rightarrow h_2 \rightarrow h_3$. The CC approach first trains a binary classifier h_1 for label 1; For label 2, the feature-space of binary classifier h_2 is extended by the prediction of classifier h_1; Similarly, for label 3, the feature-space of the binary classifier h_3 is extended by including the predictions of classifiers h_1 and h_2 (Table 5.5).

The new instance x_{new} (let, $x_{new} = [-0.05, -0.1, 0.3, 0.5]$) is then fed into all classifiers from h_1 to h_3 and the prediction is obtained by aggregating the output of the individual classifier. Table 5.6 illustrates the classification process of CC approach on unseen instance x_{new}.

Table 5.6: Classifier Chain example.

Classifiers	Prediction
$h_1 : [-0.05, -0.1, 0.3, 0.5] \quad \rightarrow \{l_1, \neg l_1\} \rightarrow \{+1, -1\}$	$+1$
$h_2 : [-0.05, -0.1, 0.3, 0.5, \mathbf{1}] \quad \rightarrow \{l_2, \neg l_2\} \rightarrow \{+1, -1\}$	-1
$h_3 : [-0.05, -0.1, 0.3, 0.5, \mathbf{1}, \mathbf{-1}] \rightarrow \{l_3, \neg l_3\} \rightarrow \{+1, -1\}$	-1

Finally, the prediction of each binary classifier h_1, h_2 and h_3 is combined to obtain the label vector prediction $y_{new} = \{+1, -1, -1\}$.

Random k-Labelsets (RAkEL) (115): The RAkEL approach is based on the label powerset (LP) method which cast the multi-label classification problem into multi-class classification problem. In LP approach, the multi-label data is first transformed into a multi-class data by considering each unique subset of labels that exist in the data set as one class. The transformation is achieved using some *injective* function mapping $\sigma : 2^y \rightarrow \mathbb{N}$ from the power set of labels to a natural number. Once the multi-class data set is ready, some multi-class classification algorithm is utilized to train the classifier $h : x \in \mathbb{R}^D \rightarrow \Gamma$, where Γ is the set of distinct classes in the transformed data. The major drawbacks with LP approach are: a) Computational complexity incurred due to the large number of unique labelsets. Multi-label classification problem with L labels will have $(min(N, 2^L))$ possible unique label powerset; b) Class imbalance problem, as a large number of labelsets would be associated with very few training instances; and c) Confined to predict labelsets observed in the training-set.

RAkEL extends the concept of LP by constructing an ensemble of LP classifiers (114; 115). Each LP classifier is trained using a small randomly selected k-labelsets from the original labels which result in computationally inexpensive, predictive complete and more balanced multi-class training-set. For example, if we break a data set with 101 labels into 3-labelset with the assumption that two labelset will not overlap, then in the worst case scenario 2^3 binary classifiers need to be trained for each 3-labelset and $2^3 \times 27$ binary classifiers for overall data set whereas for full LP requires 2^{101} binary classifiers. Given the labelset size k, let L^k denote the collection of all distinct k-labelsets in L.

The pth k-labelsets is denoted as $L^k(p) \subseteq L$ where $|L^k(p)| = k$ and $1 \leq p \leq \binom{L}{k}$. For each k-labelsets $L^k(p)$, a multi-class training-set is constructed as

$$D_{L^k(p)} = \{x_i, \sigma : (y_i \cap L^k(p)) \to \mathbb{N} | 1 \leq i \leq m\}. \qquad (5.6)$$

After that, some multi-class classification algorithm is utilized to train the classifier $h_{L^k(p)} : x \in D_{L^k(p)} \to \Gamma$, where Γ is the set of distinct classes in $D_{L^k(p)}$. Given the size of labelset k and the number of LP classifiers n, RAkEL creates an ensemble of n LP classifiers. For each LP classifier k-labelsets $L^k(p_r)$, $1 \leq r \leq n$, is randomly selected and for each, a multi-class classifier $h_{L^k(p_r)}$ is then trained. The unseen instance x_{new} is then fed into all the LP classifiers. After that, RAkEL calculates the average vote received for each label and the final output label vector y_{new} is predicted by setting a threshold τ on the average vote. For the example in Table 5.2, the classification process of RAkEL with $k = 2$, $n = 3$ and $\tau = 0.5$ is shown in Table 5.7.

Table 5.7: RAkEL example.

Classifiers	Prediction		
	l_1	l_2	l_3
$h_{LP_{1,2}}$	-1	$+1$	-
$h_{LP_{1,3}}$	-1	-	$+1$
$h_{LP_{2,3}}$	-	$+1$	-1
Average vote	$0/2$	$2/2$	$1/2$
y_{new}	-1	$+1$	$+1$

5.2.2 Algorithm Adaption Approach

This category of algorithms modifies the well-known learning algorithms to tackle multi-label classification problem directly. There are several algorithm adaption techniques proposed in the literature such as AdaBoost.MH and AdaBoost.MR (103), Multi-Label Decision Tree (18) and Multi-Label k-Nearest Neighbor (148). Here we review these fundamental methods.

Multi-Label k-Nearest Neighbor (ML-kNN) (148): It is a multi-label lazy learning approach derived from the traditional k-Nearest Neighbor (kNN) algorithm. For each unseen instance x_{new}, ML-kNN first identifies its k nearest neighbors $N(x_{new})$ in the training set. For every label j, ML-kNN then computes the count $C(j)$ which records

the number of x_{new} neighbors with label j. The membership counting vector for x_{new} can be defined as follows.

$$C(j) = \sum_{x_a \in N(x_{new})} y_a(j) == 1, j \in \{1, 2, \ldots, L\} \tag{5.7}$$

where $y_a(j)$ takes the value of 1 if instance x_a is associated with label j and 0 otherwise. The following maximum a posteriori (MAP) principle is then utilized to determine the label set y_{new} for the unseen instance x_{new}.

$$y_{new}(j) = \underset{b \in \{-1, +1\}}{arg\ max}\ P(H_b^j | E_{C(j)}^j), j \in \{1, 2, \ldots, L\} \tag{5.8}$$

where H_{+1}^j be the event that x_{new} is associated with label j and H_{-1}^j be the event that x_{new} is not associated with label j. $E_{C(j)}^j$ is the event that exactly $C(j)$ number of x_{new} neighbors has label j. Using the Bayes theorem, the same can be re-written as follows.

$$y_{new}(j) = \underset{b \in \{-1, +1\}}{arg\ max}\ P(H_b^j|)P(E_{C(j)}^j | H_b^j), j \in \{1, 2, \ldots, L\} \tag{5.9}$$

Multi-Label Decision Tree (ML-DT) (18): ML-DT extend the decision tree algorithm designed for multi-class classification to the multi-label setting. In ML-DT, the decision tree is constructed recursively in the top down manner. At every node of the tree a feature (F) is chosen which best classifies the remaining training examples. The best feature (F) is decided by considering the following information gain (IG) measure.

$$IG(X', Y', F) = entropy(X', Y') - \sum_{v \in F} \frac{|X_v'|}{|X'|} * entropy(X_v', Y_v') \tag{5.10}$$

where F is the feature under consideration, (X', Y') is the set of training examples present at the current node, (X_v', Y_v') is the subset of examples with value v for feature F and $entropy(X', Y')$ is defined as follows.

$$entropy(X', Y') = -\sum_{j}^{L} P(j) \log P(j) + (1 - P(j)) \log(1 - P(j)) \tag{5.11}$$

where $P(j)$ is the proportion of examples (X', Y') associated with label j. After the learning process is over, each leaf node will be associated with a set of class labels. For any unseen instance x_{new}, the tree is traversed from root to the leaf along the path until

reaching a leaf node. The label associated with the leaf node is predicted as a label set for x_{new}.

Tree Based Boosting (103): AdaBoost.MH and AdaBoost.MR are the multi-label version of the popular ensemble method AdaBoost which creates a strong classifier by combining many base or weak classifiers. AdaBoost.MH tries to minimize the *hamming loss* and AdaBoost.MR is designed to find hypotheses based on label ranks. In AdaBoost.MH, each training example (x_i, y_i) is presented as L binary examples of the form $\{([x_i, j], \varphi(y_i, j)) | 1 \leq j \leq L\}$. AdaBoost.MH maintains a distribution over examples and labels $(X \times Y)$ and re-weights the examples at each boosting round t. The example-label pairs that were misclassified in the previous round have higher weight. Unlike the Adaboost.MH which tries to minimize the *hamming loss*, Adaboost.MR tries to find hypothesis that *ranks* the labels of any instance in such as a way that correct labels place at the top of the ranking.

5.3 Embedding based Approach

Most of the conventional algorithms for multi-label classification perform well on relatively small sized data but have difficulty in dealing with data where feature and label space are sufficiently large. For example, simple methods such as Binary Relevance (BR), that treat each label as a separate binary classification problem fail miserably. In many real-world applications of multi-label classification, often feature and label space are assumed to be extremely large. Under such conditions, the scalability of learning algorithms is of major concern, calling for an effective data management and the use of appropriate data structures for time- and space-efficient implementations. To cope with the challenge of exponential-sized output space, modelling feature- and label-space correlation has been the major thrust of research in the recent years (72; 46; 7). Various approaches have been proposed to exploit the intrinsic information in feature- and label-space. The CC (95) discussed in Section 5.2 captures the label correlation by extending the feature space of each binary classifier with all prior binary relevance predictions in the chain. In (155), a maximum entropy method for multi-label classification is proposed in which mutual correlations among data are explicitly considered in the model. In (33), the dependencies between an individual feature and labels are modelled using

the conditional random field. Zhang et al. (146) model the joint distribution of the label space conditioned on the feature space using Bayesian Networks. In (45), multiple boosted learners are trained simultaneously, one for each label with the assumption that the hypothesis generated for one label can be helpful for the other; each hypothesis not only looks into its own single-label task but also reuses the trained hypotheses from other labels. These methods of exploiting correlation information can be quite effective in multi-label learning, but they are computationally expensive over the exponentially large feature- and label-space.

In recent years much attention on multi-label classification have been devoted to embedding based approach where the prime focus is to reduce the effective number of features and labels (41; 136; 5; 53). The embedding based approach assumes that there exists a low-dimensional space onto which the given set of feature vectors and/ or label vectors can be embedded. The intrinsic relationship among feature- and label-space is invariant when constraining embedded dimension to be significantly lower than original space (137; 109; 136). The embedding strategies can be grouped into two categories namely; (1) Feature space embedding; and (2) Label space embedding. Feature space embedding aims to design a projection function which can map the instance in original feature space to label space. Feature space embedding can be further divided into two categories - the first is to learn a projection function which directly maps instances from feature space to label space and simultaneously preserve intrinsic relationship in the original space through regularization (43; 44). The second approach models inter-label correlation implicitly using low-rank constraints on embedding (136; 12). The aim here is to design a projection function which can map the instance in original feature space to a reduced dimensional space and at the same time preserve the intrinsic information in the original feature space. A mapping is then learnt from reduced dimensional space to label space. On the other hand, the label space embedding approach transform the label vectors to an embedded space via linear or local non-linear embeddings, followed by the association between feature vectors and embedded label space for classification purpose. With proper decoding process that maps the projected data back to the original label space, the task of multi-label prediction is achieved (41; 94; 109). Some researchers suggest a simultaneous embedding of feature and label space onto the same space. Embedding of both features and labels to a single low-rank space is no way obvious and cannot be a routine extension of feature-only embedding or label-only

embedding. It is necessary to retain the intrinsic relationship while mapping feature and label vectors to a single space. In the following subsections, we present the brief literature review of feature- and label-space embedding.

5.3.1 Feature Space Embedding (FE)

Given N training examples in the form of a pair of feature matrix X and label matrix Y, the goal of FE is to transform each D-dimensional feature vector (a row of matrix X) from original feature space to a L-dimensional label vector (corresponding row in Y) by an embedding function $\mathcal{F}: X \to Y$. There are two strategies of FE. As illustrated in
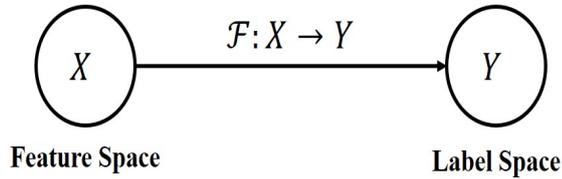
Figure 5.1: Feature space embedding

Figure 5.1, the first approach is to learn a mapping which directly transforms instance from feature space to label space. Usually, the mapping is achieved by a transformation matrix $W = [w_{.1}, w_{.2}, \ldots, w_{.L}] \in \mathbb{R}^{D \times L}$ which directly maps instances from feature space to label space. The vector $w_{.l} \in \mathbb{R}^D$, i.e., the lth column of a W can be seen as a transformation vector for the lth label. Most of the approaches falling into this category assume that the two strongly correlated labels share more features with each other than two uncorrelated labels and hence their corresponding columns in W will be similar (43; 44). The second approach illustrated in Figure 5.2 assume that the label
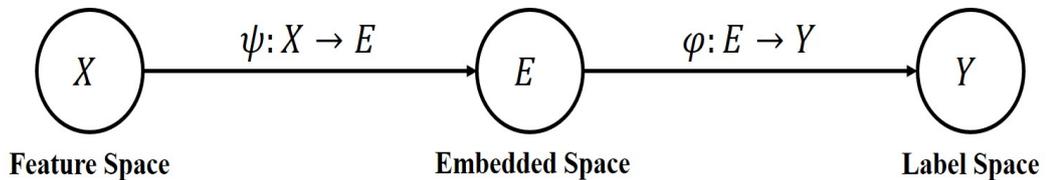
Figure 5.2: Feature space embedding

matrix Y is low-rank due to the presence of similar labels and thus model the inter-label correlation implicitly using low-rank constraints on the transformation matrix W. The algorithms falling into this category first embed each D-dimensional feature vector x to a d-dimensional ($d \ll D$) vector $e \in E \subseteq \mathbb{R}^d$ in a latent space by an embedding

function $\psi : X \to E$. Then, the algorithm train a predictive model $\varphi : E \to Y$. For an unseen instance x_{new}, a low-dimensional embeding $e_{new} = \psi(x_{new})$ is firstly obtained and then the label vector y_{new} is predicted by the decoding function $\varphi(e_{new})$. We briefly review the major approaches of FE.

As discussed previously, the goal of FE is to learn a transformation matrix W and a common formulation is the following optimization problem.

$$\min_{W} \ell(Y, XW) + \lambda R(W) \tag{5.12}$$

where $W \in \mathbb{R}^{D \times L}$, $\ell(\cdot)$ is a loss function that measures how well XW approximates Y, $R(\cdot)$ is a regularization function that promotes various desired properties in W (low-rank, sparsity, group-sparsity, etc.) and the constant $\lambda \geq 0$ is the regularization parameter which controls the extent of regularization. Huang et al. (43) assume that each label is associated with a subset of features from the original feature set and two strongly correlated labels share more features with each other. The label specific features (a subset of original feature space) learning problem is modelled by linear regression framework with sparsity constraints on the regression parameter W. The problem can be formulated as follows.

$$\min_{W} \ell(Y, XW) + \frac{\alpha}{2}\phi(W) + \lambda\|W\|_1 \tag{5.13}$$

where $\| \cdot \|_1$ is ℓ_1 norm, the second term in Eq. (5.13) models inter-label correlation with the constant $\alpha \geq 0$ to control the extent of correlation.

The above approaches require $D \times L$ parameters to model the classification problem, which will be expensive when D and L are large (136). To reduce the training cost, a generic empirical risk minimization (ERM) framework with low-rank constraint on linear parametrization $W = UV$, where $U \in \mathbb{R}^{D \times d}$ and $V \in \mathbb{R}^{d \times L}$ are of rank $d \ll D$ is proposed in Yu et al. (136). The ith row of matrix $E = XU$ can be seen as a latent space representation of ith training instance whereas the lth cloumn of matrix V can be visualized as a latent representation of label l. The basic assumption here is that although the original space is sufficiently large, the intrinsic relationship in original space can be captured by representing the instances and labels using a small number of

latent factors. The problem can be formulated as

$$\min_{U,V} \ell(Y, XUV) + \frac{\lambda}{2}(\|U\|_F^2 + \|V\|_F^2) \qquad (5.14)$$

where $\|\cdot\|_F$ is Frobenius norm, $\ell(\cdot)$ is a loss function that measures how well XUV approximates Y, and the constant $\lambda \geq 0$ is the regularization parameter which controls the extent of regularization. A semi-supervised joint learning framework in which dimensionality reduction and multi-label classification are performed simultaneously is proposed in (137). To guide the multi-label learning process, it uses local invariance properties that if two instances are similar in original space then their low-dimensional representation will also be similar. The problem is formulated as

$$\min_{U,V} \sum_{l=1}^{L} \sum_{i=1}^{N} h(y_{il}(x_i UV_l)) + \frac{\alpha}{2} \sum_{i,j=1}^{N} e_{i,j} \|x_i U - x_j U\|_F^2 + \frac{\lambda}{2} \|V\|_F^2 \qquad (5.15)$$

where V_l is the lth column of V, $h(\cdot)$, λ, α and $\|\cdot\|_F$ are same as defined previously and $e_{i,j}$ is the similarity between instance x_i and x_j in original space. The second term in Eq. (5.15) makes the transformation matrix U neighborhood aware i.e., if two instances x_i and x_j are close in the original space, then their low-dimensional representation are required to be close.

5.3.2 Label Space Embedding (LE)

Given N training examples in the form of a pair of feature matrix X and label matrix Y, the goal of LE is to transform each L-dimensional label vector (a row of matrix Y) from original label space to a d-dimensional embedded vector $e \in E \subseteq \mathbb{R}^d$ by an embedding function $\Phi : Y \rightarrow E$. Then, a predictive model $\psi : X \rightarrow E$ is trained from original feature space to embedded space. With proper decoding process $\varphi : E \rightarrow Y$ that maps the projected data back to the original label space, the task of multi-label prediction is achieved (6; 32; 41). Figure 5.3 illustrate the basic principle of LE. We briefly review the major approaches of LE.

The approach of Hsu et al. (41) projects the label vectors to a random low-dimensional space, fits a regression model in this space, then projects these predictions back to the original label space. In (109), principal component analysis (PCA) is employed on the
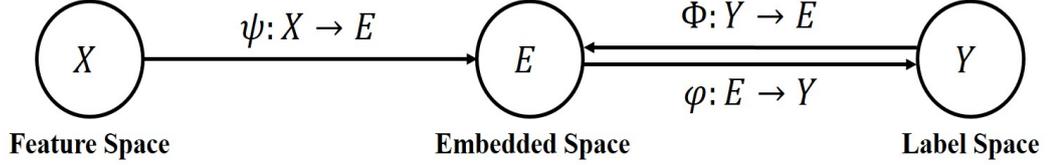
Figure 5.3: Label space embedding

label covariance matrix to extract a low-dimensional latent space. In (3), a sparsity-regularized least square reconstruction objective is used to select a small set of labels that can predict the remaining labels. Recently, Yu et al. (136) and Jing et al. (53) proposed to use trace norm regularization to identify a low-dimensional representation of the original large label space. Mineiro et al. (55) use randomized dimensionality reduction to learn a low-dimensional embedding that explicitly captures correlations between the instance features and their labels. Some methods work with label or feature similarity matrices, and seek to preserve the local structure of the data in the low-dimensional latent space. Prabhu et al. (92) propose a method to train a classification tree by minimizing the Normalized Discounted Cumulative Gain. Rai et al. (94) assume that the label vectors are generated by sampling from a weighted combination of label topics, where the mixture coefficients are determined by the instance features. Based on the assumption that all the output labels can be recovered by a small subset, multi-label classification via column subset selection approach (CSSP) is proposed in (6). Given a matrix Y, CSSP seeks to find a column index set $C \subset \{1, \ldots, L\}$ with cardinality d $(d \ll L)$ so that the columns with indices in C can approximately span Y. The subset of columns is selected using a randomized sampling procedure. The problem is formulated as follows.

$$\min_{C} \|Y - Y_C Y_C^{\dagger} Y\|_F \tag{5.16}$$

where $\| \cdot \|_F$ is Frobenius norm, Y_C denotes the submatrix consisting of columns of Y with indices in C and $Y_C Y_C^{\dagger}$ is the projection matrix onto the d-dimensional space spanned by columns of Y_C. Alternatively, there have been emerging interests in recent multi-label methods that take the correlation information as prior knowledge while modeling the embedding (encoding). These methods can be efficient when the mapped label space has significantly lower dimensionality than the original label space (41).

In recent years, matrix factorization (MF) based approach is frequently used to achieve the LE which aims at determining two matrices $U \in \mathbb{R}^{N \times d}$ and $V \in \mathbb{R}^{d \times L}$.

The matrix U can be viewed as the *basis matrix*, while the matrix V can be treated as the *coefficient matrix* and a common formulation is the following optimization problem.

$$\min_{U,V} \ell(Y, U, V) + \lambda R(U, V) \tag{5.17}$$

where $\ell(\cdot)$ is a loss function that measures how well UV approximates Y, $R(\cdot)$ is a regularization function that promotes various desired properties in U and V (sparsity, group-sparsity, etc.) and the constant $\lambda \geq 0$ is the regularization parameter which controls the extent of regularization. In (76), a MF based approach is used to learn the label encoding and decoding matrix simultaneously. The problem is formulated as follows.

$$\min_{U,V} \|Y - UV\|_F^2 + \alpha \Psi(X, U) \tag{5.18}$$

where $U \in \mathbb{R}^{N \times d}$ is the code matrix, $V \in \mathbb{R}^{d \times L}$ is the decoding matrix, $\Psi(X, U)$ is used to make U feature-aware by considering correlations between X and U as side information and the constant $\alpha \geq 0$ is the trade-off parameter. In order to reduce the noisy information in the label space, the method proposed in (52) decompose the original space to a low-dimensional space. Instead of globally projecting onto a linear low-rank subspace, the method proposed in (5) learns embeddings which non-linearly capture label correlations by preserving the pairwise distances between only the closest (rather than all) label vectors.

5.4   Outline of the Proposed Approach

In this section, we introduce the underlying principle of our proposed method, termed as MLC-HMF (*Multi-label classification using hierarchical embedding*). As discussed in Section 5.1, the proposed method draws motivation from embedding based approach of multi-label classification. The embedding approach aims at modelling the intrinsic information existing in the original space such as label correlation, instance correlation etc., to assist the learning process. The basic assumption underlying the embedding based approach is that the label matrix is low-rank and inter-label correlation can be modelled implicitly by embedding the feature/ label vectors to a reduced low-dimensional latent space. There are two major approaches for embedding namely *feature space embedding* and *label space embedding*. A detailed discussion of these

approaches is given in Section 5.3. A common characteristic of both the approaches is that they try to transform the data from original space to a more manageable (low-dimensional) space such that the learning process can be tackled efficiently without significant loss of prediction performance. One encouraging property of this low-dimensional space is that most of the structures in the original output label space can be explained and recovered. The latent low-dimensional space can be achieved via a *linear* or *non-linear* transformation and most often it is proposed to use linear transformation due to its attractive computational properties and high interpretability (73; 6; 109; 41). However, some researchers (61; 73) suggest a natural extension of their proposal of linear embedding to nonlinear cases but no detailed study is undertaken in this direction.

In many real-word applications, the low-rank assumption made by embedding methods is violated due to several reasons such as the diversity of the training set, the correlation among labels, the feature-label relationship, the learning algorithm to determine the mapping h and most importantly, the presence of *tail* labels, i.e., the infrequent labels associated with only few training examples (131; 5). Hence, embedding with a global projection (a common transformation for all feature/ label vectors) can be complicated and may not well model the optimal mapping. However, on the other hand, we observe through experimental analysis, that it is possible to have efficient embedding for subsets of training samples. Our research is based on the fact that a single linear embedding h may not take us very far in finding accurate multi-label classification. We hypothesize that feature vectors which conform to similar embedding are similar in some sense. Thus, unlike the traditional embedding approach found in the literature, we propose a piecewise-linear embedding of feature-space that generates optimal embeddings for a subset of training examples. Our method is novel in the sense that it judiciously selects a subset of training examples for training and then it assigns a suitable subset of the training set to an embedding. Finally, an unseen instance is classified using the multiple embeddings and their assigned training sets. In this section, we outline the basic principle of the proposed MLC-HMF, an approach based on feature-space embedding.

We start with the formulation given in Eq. (5.14). For exploiting correlations in the labels, one way is to factor the matrix $W = UV$ where $U \in \mathbb{R}^{D \times d}$ can be interpreted as an embedding of the features X into a d dimensional latent space and the lth column of $V \in \mathbb{R}^{d \times L}$ is a linear classifier corresponding to label l on this space. Regularization is provided by constraining the dimensionality of the latent space. The minimization in U

and V is unfortunately non-convex, and Fazel et al. (28) discovered the nuclear norm (sum of singular values) heuristic for matrix rank minimization, which is the convex relaxation of the rank minimization problem. Since XUV yields continuous values and Y is discrete, a natural choice is to use the well-known principle of maximum margin matrix factorization (MMMF) (97; 66). For a subset of training examples, the process of determining the embedding using principle of MMMF is described below.

Computing U, V**:** Let S $\subseteq$ {1, 2, ..., n} be the indices of current set X^S of training examples and the corresponding label vectors is submatrix Y^S of Y. We use smooth hinge loss function to determine U^S and V^S for a given training set (X^S, Y^S). For sake of simplicity, we drop the suffix S. The problem can be formulated as the following minimization problem.

$$\min_{U,V} J(U, V) = \sum_{l=1}^{L} \sum_{i \in S} h(y_{il}(x_i U V_{.l})) + \frac{\lambda}{2}(\|U\|_F^2 + \|V\|_F^2) \tag{5.19}$$

where $V_{.l}$ is lth column of V, $\|\cdot\|$, λ are same as defined previously and $h(\cdot)$ is smooth hinge loss function defined as

$$h(z) = \begin{cases} 0 & \text{if } z \geq 1; \\ \frac{1}{2}(1 - z)^2 & \text{if } 0 < z < 1; \\ \frac{1}{2} - z & \text{otherwise.} \end{cases} \tag{5.20}$$

The detailed discussion about smooth hinge loss is given in Section 3.2. The gradient of the variables to be optimized is determined as follows.

$$\frac{\partial J}{\partial U_{pq}} = \lambda U_{pq} + \sum_{l=1}^{L} \sum_{i=1}^{n} y_{il} h'(y_{il}(x_i U V_{.l})) x_{iq} V_{lp} \tag{5.21}$$

$$\frac{\partial J}{\partial V_{rs}} = \lambda V_{rs} + \sum_{i=1}^{n} y_{ir} h'(y_{ir}(x_i U V_{.r})) x_i U_{.s} \tag{5.22}$$

where $U_{.s}$ is the sth column of U. Gradient descent algorithm and its variants such as conjugate gradient descent and stochastic gradient descent start with random U and V and these are iteratively updated using the equations given in (5.23) and (5.24). Suffixes

t and $t + 1$ are used to indicate current values and updated values, respectively.

$$U_{pq}^{t+1} = U_{pq}^t - c\frac{\partial J}{\partial U_{pq}^t} \qquad (5.23)$$

$$V_{rs}^{t+1} = V_{rs}^t - c\frac{\partial J}{\partial V_{rs}^t} \qquad (5.24)$$

where c is the step length parameter. It is seen that conjugate gradient descent technique exhibits faster rate of convergence and we follow this technique in the present study.

Geometrical Interpretation: As discussed in Section 3.3, application of maximum margin factorization has an interesting geometrical interpretation in the present context. The matrix U maps each D-dimensional feature vector to d-dimensional space and each row of V defines a decision hyperplane associated with the respective label. An accurate embedding implies that each of the decision hyperplanes in d-dimension classifies the embedded feature points conforming to the respective rows of the label-matrix Y. Prompted by the above observations, we propose a novel algorithm MLC-HMF which is discussed in the following section.

5.5 MLC-HMF: The Proposed Method

In this section, a novel method for multi-label classification is proposed which is essentially a hierarchical matrix factorization method. Our algorithm initially separates out training examples into two disjoint components (the process of grouping is discussed later in the section). Then for each component, U and V are learnt. The U and V so computed are used to test the training vectors in the respective components so that instances which are not classified correctly would be further processed for the next round of recursion. The MLC-HMF returns subsets of the training set and the associated (U, V). It also generates residue of training examples which are not used for further classification. In a sense, our algorithm hierarchically selects training set with different degree of suitability for multi-label classification. The depth of the hierarchy determines the finer level of residue.

Identification of Subsets of Training Instances: The instance group are not given explicitly and it is necessary to learn from the feature matrix X. Thus, before the application of MMMF, we use *k-means* clustering (48) with Euclidean metric to cluster the

training set into two clusters and each of the clusters are used separately to determine the corresponding embedding. We do not claim that the above process is the best possible practice to embed the subsets of training samples which are similar in some sense. Actually, the grouping can be implemented in different ways based on the clustering algorithm, the number of clusters, or even more sophisticated distance measure other than the Euclidean metric. Nevertheless, our simple construction process of grouping yield competitive performance as shown in Section 5.6.

Guiding Embedding Process Through Classification Error: As discussed previously, U and V are determined through an optimization process and application of any of the gradient-based algorithms may end up in a local minimizing point. As a result, the resultant embedding may not yield accurate classification even for the training instances that are used for optimization. We consider the classification error during the learning process to associated the matrices U and V with a subset of instances. In the proposed method, we assign the training examples that are classified to an acceptable accuracy to the current embedding and separate out wrongly classified example for further processing recursively.

Algorithm 5 outlines the main flow of the proposed method. For each node in the tree, a joint learning framework given in Eq. (5.19) with low-rank constraint on the parametrization (embedding) and multi-label classification is performed simultaneously. At every node, we maintain the mapping U and the label feature matrix V along with the training examples whose hamming loss is less than the threshold $\mathcal{T}$. The remaining training instances are divided into two parts according to the *k-means* clustering. This process is repeated recursively until either the number of instances in the node is too small or the depth of the tree exceeds a given threshold.

Classification: The outcome of the learning process described in MLC-HMF has a tree structure. At each non-leaf node of the tree, there is a disjoint subset of training samples along with embedding U and label feature matrix V. For any unseen instance $x_{new} \in \mathbb{R}^D$, the label vector is predicted by first finding the $U's$ and $V's$ associated with the *k*-nearest neighbor instances present at non-leaf node of tree. Let $(U^i, V^i), 1 \leq i \leq K$, be the ith embedding pair. A label vector is predicted with the help of every (U^i, V^i)-pair using the rule $sign(x_{new}U^iV^i)$. Finally, majority voting rule for the fusion is applied on the labels obtained in the previous step to obtain the final label vector.

Algorithm 5: MLC-HMF ($X, Y, d, \mathcal{T}, h$)

  input : Data Matrix: X, Label Matrix: Y, Size of Reduced Dimension Space: d,
           Threshold: $\mathcal{T}$, Depth of the Hierarchy: h

  output: Tree with Mapping U and Label Feature Matrix V at Each Node

  Divide X into X^1 and X^2 using $kmeans$ clustering;
  for $i \in \{1,2\}$ **do**
      if $|X^i|$ *is small or depth is exceed* h **then**
          Let its corresponding node as leaf node;
          return
      end
      Learn the mapping U and label feature matrix V for X^i using Eq. (5.19);
      Let $\bar{X} \subseteq X^i$ is the set of instances whose hamming loss is less than the
      threshold $\mathcal{T}$ and $\bar{Y}$ is their corresponding label matrix;
      Maintain U, V and $\bar{X}$ at the current node;
      MLC-HMF ($X^i \setminus \bar{X}, Y^i \setminus \bar{Y}, k, \mathcal{T}, h$)
  end

Complexity Analysis: We analyze the computational and spatial complexity of the proposed method. The time complexity of MLC-HMF mainly comprises of two components: clustering and optimization of the problem as given in Eq. (4) at every node of the hierarchy. For simplicity of representation, we are ignoring the correctly classified instances at the present node and assume that the set of instances are divided equally between its child. Hence, the average number of instances present in a node at level l is $N/2^{l-1}$. In every iteration of conjugate gradient, the computation cost required for calculation of gradients $\frac{\partial J}{\partial U}$ and $\frac{\partial J}{\partial V}$ is $\frac{2N}{2^{l-1}}D^2 d^2 L$. Let t_1 and t_2 be the maximum number of iterations required for gradient update and *k-means* clustering, respectively. Then the overall computation cost required at every node at level l is $(\frac{2N}{2^{l-1}}D^2 d^2 L t_1 + \frac{2N}{2^{l-1}}D t_2)$. Hence, the overall computation required at level l is $2^l(\frac{2N}{2^{l-1}}D^2 d^2 L t_1 + \frac{2N}{2^{l-1}}D t_2)$. The overall computation cost required by MLC-HMF is $2^l(h+1)(\frac{2N}{2^{l-1}}D^2 d^2 L t_1 + \frac{2N}{2^{l-1}}D t_2)$, that is, $\mathcal{O}(nD^2 d^2 L t_1 h)$. The space complexity of MLC-HMF mainly comprises of maintaining the mapping U, label feature matrix V and the set of instances whose hamming loss is less than the threshold $\mathcal{T}$ at every node. The overall space required by MLC-HMF is $(2^{h+1} - 1)(Dd + Ld) + ND$, that is, $\mathcal{O}(2^{h+1}(Dd + Ld))$.

5.6 Experimental Analysis

In this section we analyze the performance of MLC-HMF by taking into account factors such as accuracy and efficiency. This section discusses the experimental setup

including the data sets and relevant statistics, the experimental protocols, the competing algorithms, the evaluation metrics as well as the parameter settings. Following this, we discuss the experimental results.

5.6.1 Data Sets

We use twelve multi-label benchmark datasets for experiments, and the detailed characteristics of these datasets are summarized in Table 5.8. All of these datasets can be downloaded from *labic*[1], *meka*[2] and *mulan*[3].

Table 5.8: Description of the experimental datasets

Data set	#instance	#Feature	#Label	Domain	LC
CAL500	502	68	174	music	26.044
emotions	593	72	6	music	1.869
genbase	662	1185	27	biology	1.252
plant	948	440	12	biology	1.080
medical	978	1449	45	text	1.245
language log	1459	1004	75	text	1.180
human	3108	440	12	biology	1.190
education	5000	550	33	text(web)	1.461
science	5000	743	40	text(web)	1.451
rcv1(subset 2)	6000	944	101	text	2.634
rcv1(subset 5)	6000	944	101	text	2.642
ohsumed	13929	1002	23	text	1.663

5.6.2 Evaluation Metrics

To measure the performance of the different algorithms, we employed six evaluation metrics popularly used in multi-label classification, i.e. *hamming loss, accuracy, exact-match, example based f_1 measure, macro f_1 and micro f_1* (149; 105). Given a test data set $\mathcal{D} = \{x_i, y_i \mid 1 \leq i \leq N\}$, where $y_i \in \{-1, 1\}^L$ is the ground truth labels associated with the ith test example, and let $\hat{y}_i$ be its predicted set of labels.

---

[1] http://computer.njnu.edu.cn/Lab/LABIC/LABIC_Software.html
[2] http://meka.sourceforge.net
[3] http://mulan.sourceforge.net/datasets-mlc.html

Hamming loss measures how many times on average, an irrelevant pair (instance, label) is predicted, i.e. a correct label is missed or an incorrect label is predicted.

$$Hamming\ loss = \frac{1}{NL} \sum_{i=1}^{N} |y_i \neq \hat{y}_i|$$

Accuracy for an instance evaluates the proportion of correctly predicted labels to the total number of active(actual and predicted) labels for that instance. The overall accuracy for a data set is the average across all instances.

$$Accuracy = \frac{1}{N} \sum_{i=1}^{N} \frac{|y_i \wedge \hat{y}_i|}{|y_i \vee \hat{y}_i|}$$

Subset Accuracy is an extension of *accuracy* used in single label case to multi-label prediction. For an instance, the prediction is considered to be correct if all the predicted labels are the same as the ground truth labels for that instance. The overall subset-accuracy for a data set is the average across all instances.

$$Subset\ Accuracy = \frac{1}{N} \sum_{i=1}^{m} I(y_i = \hat{y}_i)$$

where, I is the indicator function.

Example based F_1 **Measure** is the harmonic mean of precision and recall for each example.

$$F_1 = \frac{1}{N} \sum_{i=1}^{N} \frac{2p_i r_i}{p_i + r_i}$$

where p_i and r_i are precision and recall for the ith example.

Macro F_1 is the harmonic mean of precision and recall for each label.

$$MacroF_1 = \frac{1}{L} \sum_{i=1}^{L} \frac{2p_i r_i}{p_i + r_i}$$

where p_i and r_i are precision and recall for the ith label.

Micro F_1 treats every entry of the label vector as an individual instance regardless of

label distinction.

$$Micro\ F_1 = \frac{2\sum_{i=1}^{L} TP_i}{2\sum_{i=1}^{L} TP_i + \sum_{i=1}^{L} FP_i + \sum_{i=1}^{L} FN_i}$$

where TP_i, FP_i and FN_i are true positive, false positive and false negative for the ith label, respectively.

5.6.3 Comparing Algorithms

We conducted *ten-fold cross validation* on each data set and the mean value and standard deviation of each evaluation criterion was recorded. The following six well-known state-of-the-art algorithms were considered for comparison:

- **BSVM** (9): This is one of the representative algorithm of problem transformation methods, which treat each label as a separate binary classification problem. For every label, an independent binary classifier is trained by considering the examples with the given class label as positive and others as negative.

- **BP-MLL** (147): In this method, a modified loss function and back-propagation are used to account for multi-label data. The number of hidden neurons is set to be 20% of the input dimensionality and the maximum number of training epochs is set to be 100 for all datasets (147; 145).

- **ML-kNN** (148): It is a multi-label lazy learning approach derived from the traditional k-Nearest Neighbor (kNN) algorithm. For each unseen instance, ML-kNN first identifies its k nearest neighbors in the training set. Based on statistical information gained from the label sets of these neighboring instances, maximum a posteriori (MAP) principle is utilized to determine the label set for the unseen instance. The number of nearest neighbors considered is set to be 10 for all datasets (148; 145).

- **LIFT** (145): LIFT constructs features specific to each label by conducting clustering analysis on its positive and negative instances. After that, for every label, a binary classifier is trained with label specific feature. The ratio parameter r is set to be 0.1 for all datasets (145; 43).

- **SSJDR-MLL** (137): This method aims to learn a linear transformation which can reduce the dimension of the original instance and at the same time preserve the inherent property. The reduced dimension is fixed to 100, and the balanced parameter γ is set to be 1.0 for all datasets (137).

- **LLSF** (43): This method addresses the inconsistency problem in multi-label classification by learning label specific features for the discrimination of each class label. The hyper-parameters α, β, γ and the threshold $\mathcal{T}$ are set to be 0.1, 0.1, 0.01, and 0.5 respectively for all datasets (43).

LIBSVM (15) is employed as the binary learner for classifier induction to instantiate BSVM, LIFT and SSJDR-MLL (43; 145).

5.6.4 Parameter Setting

Most of the MLC-HMF hyper-parameters were fixed. The number of neighbors K considered during the prediction stage were fixed to 5, the number of reduced latent dimension space k were fixed to $\lceil 0.5\mathcal{L} \rceil$. The termination conditions required to repeat the recursive process such as the depth of the hierarchy and the minimum number of instances in the node were set to 5. The remaining two hyper-parameters are tuned by conducting *ten-fold cross validation*. The regularization parameter λ for each data set is tuned from the candidate set $\{10^{\frac{i}{10}}\}, \forall i \in \{1, 5, \ldots, 20\}$ and the threshold $\mathcal{T}$ is tuned in the range $[0, 0.1]$ with step size 0.02.
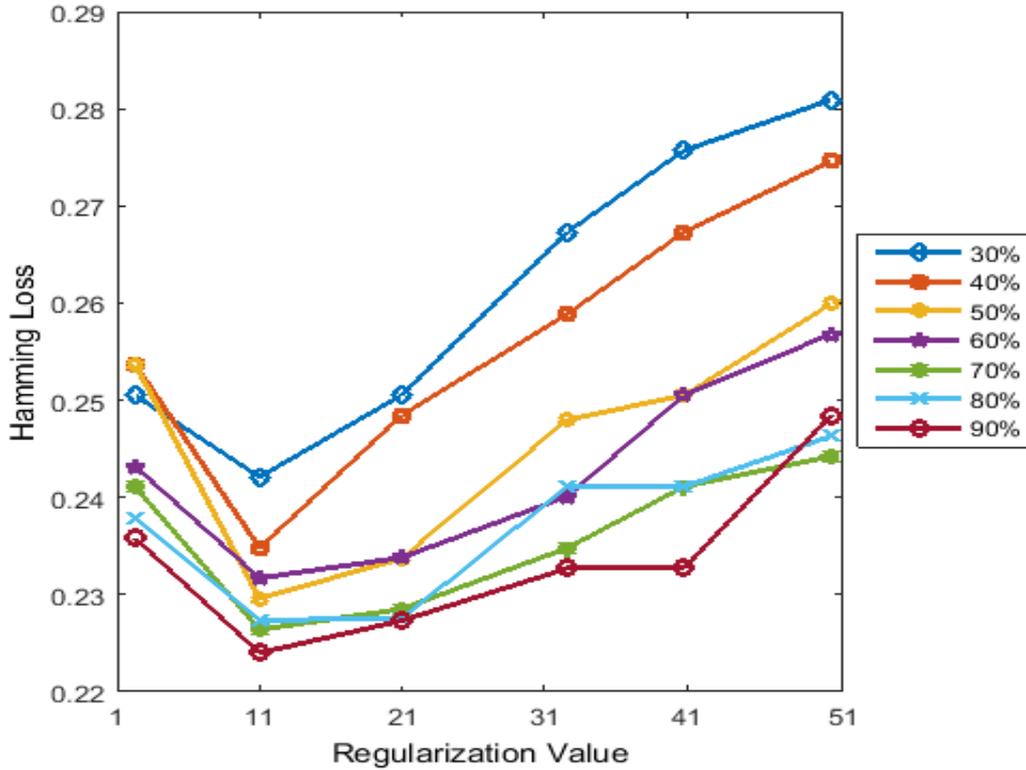
Figure 5.4: Hamming loss for different values of λ on emotions data set with varying training size percentage

In MLC-HMF, there is a need to select the regularization parameter λ at every node of the hierarchy as the size of training instances varies at every node of the hierarchy. We conducted experiments to analyze the effect of λ in Eq. (5.19) by varying the size of training data. For this experiment, we first randomly held out 10% of instances from

the overall data set which is later used to evaluate the performance. From the remaining instances we selected different percentage of instances to train the model. We repeated the selection process (train/ validation set) 3 times for every candidate λ and calculated the *hamming loss* on the held out set. Figure 5.4 depicts the variation in *hamming loss* for different values of λ. It can be seen from Figure 5.4 that the data size is not playing any crucial role in the selection of the best λ. Based on our preliminary experiments, the value of λ is fixed for every node in the hierarchy.

Table 5.9: Experimental results of comparison of algorithms (mean±std rank) in terms of Hamming Loss, Accuracy, and Subset Accuracy

Data set	Hamming Loss ↓													
	BSVM		BP-MLL		ML-kNN		LIFT		SSJDR-MLL		LLSF		MLC-HMF	
cal500	**0.137±0.006**	**1.5**	0.140±0.008	6	0.139±0.006	5	0.138±0.005	3.5	0.138±0.006	3.5	0.146±0.007	7	**0.137±0.005**	**1.5**
emotions	0.197±0.029	5	0.209±0.019	7	0.193±0.019	3	0.184±0.015	2	0.195±0.023	4	0.207±0.025	6	**0.182±0.026**	**1**
genbase	**0.002±0.001**	**2**	0.008±0.002	7	0.006±0.003	6	0.005±0.002	5	0.003±0.001	4	**0.002±0.001**	**2**	0.002±0.001	2
plant	0.090±0.002	5	0.092±0.005	6	0.088±0.003	4	0.085±0.004	2.5	**0.084±0.004**	**1**	0.123±0.006	7	0.085±0.005	2.5
medical	**0.011±0.001**	**1.5**	0.018±0.001	6	0.016±0.002	5	0.012±0.002	3	0.015±0.002	4	0.019±0.002	7	**0.011±0.001**	**1.5**
language log	**0.016±0.001**	**3**	0.020±0.001	6	**0.016±0.001**	**3**	**0.016±0.001**	**3**	**0.016±0.001**	**3**	0.047±0.004	7	**0.016±0.001**	**3**
human	0.083±0.002	4	0.085±0.002	5.5	0.082±0.002	3	**0.078±0.002**	**1.5**	**0.078±0.002**	**1.5**	0.086±0.003	6	0.085±0.004	5.5
education	**0.037±0.001**	**1**	0.053±0.002	7	0.039±0.002	5	0.038±0.002	3	0.040±0.001	6	0.038±0.001	3	0.038±0.002	3
science	**0.030±0.002**	**1**	0.043±0.002	7	0.033±0.001	5	0.031±0.001	2	0.034±0.001	6	0.032±0.001	3.5	0.032±0.002	3.5
rcv1(subset2)	**0.023±0.001**	**3**	0.026±0.001	7	0.024±0.001	6	**0.023±0.001**	**3**	**0.023±0.001**	**3**	**0.023±0.001**	**3**	**0.023±0.001**	**3**
rcv1(subset5)	**0.023±0.001**	**3**	0.025±0.001	7	0.024±0.001	6	**0.023±0.001**	**3**	**0.023±0.001**	**3**	**0.023±0.001**	**3**	**0.023±0.001**	**3**
ohsumed	0.058±0.002	3.5	0.078±0.001	7	0.071±0.001	6	0.056±0.001	2	0.067±0.001	5	**0.055±0.001**	**1**	0.058±0.001	3.5

Data set	Accuracy ↑													
	BSVM		BP-MLL		ML-kNN		LIFT		SSJDR-MLL		LLSF		MLC-HMF	
cal500	0.197±0.010	5	0.216±0.015	3	0.196±0.014	6	0.198±0.008	4	0.192±0.010	7	**0.228±0.013**	**1**	0.217±0.009	2
emotions	0.495±0.061	7	0.548±0.032	2.5	0.538±0.043	4	0.548±0.043	2.5	0.512±0.058	5	0.511±0.049	6	**0.550±0.060**	**1**
genbase	0.973±0.013	3	0.905±0.019	7	0.923±0.029	6	0.929±0.021	5	0.953±0.018	4	**0.981±0.007**	**1**	0.978±0.006	2
plant	0.037±0.014	6	0.012±0.031	7	0.079±0.021	5	0.150±0.028	4	0.163±0.033	3	**0.206±0.023**	**1**	0.173±0.033	2
medical	**0.732±0.014**	**1**	0.629±0.027	5	0.570±0.051	7	0.659±0.036	3	0.620±0.042	6	0.657±0.043	4	0.702±0.031	2
language log	0.062±0.018	5	**0.161±0.024**	**1**	0.017±0.009	7	0.109±0.024	3	0.030±0.014	6	0.095±0.024	4	0.121±0.025	2
human	0.076±0.011	6	0.000±0.001	7	0.094±0.013	5	0.201±0.012	3	**0.210±0.016**	**1**	0.193±0.020	4	0.207±0.016	2
education	0.270±0.019	4	**0.327±0.017**	**1**	0.225±0.015	6	0.255±0.016	5	0.158±0.011	7	0.293±0.017	3	0.294±0.021	2
science	0.280±0.018	3	**0.311±0.020**	**1**	0.198±0.013	6	0.280±0.010	3	0.136±0.013	7	0.264±0.015	5	0.280±0.018	3
rcv1(subset2)	0.325±0.017	2	0.324±0.011	3	0.188±0.015	7	0.272±0.014	5	0.205±0.016	6	0.281±0.013	4	**0.343±0.014**	**1**
rcv1(subset5)	0.324±0.013	2	0.321±0.014	3	0.192±0.017	7	0.285±0.015	4	0.219±0.013	6	0.275±0.013	5	**0.352±0.012**	**1**
ohsumed	0.293±0.014	5	**0.417±0.005**	**1**	0.053±0.011	7	0.337±0.013	4	0.131±0.007	6	0.356±0.013	2	0.344±0.011	3

Data set	Subset Accuracy ↑													
	BSVM		BP-MLL		ML-kNN		LIFT		SSJDR-MLL		LLSF		MLC-HMF	
cal500	**0.000±0.000**	**4**	**0.000±0.000**	**4**	**0.000±0.000**	**4**	**0.000±0.000**	**4**	**0.000±0.000**	**4**	**0.000±0.000**	**4**	**0.000±0.000**	**4**
emotions	0.271±0.074	5	0.259±0.056	7	0.292±0.054	3	**0.319±0.053**	**1**	0.290±0.068	4	0.265±0.050	6	0.305±0.077	2
genbase	0.961±0.015	3	0.837±0.021	7	0.891±0.043	6	0.897±0.031	5	0.936±0.024	4	**0.967±0.019**	**1**	0.962±0.012	2
plant	0.036±0.014	6	0.012±0.031	7	0.077±0.019	5	0.143±0.025	3	0.150±0.028	2	0.141±0.021	4	**0.162±0.034**	**1**
medical	**0.653±0.027**	**1**	0.464±0.032	7	0.499±0.055	5	0.586±0.044	3	0.528±0.064	4	0.498±0.052	6	0.630±0.043	2
language log	0.193±0.028	4	0.232±0.032	2	0.159±0.026	7	0.230±0.037	3	0.167±0.020	6	0.168±0.026	5	**0.233±0.030**	**1**
human	0.069±0.012	6	0.000±0.001	7	0.085±0.012	5	0.181±0.011	2	**0.190±0.015**	**1**	0.154±0.020	4	0.166±0.018	3
education	0.226±0.018	3	0.165±0.013	6	0.189±0.015	5	0.212±0.014	4	0.131±0.009	7	**0.240±0.015**	**1.5**	**0.240±0.020**	**1.5**
science	**0.237±0.018**	**1**	0.189±0.020	5	0.170±0.013	6	0.233±0.014	2	0.115±0.015	7	0.219±0.015	3	0.226±0.022	3
rcv1(subset2)	**0.207±0.015**	**1**	0.088±0.012	7	0.146±0.015	5	0.166±0.010	3	0.140±0.011	6	0.165±0.014	4	0.200±0.015	2
rcv1(subset5)	**0.201±0.011**	**1**	0.082±0.009	7	0.144±0.013	5	0.174±0.012	4	0.135±0.011	6	0.152±0.012	3	0.194±0.009	2
ohsumed	0.189±0.011	4	0.132±0.008	5	0.033±0.009	7	0.215±0.009	2	0.084±0.009	6	**0.222±.011**	**1**	0.209±0.011	3

5.6.5 Results and Discussion

Table 5.9 and 5.10 gives the comparative analysis of the proposed method MLC-HMF against state-of-the-art algorithms on twelve datasets. Each result is composed of $mean$, std and $rank$. For any data set and given evaluation metric where two or more

Table 5.10: Experimental results of comparison of algorithms (mean±std rank) in terms of Example Based F_1, Macro F_1, and Micro F_1

Example based F_1 ↑

Data set	BSVM		BP-MLL		ML-kNN		LIFT		SSJDR-MLL		LLSF		MLC-HMF	
cal500	0.325±0.014	5	0.350±0.020	2.5	0.323±0.020	6	0.326±0.011	4	0.316±0.013	7	**0.362±0.018**	1	0.350±0.013	2.5
emotions	0.565±0.059	7	**0.647±0.026**	1	0.620±0.041	4	0.624±0.043	3	0.583±0.058	6	0.591±0.052	5	0.629±0.057	2
genbase	0.976±0.013	3	0.924±0.021	7	0.933±0.025	6	0.938±0.018	5	0.958±0.017	4	**0.985±0.005**	1	0.983±0.006	2
plant	0.038±0.014	6	0.012±0.031	7	0.080±0.022	5	0.153±0.029	4	0.167±0.034	3	**0.230±0.026**	1	0.180±0.032	2
medical	**0.759±0.012**	1	0.686±0.028	4	0.595±0.051	7	0.684±0.036	5	0.653±0.037	6	0.715±0.041	3	0.728±0.028	2
language log	0.067±0.018	5	**0.188±0.025**	1	0.018±0.009	7	0.118±0.025	4	0.033±0.0140	6	0.132±0.026	3	0.133±0.026	2
human	0.079±0.011	6	0.001±0.001	7	0.097±0.013	5	0.214±0.013	3	0.217±0.016	2	0.206±0.021	4	**0.221±0.015**	1
education	0.286±0.019	4	**0.384±0.019**	1	0.238±0.015	6	0.271±0.018	5	0.167±0.012	7	0.312±0.017	3	0.314±0.022	2
science	0.295±0.019	4	**0.357±0.021**	1	0.207±0.013	6	0.298±0.012	3	0.143±0.013	7	0.279±0.016	5	0.300±0.017	2
rcv1(subset2)	0.370±0.019	3	**0.430±0.014**	1	0.204±0.015	7	0.313±0.016	5	0.230±0.018	6	0.326±0.012	4	0.400±0.015	2
rcv1(subset5)	0.374±0.014	3	**0.427±0.016**	1	0.210±0.019	7	0.331±0.017	4	0.254±0.013	6	0.326±0.014	5	0.415±0.014	2
ohsumed	0.331±0.015	5	**0.521±0.006**	1	0.060±0.013	7	0.382±0.015	4	0.149±0.007	6	0.404±.014	2	0.393±0.013	3

Macro F_1 ↑

Data set	BSVM		BP-MLL		ML-kNN		LIFT		SSJDR-MLL		LLSF		MLC-HMF	
cal500	0.046±0.003	6	0.053±0.008	3.5	0.053±0.007	3.5	0.049±0.003	7	0.041±0.004	5	**0.107±0.010**	1	0.079±0.006	2
emotions	0.584±0.059	7	0.626±0.029	3	0.622±0.033	5	**0.652±0.035**	1	0.623±0.046	4	0.617±0.048	6	0.649±0.057	2
genbase	0.649±0.062	2.5	0.534±0.047	6	0.520±0.026	7	0.584±0.052	5	0.631±0.061	4	**0.666±0.076**	1	0.649±0.053	2.5
plant	0.042±0.019	6	0.011±0.022	7	0.055±0.016	5	0.086±0.023	4	0.124±0.034	3	**0.164±0.031**	1	0.126±0.027	2
medical	0.320±0.030	2	0.245±0.020	5	0.206±0.026	7	0.267±0.032	4	0.223±0.022	6	**0.329±0.039**	1	0.285±0.025	3
language log	0.031±0.007	5	0.051±0.006	2	0.007±0.004	7	0.047±0.015	3.5	0.026±0.007	6	**0.074±0.014**	1	0.047±0.01	3.5
human	0.041±0.008	6	0.000±0.000	7	0.065±0.016	5	0.119±0.010	3	**0.140±0.010**	1	0.106±0.008	4	0.122±0.012	2
education	**0.140±0.016**	1	0.094±0.006	6	0.110±0.017	5	0.136±0.021	2	0.084±0.013	7	0.116±0.014	3	0.112±0.009	4
science	0.167±0.022	2	0.111±0.012	6	0.12±0.018	5	**0.182±0.019**	1	0.067±0.013	7	0.132±0.014	4	0.134±0.011	3
rcv1(subset2)	**0.198±0.014**	1	0.083±0.003	5	0.082±0.010	6	0.128±0.013	3	0.074±0.006	7	0.117±0.012	4	0.172±0.009	2
rcv1(subset5)	0.186±0.015	2	0.073±0.003	6.5	0.085±0.010	5	0.129±0.020	3	0.073±0.006	6.5	0.098±0.009	4	**0.188±0.016**	1
ohsumed	0.238±0.010	5	**0.436±0.012**	1	0.041±0.007	7	0.275±0.012	4	0.082±0.005	6	0.332±0.01	2	0.279±0.010	3

Micro F_1 ↑

Data set	BSVM		BP-MLL		ML-kNN		LIFT		SSJDR-MLL		LLSF		MLC-HMF	
cal500	0.321±0.015	5	0.347±0.019	3	0.320±0.020	6	0.323±0.011	4	0.311±0.012	7	**0.367±0.018**	1	0.348±0.012	2
emotions	0.636±0.055	7	0.670±0.028	3	0.667±0.043	4	0.679±0.033	2	0.655±0.046	5	0.639±0.049	6	**0.682±0.051**	1
genbase	0.981±0.009	2	0.917±0.016	7	0.930±0.031	6	0.947±0.018	5	0.966±0.013	4	**0.984±0.007**	1	0.978±0.009	3
plant	0.067±0.024	6	0.000±0.000	7	0.133±0.033	5	0.234±0.043	4	0.258±0.051	3	**0.279±0.023**	1	0.267±0.042	2
medical	**0.804±0.020**	1	0.706±0.022	5	0.676±0.040	7	0.758±0.040	3	0.713±0.029	4	0.701±0.028	6	0.775±0.025	2
language log	0.122±0.025	5	**0.264±0.029**	1	0.000±0.000	7	0.199±0.041	3	0.063±0.024	6	0.157±0.022	4	0.215±0.031	2
human	0.126±0.016	6	0.000±0.001	7	0.152±0.018	5	0.299±0.017	2	0.298±0.018	3	0.283±0.023	4	**0.302±0.024**	1
education	0.377±0.024	4	**0.400±0.019**	1.5	0.330±0.022	6	0.362±0.022	5	0.260±0.018	7	0.397±0.023	3	**0.400±0.026**	1.5
science	0.376±0.022	2	0.368±0.021	4	0.286±0.022	6	**0.377±0.016**	1	0.199± 0.019	7	0.357±0.017	5	0.370±0.020	3
rcv1(subset2)	0.403±0.021	3	0.419±0.011	2	0.242±0.019	7	0.347±0.016	5	0.268±0.020	6	0.361±0.013	4	**0.421±0.013**	1
rcv1(subset5)	0.413±0.012	3	0.430±0.012	2	0.253±0.023	7	0.375±0.016	4	0.298±0.016	6	0.366±0.014	5	**0.436±0.012**	1
ohsumed	0.407±0.014	5	**0.539±0.007**	1	0.088±0.017	7	0.456±0.012	4	0.204±0.009	6	0.484±.012	2	0.463±0.011	3

algorithms obtain the same performance, the rank of these algorithms are assigned with the average result of them. For each evaluation criterion, ↑ (↓) indicates the larger (smaller) the value, the better the performance. Furthermore, the best results among all comparing algorithms are highlighted in boldface.

Table 5.11: Summary of the Friedman statistics $F_F(\mathcal{K} = 7, \mathcal{N} = 12)$ and the critical value in terms of each evaluation metric($\mathcal{K}$: # Comparing Algorithms; $\mathcal{N}$: # Data Sets).

Metric	F_F	Critical Value ($\alpha = 0.05$)
Hamming Loss	5.730	
Accuracy	4.910	
Subset Accuracy	5.584	2.239
Example Base F_1	5.338	
Macro F_1	2.767	
Micro F_1	4.958	

To conduct statistical performance analysis among the algorithms being compared, we employed Friedman test which is a favorable statistical test for comparisons of more than two algorithms over multiple datasets (24). Table 5.11 provides the Friedman statistics F_F and the corresponding critical value in terms of each evaluation metric. As shown in Table 5.11 at significance level $\alpha = 0.05$, Friedman test rejects the null hypothesis of equal performance for each evaluation metrics. This leads to the use of post-hoc tests for pairwise comparisons. The Nemenyi test (24) is employed to test whether our proposed method MLC-HMF achieves a competitive performance against the comparing algorithms. The performance of two classifiers is significantly different if the corresponding average ranks differ by at least the critical difference $CD = q_\alpha \sqrt{\frac{\mathcal{K}(\mathcal{K}+1)}{6\mathcal{N}}}$. At significance level $\alpha = 0.05$, the value of $q_\alpha = 2.949$, for Nemenyi test with $\mathcal{K} = 7$ (24), and thus $CD = 2.601$. Figure 5.5 gives the CD diagrams (24) for each evaluation criterion, where the average rank of each comparing algorithm is marked along the axis (lower ranks to the left). It can be seen from the Figure 5.5 that the proposed method MLC-HMF achieve better performance than other comparing algorithms in terms of each evaluation metric.

5.7   Conclusions and Discussion

This chapter presented a new multi-label classification method, called MLC-HMF, which learns piecewise-linear embedding with a low-rank constraint on parametrization to capture nonlinear intrinsic relationships that exist in the original feature and label space. Extensive comparative studies validate the effectiveness of MLC-HMF against the state-of-the-art multi-label learning approaches.

In multi-label classification problem, infrequently occurring (tail) labels are associated with few training data and are harder to predict than frequently occurring ones but might also be more informative and rewarding. Due to the presence of tail labels the low-rank label matrix assumption fails to hold in embedding methods in real-world applications. We feel that our proposal of hierarchical embedding may pave way to overcome this difficulty and can handle tail labels more efficiently. MLC-HMF is the first ever attempt for piece-wise linear embedding in the context of multi-label learning. It is worthwhile to carryout an in depth investigation of different ways of embedding

(a) Hamming loss (b) Accuracy

(c) Subset Accuracy (d) Example based F_1

(e) Macro F_1 (f) Micro F_1
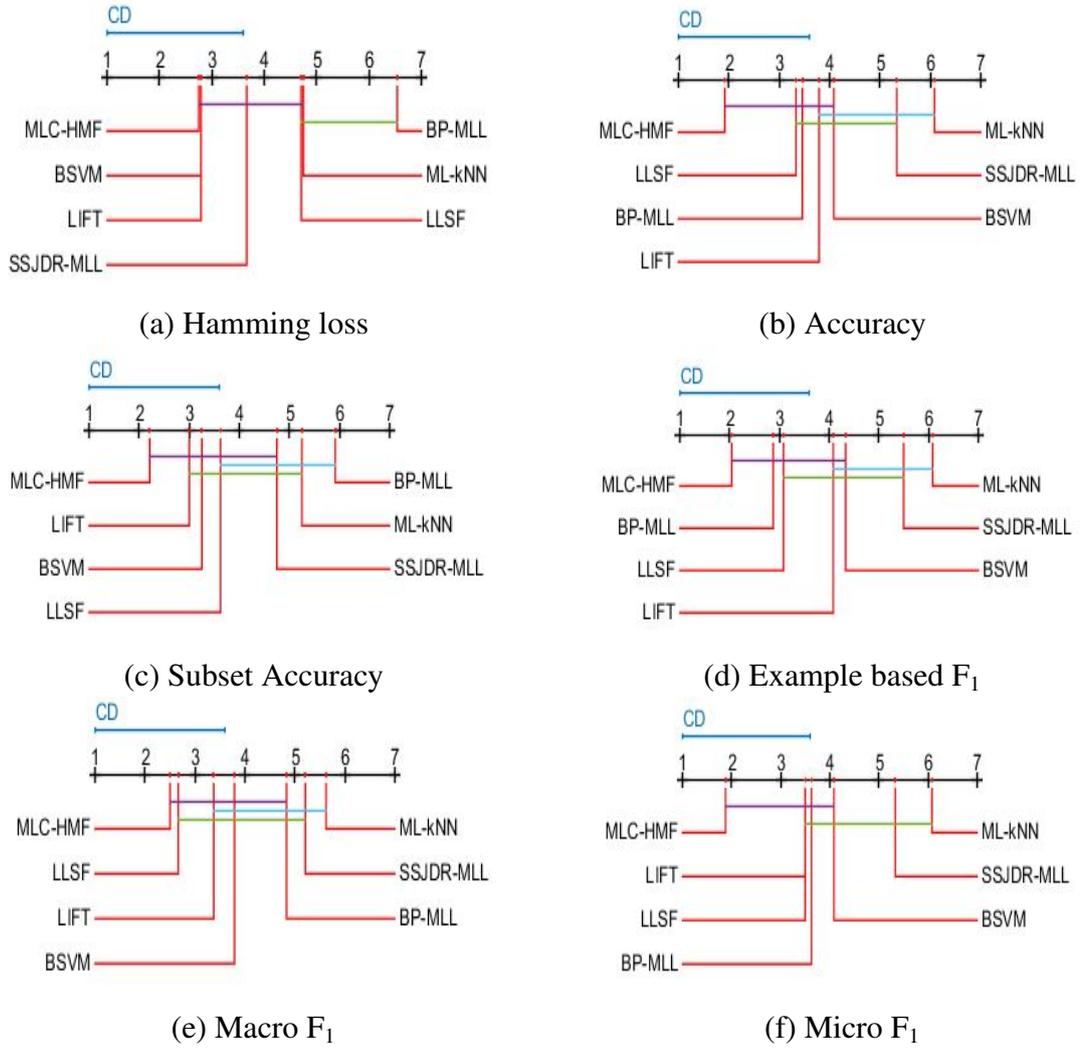
Figure 5.5: CD diagrams of the comparing algorithms under each evaluation criterion.

such as quasi-linear embeddings and their advantages over linear and non-linear embedding. We plan to carryout this line of investigation in future. Our experimental analysis provides evidence that hierarchical embedding is able to yield more accurate results for multi-label classification. This motivates a new line of future research in which computational complexity of the proposed algorithm can be investigated. The core part of the algorithm is a search based on gradient descent. It is worthwhile to investigate different algorithmic strategies of gradient descent to improve the computational efficiency.

CHAPTER 6

Group Preserving Label Embedding for Multi-Label Classification

In the previous chapter, we discussed a joint learning framework called MLC-HMF in which the feature space embedding and multi-label classification are performed simultaneously. The MLC-HMF approach is motivated by the fact that the low-rank assumption made by the embedding methods is violated in most of the real-world applications due to several reasons. Reasons such as the diversity of the training set, the correlation among labels, the feature-label relationship, the mapping to be determined by the learning algorithm and most importantly, the presence of *tail* labels to mention a few. However, on the other hand, it is possible to have a fraction of training instances share the same subset of label correlations. The underlying principle of MLC-HMF is to learn a piecewise-linear embedding of feature-space that generates optimal embeddings for a subset of training instances. As discussed in Chapter 5, there are two different approach of embedding namely, feature space embedding and label space embedding. In this chapter, we propose a novel label embedding based approach for multi-label classification.

6.1   Introduction

In this chapter, we discuss a novel label embedding approach for multi-label classification inspired by the extensive applications of data mining algorithms, where features

or data items are inherently organized into groups. In the context of multi-label classification, there are few proposals which model group information but no detailed study in this direction has so far been undertaken. To exploit label-correlations in the data locally, it is assumed in (46) that the training data are in groups with instances in the same group sharing the same label correlations. In (108), highly correlated labels are grouped together using the information from label and instance spaces and for every group, sparse meta-label-specific features are learnt.

The present research starts with the assumption that there exists a low-rank space onto which the given set of feature vectors and label vectors can be embedded. Feature vectors can be embedded as points and label vectors correspond to linear predictors, as decision hyperplanes, in this embedded space. There are similarities among labels belonging to the same group such that their low-rank representations share the same sparsity pattern. For example, the set of labels in *corel5k* data set can be grouped into *landscape-nature*, *humans*, *food* etc. (26). The features such as eye and *leg* are specific to the *humans* whereas the feature like *ridge* is specific to the group *landscape-nature*. Given the label matrix Y, it is necessary to find points in a space of reduced dimension and to determine decision hyperplanes such that the classification thereof is compatible with Y. While doing so, it is desirable that the process must retain group information of labels. We achieve this by a matrix factorization based framework in which the label matrix Y is approximated using the product of two matrices $U \in \mathbb{R}^{N \times d}$ and $V \in \mathbb{R}^{d \times L}$. In a sense the row of matrix U can be viewed as point in a reduced dimension and the column of V defines a set of decision hyperplanes. If there are any dependency properties in labels of Y (column of Y), this is retained in dependencies in decision hyperplanes (column of V) and not in embedded points. We use the $\ell_{2,1}$ norm regularization on V to exploit the shared sparsity pattern among the label groups. The second sub-objective is to learn a linear mapping that maps the feature vectors onto the same set of points which are obtained as a result of factorization of label matrix. We achieve this by a separate optimization problem. We make use of correlation coefficients to capture the similarity relation and ℓ_1 norm for regularization. We use FISTA (4) type of method to learn the label embedding and subsequently mapping from feature space to the embedded label space. Thus, in this chapter, we develop a novel multi-label classification method. To the best of our knowledge, there has not been any earlier attempt in this direction. We feel that this approach will eventually provide a robust classification

technique as demonstrated by our experimental results which looks very promising.

The rest of the chapter is organized as follows. We introduce our proposed method, termed as GroPLE *(Group Preserving Label Embedding for Multi-Label Classification)* in Section 6.2. Experimental analysis of the proposed method is reported in Section 6.3. Finally, we conclude with Section 6.4 and indicates several issues for future work.

6.2 GroPLE: The Proposed Method

In this section, a novel method of multi-label classification is proposed. The proposed method GroPLE has three major stages namely (1) Identification of groups of labels; (2) Embedding of label vectors to a low rank-space so that the sparsity characteristic of individual groups remains invariant; and (3) Determining a linear mapping that embeds the feature vectors onto the same set of points, as in stage 2, in the low-rank space.

Identification of Groups of Labels: The label groups are not given explicitly and it is necessary to learn from the label matrix Y. One approach is to cluster the columns of Y. There are several clustering algorithms proposed in the literature such as *k-means* (48; 47), *hierarchical clustering* (54) and *spectral clustering* (139; 89; 120). We adopt *spectral clustering*. We do not claim that *spectral clustering* is the best option. We first construct a graph $G = <\mathcal{V}, E>$ in the label space, where $\mathcal{V}$ denotes the vertex/ label set, and E is the edge set containing edges between each label pair. We adopt heat kernel weight with self-tuning technique (for parameter σ) as edge weight if two labels are connected $A_{i,j} = \exp\left(\frac{(-\|Y_{\cdot i} - Y_{\cdot j}\|^2)}{\sigma}\right)$ where $Y_{\cdot i}$ and $Y_{\cdot j}$ are the ith and jth column of matrix Y (139). Labels can be grouped into K clusters by performing *k-means* with K largest eigenvectors as seeds of the normalized affinity matrix $L = D^{-\frac{1}{2}} A D^{-\frac{1}{2}}$, where D is a diagonal matrix with $D_{i,i} = \sum_j A_{i,j}$.

Label Space Embedding: Given a label matrix Y, each column corresponds to a label and our assumption is that related labels form groups. Let the columns of $Y \in \{-1, 1\}^{N \times L}$ be divided into K groups as $Y = (Y^1, \ldots, Y^K)$, where $Y^k \in \{-1, 1\}^{N \times L_k}$ and $\sum_k^K L_k = L$. Matrix factorization based approach of label embedding as discussed in Section 5.3 aims to find latent factor matrices $U \in \mathbb{R}^{N \times d}$ and $V \in \mathbb{R}^{d \times L}$ to approximate Y. In the present case, where labels are divided into groups, we approximate

Y^k using U and $V^k \in \mathbb{R}^{d \times L_k}$. Ideally, there should be a subset of columns of U associated with any group and hence, the corresponding vector in V^k of a label should have nonzero values only for the entries which correspond to the subset of columns of U associated with the group. More concretely, we expect that for deciding any label groupss all the features are not important and each label in that group can be decided by linear combination of fewer group features. To achieve this, we add a $\ell_{2,1}$-norm regularization on V^k that encourages row sparsity of V^k. Then the sub-objective to learn the embedding from original label space is given by

$$\min_{U,V^1,\ldots,V^K} f(U, V^1, \ldots, V^K) = \sum_{k=1}^{K} \|Y^k - UV^k\|_F^2 + \lambda_1 \|U\|_F^2 + \lambda_2 \sum_{k=1}^{K} \|V^k\|_{2,1} \quad (6.1)$$

where for a given matrix $A \in \mathbb{R}^{n \times m}$, $\|A\|_F^2 = \sum_{i=1}^{n} \sum_{j=1}^{m} A_{ij}^2$ and $\|A\|_{2,1} = \sum_{i=1}^{n} \sqrt{\sum_{j=1}^{m} A_{ij}^2}$.

We can solve Eq. (6.1) by alternating minimization scheme that iteratively optimizes each of the factor matrices keeping the other fixed. For simplicity of notation, the matrix formed by arranging the columns of V^k, $1 \le k \le K$, according to indices of columns in Y will be referred to as V in subsequent discussions. $f(U, V)$ is written as $f_V(U)$ when V is held constant and $f_U(V)$ when U is held constant. For given V, the factor matrix U can be obtained by solving the following subproblem.

$$\min_U f_V(U) = \|Y - UV\|_F^2 + \lambda_1 \|U\|_F^2 + c \quad (6.2)$$

where $c \ge 0$ is a constant. The subproblem given in Eq. (6.2) has a closed form solution. Taking the derivative of $f_V(U)$ w.r.t U, and setting the derivative (in matrix notation) to zero, we have

$$\nabla f_V(U) = 2((Y - UV)(-V^T) + \lambda_1 U) = 0$$
$$\Rightarrow U = YV^T(VV^T + \lambda_1 I)^{-1} \quad (6.3)$$

For fixed U, the matrix V^k, $k \in \{1, \ldots, K\}$, can be obtained by solving the following subproblem

$$\min_{V^k} f_U(V^k) = \|Y^k - UV^k\|_F^2 + \lambda_2 \|V^k\|_{2,1} + c \quad (6.4)$$

The above objective function is a composite convex function involving the sum of a

smooth and a non-smooth function of the form

$$\min_{V^k} f_U(V^k) = g(V^k) + h(V^k) \tag{6.5}$$

where $g(V^k) = \|Y^k - UV^k\|_F^2$ is convex and differentiable and $h(V^k) = \lambda_2 \|V^k\|_{2,1}$ is closed, convex but non-differentiable.

We further show that for any two matrices $V'^k, V''^k \in \mathbb{R}^{d \times L_k}$, the function $g(V^k)$ is Lipschitz continuous. The gradient of $g(V^k)$ (in matrix notation) is given by

$$\nabla g(V^k) = 2(U^T U V^k - U^T Y)$$

For any two matrices V'^k and V''^k, we have

$$\|\nabla g(V'^k) - \nabla g(V''^k)\|_F^2 = \|2(U^T U V'^k - U^T Y) - 2(U^T U V''^k - U^T Y)\|_F^2$$
$$= \|2U^T U(V'^k - V''^k)\|_F^2$$
$$\leq \|2U^T U\|_F^2 \|V'^k - V''^k\|_F^2$$

Therefore, the Lipschitz constant is

$$L_g = \sqrt{\|2U^T U\|_F^2} \tag{6.6}$$

We employ Accelerated Proximal Gradient search (112) which is specifically tailored to minimize the optimization problem given in Eq. (6.5). Such an optimization strategy is suitable in the present situation as the computation of the proximal operation is inexpensive. The optimization step of Accelerated Proximal Gradient iterates as follows.

$$G_t = V^{k(t)} + \frac{b_{t-1} - 1}{b_t}(V^{k(t)} - V^{k(t-1)}) \tag{6.7}$$

$$V^{k(t)} = prox_h(G_t - \frac{1}{L_g}\nabla g(G_t)) \tag{6.8}$$

It is shown in (112) that setting b_t satisfying $b_t^2 - b_t \leq b_{t-1}^2$ can improve convergence rate to $O(\frac{1}{t^2})$. $V^{k(t)}$ is the result of tth iteration. Proximal mapping of a convex function

99

h is given by

$$prox_h(V^k) = \underset{W}{argmin} \left(h(W) + \frac{1}{2}\|W - V^k\|_2^2 \right) \qquad (6.9)$$

In the present situation, where $h(V^k) = \lambda_2\|V^k\|_{2,1}$, $prox_h(V^k)$ is the shrinkage function $S[\cdot]$ and is given by

$$S_{\frac{\lambda_2}{L_g}}[V^k] = \left[\frac{v_i^k}{\|v_i^k\|_2}(\|v_i^k\|_2 - \lambda_2/L_g)_+ \right]_{i=1}^{i=d} \qquad (6.10)$$

where $(z)_+ = max(z, 0)$ and v_i^k is the ith row of V^k. Algorithm 6 outlines the main flow of the optimization steps to solve Eq. (6.1).

Algorithm 6: Label-Embedding (Y, d, K, λ_1, λ_2)

---

    input : Label Matrix: Y, Size of Latent Dimension Space: d, Number of
              Groups: K, Regularization Parameters: λ_1 and λ_2
    output: Basis Matrix: U, Coefficient Matrix: V

    initialize : U
    Form label groups Y^1, Y^2, …, Y^K
    repeat
        for $k \in \{1,…, K\}$ **do**
            $V^k \leftarrow$ APG(U, Y^k, λ_2)
        end
        $V \leftarrow Combine(V^1, V^2, …, V^k)$
        $U \leftarrow YV^T(VV^T + \lambda_1 I)^{-1}$
    until *stop criterion reached*;

Algorithm 7: APG (U, Y^k, λ_2)

---

    input : Basic Matrix: U, Label Matrix: Y^k and Regularization Parameters: λ_2
    output: Coefficient Matrix: V^k

    initialize :
    $b_0, b_1 \leftarrow 1, V_0^k, V_1^k \leftarrow (U^TU + \gamma I)^{-1}U^TY^k$
    repeat
        $G_t \leftarrow V^{k(t)} + \frac{b_{-1}-1}{b_t}(V^{k(t)} - V^{k(t-1)})$
        $V^{k(t)} = S_{\frac{\lambda_2}{L_g}}[G_t - \frac{1}{L_g}\nabla g(G_t)]$
        $b_t \leftarrow \frac{1+\sqrt{1+4b_t^2}}{2}$
        $t \leftarrow t + 1$
    until *stop criterion reached*;
    $V^k \leftarrow V_t^k$

Feature Space Embedding: The U matrix computed above as a result of the learning process described in Algorithm 6 represents a set of points and it is desired that these

points, in some sense, represent the training objects. We assume that there exists a linear embedding $Z \in \mathbb{R}^{D \times d}$ that maps the feature matrix X to U. Thus, we justify our hypothesis that there exists a low-rank space where both X and Y are embedded and this embedding retains the intrinsic feature-label relation as well as the group information. In order to achieve the embedding of feature vectors, we try to capture the correlation among the embedded representation of Y and formulate the objective function as follows.

$$\min_{Z} \|XZ - U\|_F^2 + \alpha \sum_{j=1}^{d} R_{ij} Z_i^T Z_j + \beta \|Z\|_1 \qquad (6.11)$$

where Z_i is the ith column of matrix Z and $R_{ij} = 1 - C_{ij}$, where C_{ij} represent the correlation coefficient between ith and jth column of matrix U. The above objective function is of the form defined in Eq. (6.5) where $g(Z) = \|XZ - U\|_F^2 + \alpha \operatorname{Tr}(RZ^T Z)$ is convex and Lipschitz continuous and $h(Z) = \beta \|Z\|_1$ is closed, convex but non-differentiable. The Accelerated Proximal Gradient technique described previously is used to solve Eq. (6.11). The optimization step of Accelerated Proximal Gradient initializes $Z_0 = Z_1 = (X^T X + \gamma I)^{-1}$ and iterates as follows.

$$G_t = Z^{(t)} + \frac{b_{t-1} - 1}{b_t}(Z^{(t)} - Z^{(t-1)}) \qquad (6.12)$$

$$Z^{(t)} = prox_h(G_t - \frac{1}{L_g} \nabla g(G_t)) \qquad (6.13)$$

where $Z^{(t)}$ is the result of tth iteration, the Lipschitz constant L_g and $prox_h(Z)$ is given below. The gradient of $g(Z)$ (in matrix notation) is given by

$$\nabla g(Z) = 2X^T(XZ - U) + \alpha ZR$$

For any two matrices $Z', Z'' \in \mathbb{R}^{D \times d}$, we have

$$
\begin{aligned}
\|\nabla g(Z') - \nabla g(Z'')\|_F^2 &= \|2(X^T X Z' - X^T U) + \alpha Z' R - 2(X^T X Z'' - X^T U) - \alpha Z'' R\|_F^2 \\
&= \|2X^T X(Z' - Z'') + \alpha(Z' - Z'')R\|_F^2 \\
&\leq \|2X^T X\|_F^2 \|Z' - Z''\|_F^2 + \|\alpha R\|_F^2 \|Z' - Z''\|_F^2 \\
&= \|2X^T X + \alpha R\|_F^2 \|Z' - Z''\|_F^2
\end{aligned}
$$

Therefore, the Lipschitz constant is

$$L_g = \sqrt{\|2X^T X + \alpha R\|_F^2} \tag{6.14}$$

The proximal mapping of $h(Z) = \beta\|Z\|_1$ is a shrinkage function $S[\cdot]$ and is given by

$$prox_h(z) = S_{\frac{\beta}{L_g}}[Z] = \begin{cases} Z_{ij} - \beta/L_g, & Z_{ij} > \beta/L_g \\ 0, & -\beta/L_g \le Z_{ij} \le \beta/L_g \\ Z_{ij} + \beta/L_g, & Z_{ij} < \beta/L_g \end{cases} \tag{6.15}$$

where, $1 \le i \le d$ and $1 \le j \le k$.

Complexity Analysis: We analyze the computational complexity of the proposed method. The time complexity of GroPLE mainly comprises of three components: formation of label groups and the optimization of the problem given in Eq. (6.1) and (6.11). The formation of label groups has two parts: construction of neighbourhood graph and spectral decomposition of a graph Laplacian. This part takes $O(NL^2 + L^3)$. For each iteration in Algorithm 6, updating U requires $O(NLd + d^3 + Nd^2)$. For simplicity of representation, we are ignoring the number of groups K and using the total number of labels L. Hence, the updation of V takes $O(NLd + d^3 + Nd^2 + Ld^2)$. Let t_1 be the maximum number of iterations required for gradient update, then overall computation required in LE process is $O(NL^2 + L^3) + O(t_1(NLd + d^3 + Nd^2)) + O(t_1(NLd + d^3 + Nd^2 + Ld^2))$, that is, $O(t_1(NLd + d^3 + Nd^2 + Ld^2))$. Similarly, the complexity of feature space embedding is $O(ND^2 + D^3) + O(2NDd + Dd^2)$, that is, $O(t_2(2NDd + Dd^2))$, where t_2 is the number of iteration. Hence the overall computation required by GroPLE is $O(t_1(NLd + d^3 + Nd^2 + Ld^2) + t_2(2NDd + Dd^2))$.

6.3 Experimental Analysis

To validate the proposed GroPLE, we perform experiments on twelve commonly used multi-label benchmark data sets. The detailed characteristics of these data sets are summarized in Table 6.1. All the data sets are publicly available and can be downloaded

from *meka*[1] and *mulan*[2].

Table 6.1: Description of the experimental datasets.

Data set	#instance	#Feature	#Label	Domain	LC
genbase	662	1185	27	biology	1.252
medical	978	1449	45	text	1.245
CAL500	502	68	174	music	26.044
corel5k	5000	499	374	image	3.522
rcv1 (subset 1)	6000	944	101	text	2.880
rcv1 (subset 2)	6000	944	101	text	2.634
rcv1 (subset 3)	6000	944	101	text	2.614
bibtex	7395	1836	159	text	2.402
corle16k001	13766	500	153	image	2.859
delicious	16105	500	983	text(web)	19.020
mediamill	43907	120	101	video	4.376
bookmarks	87856	2150	208	text	2.028

6.3.1 Evaluation Metrics

To measure the performance of different algorithms, we have employed four evaluation metrics popularly used in multi-label classification, i.e. *accuracy, example based f_1 measure, macro f_1* and *micro f_1* (149; 105). Given a test data set $\mathcal{D} = \{x_i, y_i \mid 1 \leq i \leq N\}$, where $y_i \in \{-1, 1\}^L$ is the ground truth labels associated with the ith test example, and $\hat{y}_i$ be its predicted set of labels. The detailed discussion of these evaluation metrics is given in Section 5.6.

6.3.2 Baseline Methods

For performance comparison, we consider seven well-known state-of-the-art algorithms and these are the following.

- **BSVM** (9): This is one of the representative algorithms of problem transformation methods, which treat each label as a separate binary classification problem. For every label, an independent binary classifier is trained by considering the examples with the given class label as positive and others as negative. LIBSVM (15) is employed as the binary learner for classifier induction to instantiate BSVM.

---

[1] http://meka.sourceforge.net
[2] http://mulan.sourceforge.net/datasets-mlc.html

- **PLST** (109): Principal label space transformation (PLST) uses singular value decomposition (SVD) to project the original label space into a low dimensional label space.

- **CPLST** (16): CPLST is a feature-aware conditional principal label space transformation which utilizes the feature information during label embedding.

- **CSSP** (6): Using randomized sampling procedure, CSSP sample a small subset of class labels that can approximately span the original label space. Once this subset of labels are selected, a binary classifier is trained for each of them.

- **FAiE** (76): FAiE encodes the original label space to a low-dimensional latent space via feature-aware implicit label space encoding. It directly learns a feature-aware code matrix and a linear decoding matrix via jointly maximizing recoverability of the original label space.

- **LEML** (136): In this method a framework is developed to model multi-label classification as generic empirical risk minimization (ERM) problem with low-rank constraint on linear transformation. It can also be seen as a joint learning framework in which dimensionality reduction and multi-label classification are performed simultaneously.

- **MLSF** (108): Based on the assumption that meta-labels with specific features exist in the scenario of multi-label classification, MLSF embed label correlations into meta-labels in such a way that the member labels in a meta-label share strong dependency with each other but have weak dependency with the other non-member labels.

A linear ridge regression model is used in PLST, CPLST, CSSP and FAiE to learn the association between feature space and reduced label space. For PLST, CPLST, CSSP, FAiE and LEML the number of reduced dimensions d is searched in $\{\lceil 0.1L \rceil,$ $\lceil 0.2L \rceil, \ldots, \lceil 0.8L \rceil\}$. The regularization parameter in ridge regression, the parameter α in FAiE and the parameter λ in LEML are searched in the range $\{10^{-4}, 10^{-3}, \ldots, 10^4\}$. For MLSF, the number of meta-labels K is searched in $\{\lceil L/5 \rceil, \lceil L/10 \rceil, \lceil L/15 \rceil, \lceil L/20 \rceil\}$ and the parameters γ and ρ are tuned from the candidate set $\{10^{-4}, 10^{-3}, \ldots, 10^4\}$. The remaining hyper-parameters were kept fixed across all datasets as was done in (108). Implementations of PLST, CPLST, CSSP, FAiE, LEML, MLSF were provided by the authors.

6.3.3 Results and Discussion

We first demonstrate the effect of group sparsity regularization on the recovered matrix V on *Medical* data set. The regularization parameter λ_1 and λ_2 in Eq. (6.1) are

selected using Cross-validation. The label matrix Y is divided into five groups using the method described in Section 6.2. The recovered feature matrix V^k for each group is
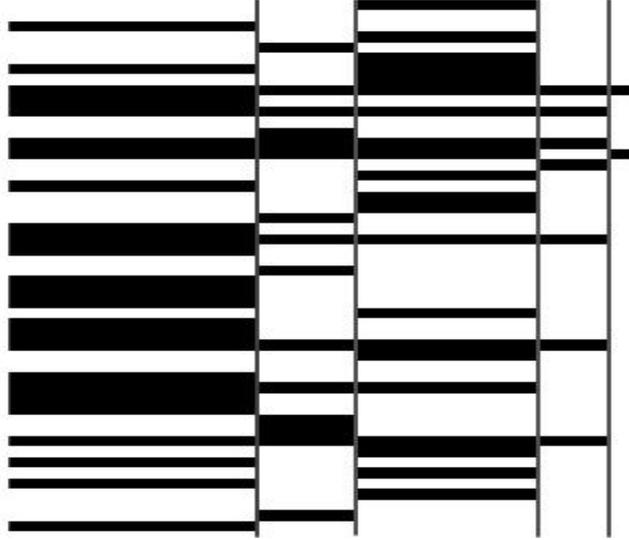

Figure 6.1: Latent factor matrix V^k recovered with five label groups.

depicted in Figure 6.1. The shaded color represents the non-zero rows in the recovered matrix V^k and a separation line is artificially drawn to distinguish between the latent factor matrix of the different groups. It is evident from Figure 6.1 that the feature matrix recovered for different groups exhibits a sparsity pattern and the labels corresponding to objects in the same group have similar sparsity.

To study the sensitivity of GroPLE with respect to the regularization parameters λ_1 and λ_2, we conducted experiments on *Medical* and *Genbase* data sets. We perform *five-fold cross validation* on each data set and the mean value of accuracy is recorded. In this experiment, the latent dimension space d is fixed to 100, the number of groups K is fixed to 5 and the regularization parameters λ_1 and λ_2 are searched in $\{10^{-3}, 10^{-2}, \dots, 10^2\}$. For each (λ_1, λ_2) -pair, the regularization parameters α, β are searched in $\{10^{-4}, 10^{-3}, \dots, 10^4\}$. Figure 6.2 report the influence of parameters λ_1 and λ_2 on *Medical* and *Genbase* data set. It can be seen that in most cases: (a) GroPLE perform worse when the value of λ_1 is large; (b) The performance of GroPLE is stable with the different value of group sparsity regularization λ_2, but the larger value such as $\lambda_2 \geq 1$ is often harmful. Therefore, we fixed the regularization parameter λ_1 and λ_2 to 0.001 and 1, respectively, for the subsequent experiments.

We have also analyzed the performance of GroPLE with respect to the latent dimension d and the number of groups K on *rcv1 (subset 1)* data set. We have conducted
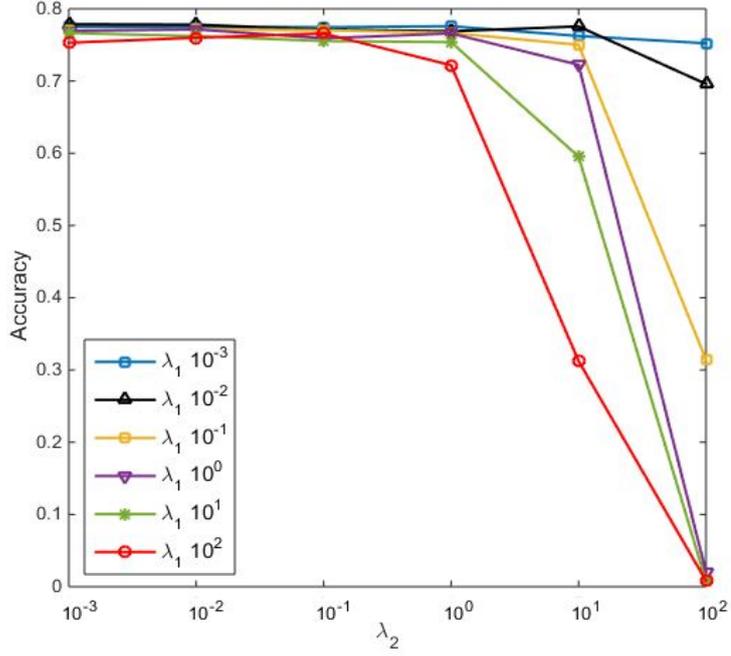
(a) Medical dataset
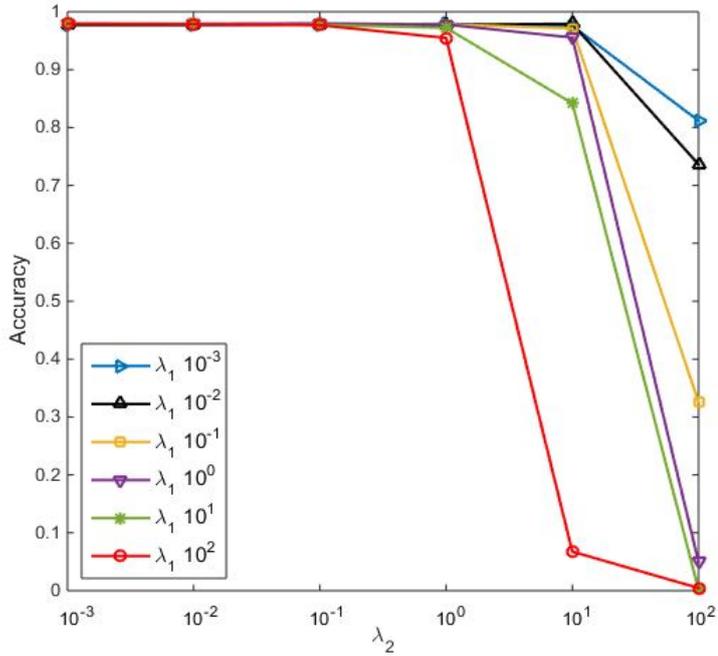
(b) Genbase dataset

Figure 6.2: Influence of regularization parameters λ_1 and λ_2.

five-fold cross validation and the mean value of accuracy is recorded. The latent dimension d is varied from $[20, 100]$ with step size 20 and the number of groups K is selected from $\{1, 10, 20, 30, 40\}$. The regularization parameters α and β are tuned in the range given previously. The plots of Figure 6.3 shows the classification performance of GroPLE in terms of *accuracy* and *Macro F_1*. It can be seen from Figure 6.3 that the
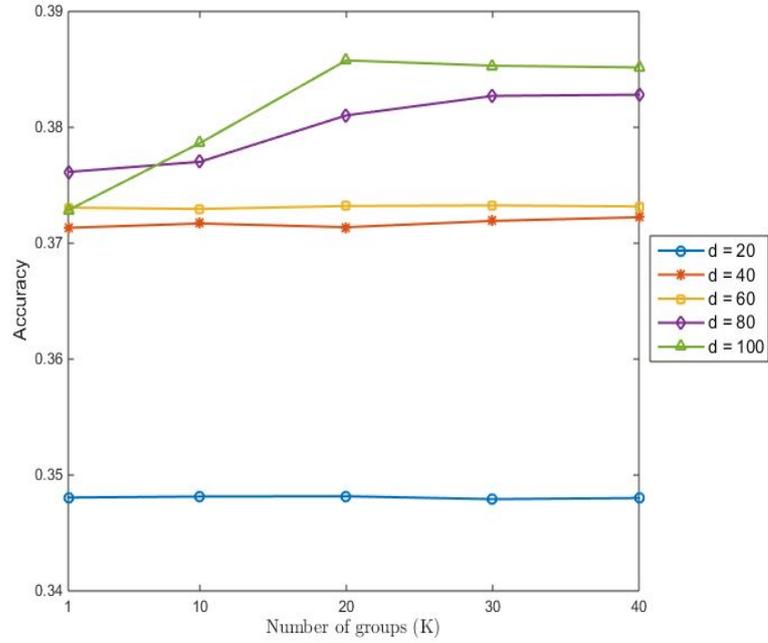
(a) Accuracy
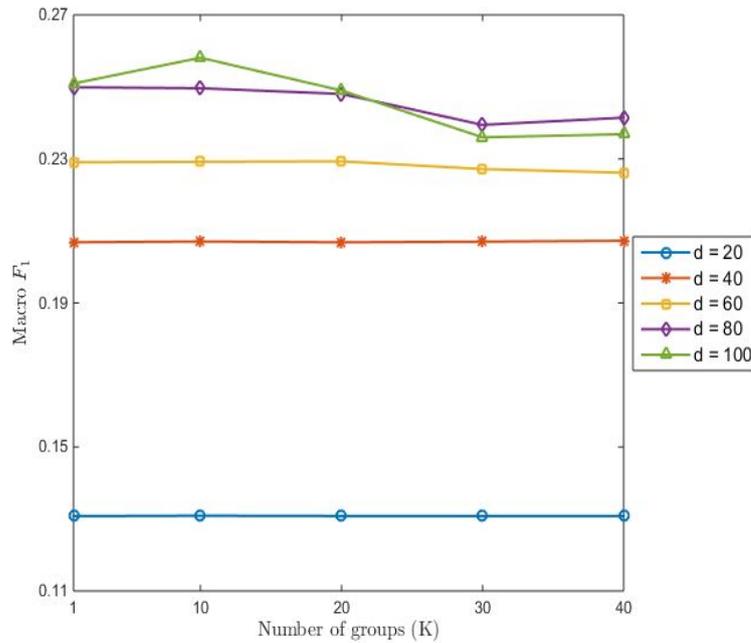
(b) Macro F_1

Figure 6.3: Performance of GroPLE on *rcv1 (subset 1)* data set with different group size.

classification performance of GroPLE is nearly constant for different group size when the latent dimension d is small which is also obvious as there are less number of features to differentiate between one group from others. The classification performance improved as we increased the number of groups K for sufficiently large d. It can also be seen that the performance in terms of *Macro F_1* degrade for sufficiently large group

size K. Hence, by considering the balance between latent dimension d and number of groups K, we fixed the value of K and d to 10 and 100, respectively, for subsequent experiments.

Table 6.2: Experimental results of each comparing algorithm (mean±std rank) in terms of Accuracy, Example Based F_1, Macro F_1, and Micro F_1. Method that cannot be run with available resources are denoted as "-".

Data set	Accuracy							
	GroPLE	PLST	CPLST	CSSP	FAiE	LEML	MLSF	BSVM
genbase	0.972 ± 0.014	0.968 ± 0.013	0.971 ± 0.013	0.931 ± 0.042	0.965 ± 0.014	0.971 ± 0.014	$\mathbf{0.974 \pm 0.011}$	0.973 ± 0.010
medical	0.769 ± 0.031	0.762 ± 0.034	0.763 ± 0.033	0.735 ± 0.041	0.765 ± 0.035	0.690 ± 0.025	$\mathbf{0.779 \pm 0.029}$	0.753 ± 0.034
CAL500	0.234 ± 0.006	0.224 ± 0.007	0.224 ± 0.007	0.224 ± 0.006	$\mathbf{0.235 \pm 0.007}$	0.222 ± 0.006	0.177 ± 0.010	0.202 ± 0.007
corel5k	$\mathbf{0.151 \pm 0.002}$	0.054 ± 0.002	0.054 ± 0.002	0.052 ± 0.004	0.089 ± 0.003	0.053 ± 0.002	0.088 ± 0.008	0.081 ± 0.002
rcv1 (subset 1)	$\mathbf{0.335 \pm 0.003}$	0.231 ± 0.009	0.232 ± 0.008	0.218 ± 0.004	0.265 ± 0.008	0.223 ± 0.006	0.322 ± 0.071	0.288 ± 0.010
rcv1 (subset 2)	$\mathbf{0.371 \pm 0.011}$	0.299 ± 0.014	0.299 ± 0.013	0.297 ± 0.013	0.328 ± 0.011	0.286 ± 0.013	0.365 ± 0.010	0.348 ± 0.012
rcv1 (subset 3)	$\mathbf{0.375 \pm 0.016}$	0.289 ± 0.014	0.289 ± 0.014	0.276 ± 0.017	0.319 ± 0.018	0.280 ± 0.014	0.373 ± 0.009	0.347 ± 0.009
bibtex	$\mathbf{0.329 \pm 0.004}$	0.284 ± 0.009	0.288 ± 0.007	0.266 ± 0.010	0.310 ± 0.004	0.288 ± 0.007	0.328 ± 0.005	0.326 ± 0.008
corel16k001	$\mathbf{0.158 \pm 0.006}$	0.039 ± 0.001	0.038 ± 0.001	0.035 ± 0.003	0.080 ± 0.002	0.039 ± 0.002	0.034 ± 0.006	0.025 ± 0.002
delicious	$\mathbf{0.185 \pm 0.001}$	0.109 ± 0.002	0.109 ± 0.002	0.107 ± 0.002	0.134 ± 0.001	0.093 ± 0.001	0.118 ± 0.006	0.130 ± 0.001
mediamill	$\mathbf{0.434 \pm 0.004}$	0.414 ± 0.003	0.414 ± 0.003	0.406 ± 0.016	0.425 ± 0.004	0.411 ± 0.003	0.395 ± 0.012	0.393 ± 0.003
bookmarks	$\mathbf{0.283 \pm 0.002}$	0.174 ± 0.002		0.169 ± 0.007	-	0.175 ± 0.002	0.163 ± 0.004	-

Data set	Example based F_1							
	GroPLE	PLST	CPLST	CSSP	FAiE	LEML	MLSF	BSVM
genbase	0.978 ± 0.013	0.975 ± 0.012	0.977 ± 0.013	0.939 ± 0.043	0.973 ± 0.012	0.977 ± 0.013	0.978 ± 0.015	$\mathbf{0.980 \pm 0.010}$
medical	0.800 ± 0.031	0.793 ± 0.030	0.794 ± 0.033	0.766 ± 0.039	0.796 ± 0.034	0.738 ± 0.027	$\mathbf{0.807 \pm 0.026}$	0.783 ± 0.032
CAL500	$\mathbf{0.370 \pm 0.008}$	0.357 ± 0.009	0.358 ± 0.009	0.358 ± 0.007	0.369 ± 0.009	0.355 ± 0.008	0.296 ± 0.014	0.330 ± 0.010
corel5k	$\mathbf{0.230 \pm 0.003}$	0.078 ± 0.002	0.078 ± 0.002	0.075 ± 0.005	0.127 ± 0.004	0.076 ± 0.003	0.126 ± 0.005	0.114 ± 0.004
rcv1 (subset 1)	$\mathbf{0.447 \pm 0.002}$	0.298 ± 0.009	0.299 ± 0.009	0.283 ± 0.006	0.339 ± 0.009	0.288 ± 0.006	0.399 ± 0.006	0.379 ± 0.012
rcv1 (subset 2)	$\mathbf{0.466 \pm 0.011}$	0.346 ± 0.015	0.346 ± 0.014	0.343 ± 0.014	0.380 ± 0.011	0.331 ± 0.014	0.429 ± 0.011	0.423 ± 0.013
rcv1 (subset 3)	$\mathbf{0.471 \pm 0.016}$	0.335 ± 0.015	0.336 ± 0.015	0.321 ± 0.019	0.372 ± 0.019	0.325 ± 0.014	0.434 ± 0.015	0.422 ± 0.008
bibtex	$\mathbf{0.415 \pm 0.004}$	0.335 ± 0.009	0.339 ± 0.008	0.314 ± 0.010	0.367 ± 0.004	0.338 ± 0.008	0.399 ± 0.009	0.383 ± 0.008
corel16k001	$\mathbf{0.227 \pm 0.006}$	0.053 ± 0.002	0.053 ± 0.002	0.047 ± 0.005	0.111 ± 0.002	0.053 ± 0.002	0.035 ± 0.011	0.034 ± 0.002
delicious	$\mathbf{0.293 \pm 0.001}$	0.169 ± 0.002	0.169 ± 0.002	0.167 ± 0.003	0.209 ± 0.001	0.142 ± 0.002	0.181 ± 0.010	0.201 ± 0.002
mediamill	$\mathbf{0.552 \pm 0.004}$	0.533 ± 0.004	0.533 ± 0.004	0.524 ± 0.018	0.544 ± 0.004	0.530 ± 0.003	0.507 ± 0.019	0.515 ± 0.003
bookmarks	$\mathbf{0.324 \pm 0.006}$	0.179 ± 0.002	-	0.173 ± 0.007	-	0.180 ± 0.002	0.166 ± 0.005	-

Data set	Macro F_1							
	GroPLE	PLST	CPLST	CSSP	FAiE	LEML	MLSF	BSVM
genbase	0.711 ± 0.086	0.709 ± 0.068	0.711 ± 0.075	0.673 ± 0.069	0.682 ± 0.066	0.705 ± 0.078	0.736 ± 0.084	$\mathbf{0.737 \pm 0.082}$
medical	0.374 ± 0.034	0.373 ± 0.025	0.378 ± 0.030	0.360 ± 0.026	$\mathbf{0.383 \pm 0.025}$	0.342 ± 0.025	0.385 ± 0.036	0.383 ± 0.037
CAL500	$\mathbf{0.129 \pm 0.007}$	0.110 ± 0.006	0.104 ± 0.005	0.107 ± 0.005	0.118 ± 0.004	0.110 ± 0.005	0.034 ± 0.011	0.057 ± 0.001
corel5k	$\mathbf{0.049 \pm 0.002}$	0.016 ± 0.002	0.016 ± 0.001	0.015 ± 0.002	0.026 ± 0.001	0.016 ± 0.002	0.046 ± 0.002	0.044 ± 0.002
rcv1 (subset 1)	$\mathbf{0.263 \pm 0.006}$	0.126 ± 0.006	0.125 ± 0.005	0.119 ± 0.006	0.163 ± 0.009	0.125 ± 0.005	0.257 ± 0.014	0.257 ± 0.005
rcv1 (subset 2)	$\mathbf{0.247 \pm 0.003}$	0.111 ± 0.006	0.112 ± 0.006	0.109 ± 0.006	0.148 ± 0.004	0.111 ± 0.006	0.241 ± 0.006	0.240 ± 0.007
rcv1 (subset 3)	$\mathbf{0.244 \pm 0.004}$	0.109 ± 0.003	0.109 ± 0.003	0.101 ± 0.005	0.144 ± 0.007	0.108 ± 0.004	0.235 ± 0.013	0.231 ± 0.012
bibtex	0.304 ± 0.014	0.197 ± 0.005	0.208 ± 0.008	0.178 ± 0.007	0.236 ± 0.007	0.208 ± 0.008	0.326 ± 0.006	$\mathbf{0.327 \pm 0.004}$
corel16k001	$\mathbf{0.088 \pm 0.004}$	0.015 ± 0.001	0.015 ± 0.002	0.014 ± 0.002	0.023 ± 0.002	0.015 ± 0.002	0.040 ± 0.002	0.036 ± 0.006
delicious	0.089 ± 0.000	0.048 ± 0.001	0.048 ± 0.002	0.046 ± 0.002	0.059 ± 0.002	0.048 ± 0.002	$\mathbf{0.102 \pm 0.001}$	0.100 ± 0.003
mediamill	$\mathbf{0.087 \pm 0.001}$	0.045 ± 0.001	0.045 ± 0.001	0.043 ± 0.002	0.055 ± 0.001	0.043 ± 0.001	0.041 ± 0.001	0.032 ± 0.001
bookmarks	$\mathbf{0.140 \pm 0.003}$	0.057 ± 0.001	-	0.053 ± 0.004	-	0.059 ± 0.001	0.040 ± 0.001	-

Data set	Micro F_1							
	GroPLE	PLST	CPLST	CSSP	FAiE	LEML	MLSF	BSVM
genbase	0.967 ± 0.031	0.969 ± 0.012	0.972 ± 0.013	0.951 ± 0.022	0.967 ± 0.013	0.973 ± 0.013	0.978 ± 0.012	$\mathbf{0.979 \pm 0.008}$
medical	0.821 ± 0.028	0.821 ± 0.030	0.822 ± 0.029	0.806 ± 0.029	$\mathbf{0.823 \pm 0.030}$	0.739 ± 0.019	0.822 ± 0.033	0.812 ± 0.027
CAL500	$\mathbf{0.375 \pm 0.007}$	0.360 ± 0.008	0.361 ± 0.009	0.361 ± 0.006	0.374 ± 0.008	0.359 ± 0.007	0.290 ± 0.038	0.327 ± 0.009
corel5k	$\mathbf{0.242 \pm 0.003}$	0.105 ± 0.003	0.105 ± 0.002	0.101 ± 0.006	0.162 ± 0.004	0.103 ± 0.003	0.154 ± 0.007	0.143 ± 0.005
rcv1 (subset 1)	$\mathbf{0.463 \pm 0.005}$	0.350 ± 0.010	0.351 ± 0.010	0.336 ± 0.005	0.384 ± 0.009	0.344 ± 0.008	0.405 ± 0.006	0.394 ± 0.009
rcv1 (subset 2)	$\mathbf{0.455 \pm 0.009}$	0.373 ± 0.016	0.373 ± 0.015	0.372 ± 0.015	0.403 ± 0.012	0.362 ± 0.015	0.413 ± 0.012	0.411 ± 0.009
rcv1 (subset 3)	$\mathbf{0.460 \pm 0.010}$	0.366 ± 0.009	0.366 ± 0.009	0.353 ± 0.012	0.396 ± 0.013	0.360 ± 0.010	0.415 ± 0.011	0.409 ± 0.008
bibtex	0.417 ± 0.015	0.390 ± 0.006	0.396 ± 0.007	0.371 ± 0.007	0.420 ± 0.004	0.396 ± 0.007	$\mathbf{0.424 \pm 0.007}$	$\mathbf{0.424 \pm 0.002}$
corel16k001	$\mathbf{0.256 \pm 0.007}$	0.071 ± 0.003	0.070 ± 0.003	0.063 ± 0.008	0.137 ± 0.004	0.071 ± 0.004	0.050 ± 0.013	0.049 ± 0.003
delicious	$\mathbf{0.231 \pm 0.004}$	0.194 ± 0.003	0.194 ± 0.003	0.191 ± 0.003	0.241 ± 0.002	0.172 ± 0.002	0.208 ± 0.011	0.226 ± 0.003
mediamill	$\mathbf{0.581 \pm 0.003}$	0.547 ± 0.003	0.547 ± 0.003	0.539 ± 0.012	0.562 ± 0.003	0.543 ± 0.002	0.524 ± 0.012	0.520 ± 0.002
bookmarks	$\mathbf{0.278 \pm 0.008}$	0.201 ± 0.003	-	0.192 ± 0.012	-	0.202 ± 0.002	0.180 ± 0.004	-

Table 6.2 gives the comparative analysis of the proposed method GroPLE against state-of-the-art algorithms on eleven data sets. We have conducted *five-fold cross validation* and the $mean$ and std is recorded. The best results among all the algorithms

being compared are highlighted in boldface. For each data set, the number of latent dimension space d is fixed to 100 and the number of groups K is set to 10. The regularization parameter λ_1 and λ_2 are fixed to 0.001 and 1, respectively, and the parameters α and β are searched in the range given previously.

Table 6.3: Summary of the Friedman statistics $F_F(\mathcal{K} = 8, \mathcal{N} = 11)$ and the critical value in terms of each evaluation metric($\mathcal{K}$: # Comparing Algorithms; $\mathcal{N}$: # Data Sets).

Metric	F_F	Critical Value ($\alpha = 0.05$)
Accuracy	11.692	
Example Base F_1	12.353	
Macro F_1	12.686	2.143
Micro F_1	7.470	

To conduct statistical performance analysis among the algorithms being compared, we employed *Friedman test*[3] which is a favorable statistical test for comparing more than two algorithms over multiple data sets (24). Table 6.3 provides the *Friedman statistics F_F* and the corresponding critical value in terms of each evaluation metric. As shown in Table 6.3 at significance level $\alpha = 0.05$, Friedman test rejects the null hypothesis of equal performance for each evaluation metric. This leads to the use of post-hoc tests for pairwise comparisons. The Nemenyi test (24) is employed to test whether our proposed method GroPLE achieves a competitive performance against the algorithms being compared. The performance of two classifiers is significantly different if the corresponding average ranks differ by at least the critical difference $CD = q_\alpha \sqrt{\frac{\mathcal{K}(\mathcal{K}+1)}{6\mathcal{N}}}$. At significance level $\alpha = 0.05$, the value of $q_\alpha = 3.031$, for Nemenyi test with $\mathcal{K} = 8$ (24), and thus $CD = 3.166$. Figure 6.4 gives the CD diagrams (24) for each evaluation criterion, where the average rank of each comparing algorithm is marked along the axis (lower ranks to the left). It can be seen from the Figure 6.4 that the proposed method GroPLE achieve better performance as compared to the other algorithms in terms of each evaluation metric.

---

[3]Results of bookmarks data set is not included for this test.

(a) Accuracy

(b) Example based F_1

(c) Macro F_1

(d) Micro F_1
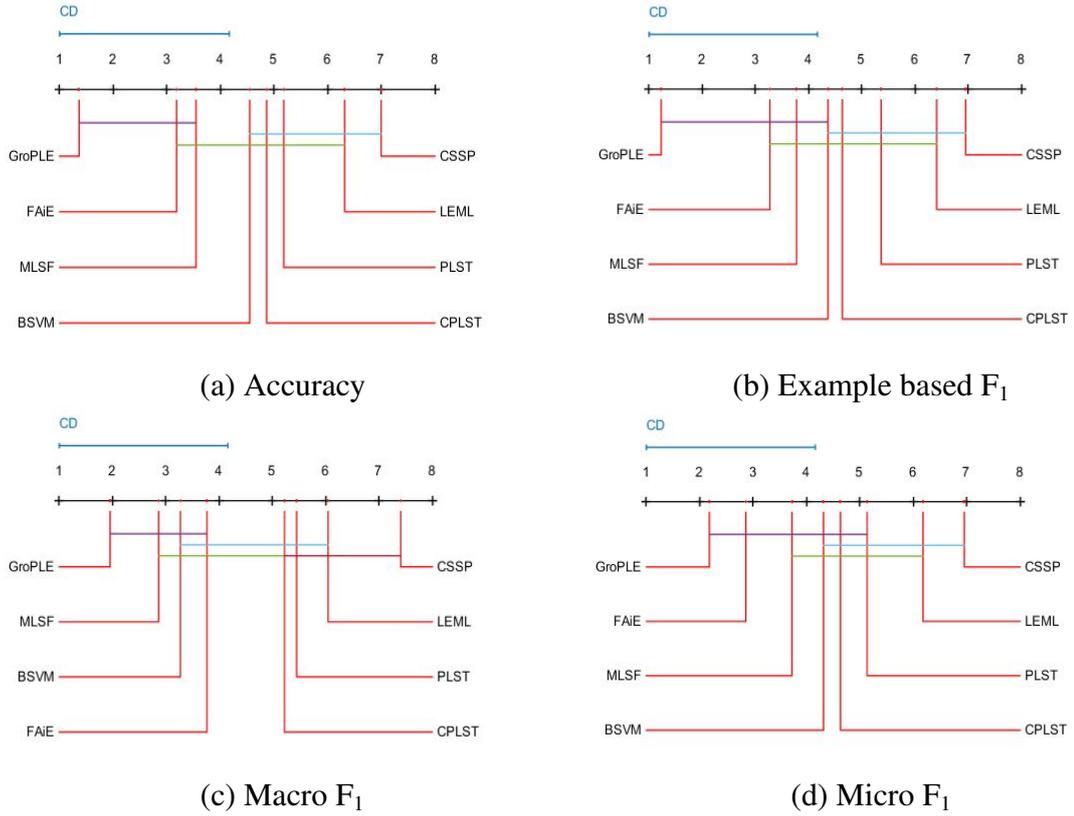
Figure 6.4: CD diagrams of the comparing algorithms under each evaluation criterion.

6.4 Conclusions

This chapter presented a new multi-label classification method, called GroPLE, which embeds the original label vectors to a low-rank space by retaining the group dependencies. We ensure that the labels belonging to the same group share the same sparsity pattern in their low-rank representations. In order to achieve the embedding of feature vectors, a linear mapping is then learnt that maps the feature vectors onto the same set of points which are obtained as a result of label embedding phase. We achieve this by a separate optimization problem. Extensive comparative studies validate the effectiveness of GroPLE against the state-of-the-art multi-label learning approaches.

In the future, it is interesting to see whether GroPLE can be further improved by considering side information from feature space while label embedding. Furthermore, designing other ways to fulfill the strategy of group formation and modeling group-specific label embedding is a direction worth studying.

CHAPTER 7

Conclusions and Future Work

In this thesis, we focused on the development of novel techniques for collaborative filtering and multi-label classification, which are two important research areas in the domain of recommender systems and machine learning. We show that the problem of collaborative filtering (CF) and multi-label classification (MLC) can be visualized as a matrix factorization (MF) problem. In Chapter 3 and Chapter 4, we discussed two novel approaches for CF namely, HMF (*Hierarchical Matrix Factorization*) and PMMMF (*Proximal Maximum Margin Matrix Factorization*), respectively. The basic factorization technique used in Chapters 3 and 4 is that of MMMF (*Maximum Margin Matrix Factorization*) wherein a multi-class extension of the hinge loss function is proposed to handle the factorization of discrete ordinal rating matrix. Assuming MMMF to be matrix analog of ordinal regression under hinge loss, in Chapter 3, we examined whether a set of bi-level maximum margin matrix factorizations can be used to solve matrix completion problem for multi-level ordinal rating matrix in more efficient and effective manner. We proposed a novel method termed as HMF for constructing a hierarchical two-class structure of binary matrix factorization to handle matrix completion of ordinal rating matrix. Our experimental results demonstrate the advantages of the proposed method over other matrix factorization based collaborative filtering techniques.

It is more than a decade since MMMF has been proposed and till date, to the best of our knowledge, no other alternative criterion has been investigated for matrix factorization other than the maximum margin criterion. In Chapter 4, we proposed a different criterion, namely *proximity*, motivated by the advent of Proximal SVMs (PSVMs) for binary classification, where two *parallel* planes are generated one for each class, un-

like the standard SVMs. We conducted experiments on real and synthetic datasets to validate the effectiveness of PMMMF.

In the later part of the thesis we extended the concept of matrix factorization for multi-label classification and proposed two novel techniques namely, MLC-HMF (*Multi-label Classification Using Hierarchical Embedding*) and GroPLE (*Group Preserving Label Embedding for Multi-Label Classification*) in Chapters 5 and 6, respectively. In Chapter 5, we developed a matrix factorization based feature space embedding method termed as MLC-HMF for multi-label classification. MLC-HMF is an embedding based approach which learns piecewise-linear embedding with a low-rank constraint on parametrization to capture nonlinear intrinsic relationships that exists in the original feature and label space. We conducted extensive comparative studies which validate the effectiveness of the proposed method MLC-HMF. In Chapter 6, we assume that the input data form groups and as a result, the label matrix exhibits a sparsity pattern and hence the labels corresponding to objects in the same group have similar sparsity. We study the embedding of labels together with the group information with an objective to build an efficient multi-label classification. Extensive comparative studies validate the effectiveness of GroPLE against the state-of-the-art multi-label learning approaches.

There are many interesting directions in which the research work depicted in this thesis can be carried out in the future. As discussed earlier, HMF and PMMMF are novel matrix factorization techniques developed for collaborative filtering with ordinal preferences. HMF is a stage-wise matrix completion technique that makes use of several bi-level MMMFs in a hierarchical fashion. Investigating different ways in which a hierarchy can be built is one important direction that can be pursued. The core part of HMF is a binary loss function and studying the impact of different binary loss functions can be another direction to pursue future research. As discussed in Chapter 4, in every iteration of the gradient descent method PMMMF requires less number of variables to be updated as compared to that of MMMF. But on the other hand, the updation step in PMMMF takes more time because of the repeated computation of the *mean* for obtaining the *thresholds* for every user. To reduce the computation cost, one can think of a better approximation of the mean and the thresholds or even think of designing novel optimization techniques that best suit the optimization problem of PMMMF. We realize that there are more general problems of matrix factorization for matrices having discrete entries which are not ordered. Extending the concept of HMF and PMMMF

for such matrices is another line of investigation that can be carried out in the future.

In Chapter 5 and Chapter 6, we proposed two different embedding approach for multi-label classification namely, MLC-HMF and GroPLE, respectively. MLC-HMF is based on the embedding of feature space whereas GroPLE is a label space embedding approach. We developed novel matrix factorization techniques to achieve the embedding of feature and label space. It is worthwhile to carryout an in depth investigation of different ways of embedding such as quasilinear embeddings and their advantages over linear and non-linear embedding. One can plan to carryout this line of investigation in the future. Reducing the computational complexity of MLC-HMF by investigating the different ways of building an hierarchy or formation of group instances is another direction to pursue. In the future, it is interesting to see whether GroPLE can be further improved by considering the side information from the feature space while label embedding. Furthermore, designing other ways to fulfill the strategy of group formation and modeling group-specific label embedding is a direction worth studying.

REFERENCES

[1] Rohit Babbar and Bernhard Schölkopf. Dismec: distributed sparse machines for extreme multi-label classification. In *Proceedings of the Tenth ACM International Conference on Web Search and Data Mining*, pages 721–729. ACM, 2017.

[2] Claus Bahlmann, Bernard Haasdonk, and Hans Burkhardt. Online handwriting recognition with support vector machines-a kernel approach. In *Frontiers in handwriting recognition, 2002. proceedings. eighth international workshop on*, pages 49–54. IEEE, 2002.

[3] Krishnakumar Balasubramanian and Guy Lebanon. The landmark selection method for multiple output prediction. *arXiv preprint arXiv:1206.6479*, 2012.

[4] Amir Beck and Marc Teboulle. A fast iterative shrinkage-thresholding algorithm for linear inverse problems. *SIAM journal on imaging sciences*, 2(1):183–202, 2009.

[5] Kush Bhatia, Himanshu Jain, Purushottam Kar, Manik Varma, and Prateek Jain. Sparse local embeddings for extreme multi-label classification. In *Advances in Neural Information Processing Systems*, pages 730–738, 2015.

[6] Wei Bi and James Kwok. Efficient multi-label classification with many labels. In *International Conference on Machine Learning*, pages 405–413, 2013.

[7] Wei Bi and James T Kwok. Multilabel classification with label correlations and missing labels. In *AAAI*, pages 1680–1686, 2014.

[8] Mathieu Blondel, Kazuhiro Seki, and Kuniaki Uehara. Block coordinate descent algorithms for large-scale sparse multiclass classification. *Machine learning*, 93(1):31–52, 2013.

[9] Matthew R Boutell, Jiebo Luo, Xipeng Shen, and Christopher M Brown. Learning multi-label scene classification. *Pattern recognition*, 37(9):1757–1771, 2004.

[10] Christopher JC Burges. A tutorial on support vector machines for pattern recognition. *Data mining and knowledge discovery*, 2(2):121–167, 1998.

[11] Hyeran Byun and Seong-Whan Lee. Applications of support vector machines for pattern recognition: A survey. In *Pattern recognition with support vector machines*, pages 213–236. Springer, 2002.

[12] Ricardo S Cabral, Fernando Torre, João P Costeira, and Alexandre Bernardino. Matrix completion for multi-label image classification. In *Advances in Neural Information Processing Systems*, pages 190–198, 2011.

[13] Xiangyong Cao, Yang Chen, Qian Zhao, Deyu Meng, Yao Wang, Dong Wang, and Zongben Xu. Low-rank matrix factorization under general mixture noise distributions. In *Proceedings of the IEEE International Conference on Computer Vision*, pages 1493–1501, 2015.

[14] Xiangyong Cao, Qian Zhao, Deyu Meng, Yang Chen, and Zongben Xu. Robust low-rank matrix factorization under general mixture noise distributions. *IEEE Transactions on Image Processing*, 25(10):4677–4690, 2016.

[15] Chih-Chung Chang and Chih-Jen Lin. LIBSVM: A library for support vector machines. *ACM Transactions on Intelligent Systems and Technology*, 2:27:1–27:27, 2011.

[16] Yao-Nan Chen and Hsuan-Tien Lin. Feature-aware label space dimension reduction for multi-label classification. In *Advances in Neural Information Processing Systems*, pages 1529–1537, 2012.

[17] Andrzej Cichocki, Rafal Zdunek, and Shun-ichi Amari. CsiszarâĂŹs divergences for non-negative matrix factorization: Family of new algorithms. In *International Conference on Independent Component Analysis and Signal Separation*, pages 32–39. Springer, 2006.

[18] Amanda Clare and Ross D King. Knowledge discovery in multi-label phenotype data. In *European Conference on Principles of Data Mining and Knowledge Discovery*, pages 42–53. Springer, 2001.

[19] Jose A Costa, Neal Patwari, and Alfred O Hero III. Distributed weighted-multidimensional scaling for node localization in sensor networks. *ACM Transactions on Sensor Networks (TOSN)*, 2(1):39–64, 2006.

[20] Koby Crammer and Yoram Singer. On the algorithmic implementation of multiclass kernel-based vector machines. *Journal of machine learning research*, 2(Dec):265–292, 2001.

[21] Fernando De la Torre. A least-squares framework for component analysis. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 34(6):1041–1055, 2012.

[22] Rameswar Debnath, N Takahide, and Haruhisa Takahashi. A decision based one-against-one method for multi-class support vector machine. *Pattern Analysis and Applications*, 7(2):164–175, 2004.

[23] Dennis DeCoste. Collaborative prediction using ensembles of maximum margin matrix factorizations. In *Proceedings of the 23rd international conference on Machine learning*, pages 249–256. ACM, 2006.

[24] Janez Demšar. Statistical comparisons of classifiers over multiple data sets. *Journal of Machine learning research*, 7(Jan):1–30, 2006.

[25] Brian Eriksson, Laura Balzano, and Robert Nowak. High-rank matrix completion. In *Artificial Intelligence and Statistics*, pages 373–381, 2012.

[26] Ali Fakhari and Amir Masoud Eftekhari Moghadam. Combination of classification and regression in decision tree for multi-labeling image annotation and retrieval. *Applied Soft Computing*, 13(2):1292–1302, 2013.

[27] Maryam Fazel. *Matrix rank minimization with applications*. PhD thesis, PhD thesis, Stanford University, 2002.

[28] Maryam Fazel, Haitham Hindi, and Stephen P Boyd. A rank minimization heuristic with application to minimum order system approximation. In *American Control Conference, 2001. Proceedings of the 2001*, volume 6, pages 4734–4739. IEEE, 2001.

[29] Cédric Févotte, Nancy Bertin, and Jean-Louis Durrieu. Nonnegative matrix factorization with the itakura-saito divergence: With application to music analysis. *Neural computation*, 21(3):793–830, 2009.

[30] Cédric Févotte and Nicolas Dobigeon. Nonlinear hyperspectral unmixing with robust nonnegative matrix factorization. *IEEE Transactions on Image Processing*, 24(12):4810–4819, 2015.

[31] Glenn Fung and Olvi L. Mangasarian. Proximal support vector machine classifiers. In *Proceedings of the Seventh ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, KDD '01, pages 77–86, New York, NY, USA, 2001. ACM.

[32] Johannes Fürnkranz, Eyke Hüllermeier, Eneldo Loza Mencía, and Klaus Brinker. Multilabel classification via calibrated label ranking. *Machine learning*, 73(2):133–153, 2008.

[33] Nadia Ghamrawi and Andrew McCallum. Collective multi-label classification. In *Proceedings of the 14th ACM international conference on Information and knowledge management*, pages 195–200. ACM, 2005.

[34] Amir Globerson and Sam Roweis. Nightmare at test time: robust learning by feature deletion. In *Proceedings of the 23rd international conference on Machine learning*, pages 353–360. ACM, 2006.

[35] Shuhang Gu, Lei Zhang, Wangmeng Zuo, and Xiangchu Feng. Weighted nuclear norm minimization with application to image denoising. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 2862–2869, 2014.

[36] Christine Guillemot and Olivier Le Meur. Image inpainting: Overview and recent advances. *IEEE signal processing magazine*, 31(1):127–144, 2014.

[37] Benjamin Haeffele, Eric Young, and Rene Vidal. Structured low-rank matrix factorization: Optimality, algorithm, and applications to image processing. In *International Conference on Machine Learning*, pages 2007–2015, 2014.

[38] Thomas Hofmann. Latent semantic models for collaborative filtering. *ACM Transactions on Information Systems (TOIS)*, 22(1):89–115, 2004.

[39] Patrik O Hoyer. Non-negative matrix factorization with sparseness constraints. *Journal of machine learning research*, 5(Nov):1457–1469, 2004.

[40] Chih-Wei Hsu and Chih-Jen Lin. A comparison of methods for multiclass support vector machines. *IEEE transactions on Neural Networks*, 13(2):415–425, 2002.

[41] Daniel J Hsu, Sham Kakade, John Langford, and Tong Zhang. Multi-label prediction via compressed sensing. In *NIPS*, volume 22, pages 772–780, 2009.

[42] Yifan Hu, Yehuda Koren, and Chris Volinsky. Collaborative filtering for implicit feedback datasets. In *Data Mining, 2008. ICDM'08. Eighth IEEE International Conference on*, pages 263–272. IEEE, 2008.

[43] Jun Huang, Guorong Li, Qingming Huang, and Xindong Wu. Learning label specific features for multi-label classification. In *Data Mining (ICDM), 2015 IEEE International Conference on*, pages 181–190. IEEE, 2015.

[44] Jun Huang, Guorong Li, Qingming Huang, and Xindong Wu. Learning label-specific features and class-dependent labels for multi-label classification. *IEEE Transactions on Knowledge and Data Engineering*, 28(12):3309–3323, 2016.

[45] Sheng-Jun Huang, Yang Yu, and Zhi-Hua Zhou. Multi-label hypothesis reuse. In *Proceedings of the 18th ACM SIGKDD international conference on Knowledge discovery and data mining*, pages 525–533. ACM, 2012.

[46] Sheng-Jun Huang, Zhi-Hua Zhou, and ZH Zhou. Multi-label learning by exploiting label correlations locally. In *Twenty-Sixth AAAI Conference on Artificial Intelligence*, pages 949–955, 2012.

[47] Anil K Jain. Data clustering: 50 years beyond k-means. *Pattern recognition letters*, 31(8):651–666, 2010.

[48] Anil K Jain, M Narasimha Murty, and Patrick J Flynn. Data clustering: a review. *ACM computing surveys (CSUR)*, 31(3):264–323, 1999.

[49] Prateek Jain, Praneeth Netrapalli, and Sujay Sanghavi. Low-rank matrix completion using alternating minimization. In *Proceedings of the forty-fifth annual ACM symposium on Theory of computing*, pages 665–674. ACM, 2013.

[50] Jayadeva, Reshma Khemchandani, and Suresh Chandra. Twin support vector machines for pattern classification. *IEEE Trans. Pattern Anal. Mach. Intell.*, 29(5):905–910, 2007.

[51] Shuiwang Ji and Jieping Ye. Linear dimensionality reduction for multi-label classification. In *IJCAI*, volume 9, pages 1077–1082, 2009.

[52] Ling Jian, Jundong Li, Kai Shu, and Huan Liu. Multi-label informed feature selection. In *25th International Joint Conference on Artificial Intelligence*, pages 1627–1633, 2016.

[53] Liping Jing, Liu Yang, Jian Yu, and Michael K Ng. Semi-supervised low-rank mapping learning for multi-label classification. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 1483–1491, 2015.

[54] Stephen C Johnson. Hierarchical clustering schemes. *Psychometrika*, 32(3):241–254, 1967.

[55] Nikos Karampatziakis and Paul Mineiro. Scalable multilabel prediction via randomized methods. *arXiv preprint arXiv:1502.02710*, 2015.

[56] Dongmin Kim, Suvrit Sra, and Inderjit S Dhillon. Fast newton-type methods for the least squares nonnegative matrix approximation problem. In *Proceedings of the 2007 SIAM international conference on data mining*, pages 343–354. SIAM, 2007.

[57] Hannah Kim, Jaegul Choo, Jingu Kim, Chandan K Reddy, and Haesun Park. Simultaneous discovery of common and discriminative topics via joint nonnegative matrix factorization. In *Proceedings of the 21th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, pages 567–576. ACM, 2015.

[58] Jingu Kim, Renato DC Monteiro, and Haesun Park. Group sparsity in non-negative matrix factorization. In *Proceedings of the 2012 SIAM International Conference on Data Mining*, pages 851–862. SIAM, 2012.

[59] Jingu Kim and Haesun Park. Toward faster nonnegative matrix factorization: A new algorithm and comparisons. In *Data Mining, 2008. ICDM'08. Eighth IEEE International Conference on*, pages 353–362. IEEE, 2008.

[60] Jingu Kim and Haesun Park. Fast nonnegative matrix factorization: An active-set-like method and comparisons. *SIAM Journal on Scientific Computing*, 33(6):3261–3281, 2011.

[61] Keigo Kimura, Mineichi Kudo, and Lu Sun. Simultaneous nonlinear label-instance embedding for multi-label classification. In *S+SSPR*, pages 15–25. Springer, 2016.

[62] Yehuda Koren. Factorization meets the neighborhood: a multifaceted collaborative filtering model. In *Proceedings of the 14th ACM SIGKDD international conference on Knowledge discovery and data mining*, pages 426–434. ACM, 2008.

[63] Yehuda Koren. Collaborative filtering with temporal dynamics. *Communications of the ACM*, 53(4):89–97, 2010.

[64] Yehuda Koren and Robert Bell. Advances in collaborative filtering. In *Recommender systems handbook*, pages 145–186. Springer, 2011.

[65] Yehuda Koren, Robert Bell, and Chris Volinsky. Matrix factorization techniques for recommender systems. *Computer*, 42(8), 2009.

[66] Vikas Kumar, Arun K Pujari, Sandeep Kumar Sahu, Venkateswara Rao Kagita, and Vineet Padmanabhan. Collaborative filtering using multiple binary maximum margin matrix factorizations. *Information Sciences*, 380:1–11, 2017.

[67] Neil D Lawrence and Raquel Urtasun. Non-linear matrix factorization with Gaussian processes. In *Proceedings of the 26th Annual International Conference on Machine Learning*, pages 601–608. ACM, 2009.

[68] Daniel D Lee and H Sebastian Seung. Learning the parts of objects by non-negative matrix factorization. *Nature*, 401(6755):788–791, 1999.

[69] Daniel D Lee and H Sebastian Seung. Algorithms for non-negative matrix factorization. In *Advances in neural information processing systems*, pages 556–562, 2001.

[70] Daniel D Lee and H Sebastian Seung. Algorithms for non-negative matrix factorization. In *Advances in neural information processing systems*, pages 556–562, 2001.

[71] Yoonkyung Lee, Yi Lin, and Grace Wahba. Multicategory support vector machines: Theory and application to the classification of microarray data and satellite radiance data. *Journal of the American Statistical Association*, 99(465):67–81, 2004.

[72] Xi Li, Xueyi Zhao, Zhongfei Zhang, Fei Wu, Yueting Zhuang, Jingdong Wang, and Xuelong Li. Joint multilabel classification with community-aware label graph learning. *IEEE Transactions on Image Processing*, 25(1):484–493, 2016.

[73] Xin Li and Yuhong Guo. Multi-label classification with feature-aware non-linear label space transformation. In *IJCAI*, pages 3635–3642, 2015.

[74] Ya Li, Xinmei Tian, Mingli Song, and Dacheng Tao. Multi-task proximal support vector machine. *Pattern Recognition*, 48(10):3249–3257, 2015.

[75] Matthew M Lin, Bo Dong, and Moody T Chu. Integer matrix factorization and its application. *Tech. Rep.*, 2005.

[76] Zijia Lin, Guiguang Ding, Mingqing Hu, and Jianmin Wang. Multi-label classification via feature-aware implicit label space encoding. In *ICML*, pages 325–333, 2014.

[77] Qing Ling, Yangyang Xu, Wotao Yin, and Zaiwen Wen. Decentralized low-rank matrix completion. In *Acoustics, Speech and Signal Processing (ICASSP), 2012 IEEE International Conference on*, pages 2925–2928. IEEE, 2012.

[78] Jialu Liu, Chi Wang, Jing Gao, and Jiawei Han. Multi-view clustering via joint nonnegative matrix factorization. In *Proceedings of the 2013 SIAM International Conference on Data Mining*, pages 252–260. SIAM, 2013.

[79] Meng Liu, Yong Luo, Dacheng Tao, Chao Xu, and Yonggang Wen. Low-rank multi-view learning in matrix completion for multi-label image classification. In *AAAI*, pages 2778–2784, 2015.

[80] Yufeng Liu, Hao Helen Zhang, and Yichao Wu. Hard or soft classification? large-margin unified machines. *Journal of the American Statistical Association*, 106(493):166–177, 2011.

[81] Jie Lu, Dianshuang Wu, Mingsong Mao, Wei Wang, and Guangquan Zhang. Recommender system application developments: a survey. *Decision Support Systems*, 74:12–32, 2015.

[82] Si Lu, Xiaofeng Ren, and Feng Liu. Depth enhancement via low-rank matrix completion. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 3390–3397, 2014.

[83] Julien Mairal, Jean Ponce, Guillermo Sapiro, Andrew Zisserman, and Francis R Bach. Supervised dictionary learning. In *Advances in neural information processing systems*, pages 1033–1040, 2009.

[84] Olvi L Mangasarian et al. Generalized support vector machines. *Advances in Neural Information Processing Systems*, pages 135–146, 1999.

[85] Benjamin M Marlin. Modeling user rating profiles for collaborative filtering. In *Advances in neural information processing systems*, page None, 2003.

[86] Andriy Mnih and Ruslan Salakhutdinov. Probabilistic matrix factorization. In *Advances in neural information processing systems*, pages 1257–1264, 2007.

[87] Jinseok Nam, Jungi Kim, Eneldo Loza Mencía, Iryna Gurevych, and Johannes Fürnkranz. Large-scale multi-label text classificationâĂŤrevisiting neural networks. In *Joint european conference on machine learning and knowledge discovery in databases*, pages 437–452. Springer, 2014.

[88] Nathan Srebro Nati and Tommi Jaakkola. Weighted low-rank approximations. In *In 20th International Conference on Machine Learning*, pages 720–727. AAAI Press, 2003.

[89] Andrew Y Ng, Michael I Jordan, and Yair Weiss. On spectral clustering: Analysis and an algorithm. In *Advances in neural information processing systems*, pages 849–856, 2002.

[90] Pentti Paatero. Least squares formulation of robust non-negative factor analysis. *Chemometrics and intelligent laboratory systems*, 37(1):23–35, 1997.

[91] Arkadiusz Paterek. Improving regularized singular value decomposition for collaborative filtering. In *Proceedings of KDD cup and workshop*, volume 2007, pages 5–8, 2007.

[92] Yashoteja Prabhu and Manik Varma. Fastxml: A fast, accurate and stable tree-classifier for extreme multi-label learning. In *Proceedings of the 20th ACM SIGKDD international conference on Knowledge discovery and data mining*, pages 263–272. ACM, 2014.

[93] Buyue Qian and Ian Davidson. Semi-supervised dimension reduction for multi-label classification. In *AAAI*, volume 10, pages 569–574, 2010.

[94] Piyush Rai, Changwei Hu, Ricardo Henao, and Lawrence Carin. Large-scale bayesian multi-label learning via topic-based label embeddings. In *Advances in Neural Information Processing Systems*, pages 3222–3230, 2015.

[95] Jesse Read, Bernhard Pfahringer, Geoff Holmes, and Eibe Frank. Classifier chains for multi-label classification. *Machine Learning and Knowledge Discovery in Databases*, pages 254–269, 2009.

[96] Steffen Rendle, Christoph Freudenthaler, Zeno Gantner, and Lars Schmidt-Thieme. Bpr: Bayesian personalized ranking from implicit feedback. In *Proceedings of the Twenty-Fifth Conference on Uncertainty in Artificial Intelligence*, pages 452–461. AUAI Press, 2009.

[97] Jason DM Rennie and Nathan Srebro. Fast maximum margin matrix factorization for collaborative prediction. In *Proceedings of the 22nd international conference on Machine learning*, pages 713–719. ACM, 2005.

[98] Jason DM Rennie and Nathan Srebro. Loss functions for preference levels: Regression with discrete ordered labels. In *Proceedings of the IJCAI multidisciplinary workshop on advances in preference handling*, pages 180–186. Kluwer Norwell, MA, 2005.

[99] Ryan Rifkin and Aldebaro Klautau. In defense of one-vs-all classification. *Journal of machine learning research*, 5(Jan):101–141, 2004.

[100] Ruslan Salakhutdinov and Andriy Mnih. Bayesian probabilistic matrix factorization using markov chain monte carlo. In *Proceedings of the 25th international conference on Machine learning*, pages 880–887. ACM, 2008.

[101] Ruslan Salakhutdinov, Andriy Mnih, and Geoffrey Hinton. Restricted boltzmann machines for collaborative filtering. In *Proceedings of the 24th international conference on Machine learning*, pages 791–798. ACM, 2007.

[102] Badrul Sarwar, George Karypis, Joseph Konstan, and John Riedl. Incremental singular value decomposition algorithms for highly scalable recommender systems. In *Fifth International Conference on Computer and Information Science*, pages 27–28. Citeseer, 2002.

[103] Robert E Schapire and Yoram Singer. Boostexter: A boosting-based system for text categorization. *Machine learning*, 39(2-3):135–168, 2000.

[104] Ajit P Singh and Geoffrey J Gordon. A unified view of matrix factorization models. In *Machine Learning and Knowledge Discovery in Databases*, pages 358–373. Springer, 2008.

[105] Mohammad S Sorower. A literature survey on algorithms for multi-label learning. *Oregon State University, Corvallis*, 2010.

[106] Eleftherios Spyromitros, Grigorios Tsoumakas, and Ioannis Vlahavas. An empirical study of lazy multilabel classification algorithms. In *Hellenic conference on Artificial Intelligence*, pages 401–406. Springer, 2008.

[107] Nathan Srebro, Jason Rennie, and Tommi S Jaakkola. Maximum-margin matrix factorization. In *Advances in neural information processing systems*, pages 1329–1336, 2004.

[108] Lu Sun, Mineichi Kudo, and Keigo Kimura. Multi-label classification with meta-label-specific features. In *Pattern Recognition (ICPR), 2016 23rd International Conference on*, pages 1612–1617. IEEE, 2016.

[109] Farbound Tai and Hsuan-Tien Lin. Multilabel classification with principal label space transformation. *Neural Computation*, 24(9):2508–2542, 2012.

[110] Gábor Takács, István Pilászy, Bottyán Németh, and Domonkos Tikk. Major components of the gravity recommendation system. *ACM SIGKDD Explorations Newsletter*, 9(2):80–83, 2007.

[111] Jun Tang, Ke Wang, and Ling Shao. Supervised matrix factorization hashing for cross-modal retrieval. *IEEE Transactions on Image Processing*, 25(7):3157–3166, 2016.

[112] Kim-Chuan Toh and Sangwoon Yun. An accelerated proximal gradient algorithm for nuclear norm regularized linear least squares problems. *Pacific Journal of optimization*, 6(615-640):15, 2010.

[113] Grigorios Tsoumakas and Ioannis Katakis. Multi-label classification: An overview. *International Journal of Data Warehousing and Mining*, 3(3), 2006.

[114] Grigorios Tsoumakas, Ioannis Katakis, and Ioannis Vlahavas. Random k-labelsets for multilabel classification. *IEEE Transactions on Knowledge and Data Engineering*, 23(7):1079–1089, 2011.

[115] Grigorios Tsoumakas and Ioannis Vlahavas. Random k-labelsets: An ensemble method for multilabel classification. In *European Conference on Machine Learning*, pages 406–417. Springer, 2007.

[116] Satoru Tsuge, Masami Shishibori, Shingo Kuroiwa, and Kenji Kita. Dimensionality reduction using non-negative matrix factorization for information retrieval. In *Systems, Man, and Cybernetics, 2001 IEEE International Conference on*, volume 2, pages 960–965. IEEE, 2001.

[117] Mark H Van Benthem and Michael R Keenan. Fast algorithm for the solution of large-scale non-negativity-constrained least squares problems. *Journal of chemometrics*, 18(10):441–450, 2004.

[118] Vladimir Vapnik. *The nature of statistical learning theory*. Springer Science & Business Media, 2013.

[119] Maksims Volkovs and Guang Wei Yu. Effective latent models for binary feedback in recommender systems. In *Proceedings of the 38th International ACM SIGIR Conference on Research and Development in Information Retrieval*, SIGIR '15, pages 313–322, 2015.

[120] Ulrike Von Luxburg. A tutorial on spectral clustering. *Statistics and computing*, 17(4):395–416, 2007.

[121] Thanh T Vu, Benjamin Bigot, and Eng Siong Chng. Combining non-negative matrix factorization and deep neural networks for speech enhancement and automatic speech recognition. In *Acoustics, Speech and Signal Processing (ICASSP), 2016 IEEE International Conference on*, pages 499–503. IEEE, 2016.

[122] Fei Wang, Tao Li, and Changshui Zhang. Semi-supervised clustering via matrix factorization. In *Proceedings of the 2008 SIAM International Conference on Data Mining*, pages 1–12. SIAM, 2008.

[123] Jiang Wang, Yi Yang, Junhua Mao, Zhiheng Huang, Chang Huang, and Wei Xu. Cnn-rnn: A unified framework for multi-label image classification. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 2285–2294, 2016.

[124] Shiping Wang, Witold Pedrycz, Qingxin Zhu, and William Zhu. Subspace learning for unsupervised feature selection via matrix factorization. *Pattern Recognition*, 48(1):10–19, 2015.

[125] Zi Wang and Fei Sha. Discriminative non-negative matrix factorization for single-channel speech separation. In *Acoustics, Speech and Signal Processing (ICASSP), 2014 IEEE International Conference on*, pages 3749–3753. IEEE, 2014.

[126] Markus Weimer, Alexandros Karatzoglou, and Alexander J. Smola. Improving maximum margin matrix factorization. *Machine Learning*, 72(3):263–276, 2008.

[127] Kilian Q Weinberger and Lawrence K Saul. Unsupervised learning of image manifolds by semidefinite programming. *International journal of computer vision*, 70(1):77–90, 2006.

[128] Jason Weston and Chris Watkins. Multi-class support vector machines. Technical report, Technical Report CSD-TR-98-04, Department of Computer Science, Royal Holloway, University of London, May, 1998.

[129] Jason Weston, Chris Watkins, et al. Support vector machines for multi-class pattern recognition. In *ESANN*, volume 99, pages 219–224, 1999.

[130] Jörg Wicker, Bernhard Pfahringer, and Stefan Kramer. Multi-label classification using boolean matrix decomposition. In *Proceedings of the 27th Annual ACM Symposium on Applied Computing*, pages 179–186. ACM, 2012.

[131] Chang Xu, Dacheng Tao, and Chao Xu. Robust extreme multi-label learning. In *Proceedings of the 22nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, San Francisco, CA, USA August*, pages 13–17, 2016.

[132] Minjie Xu, Jun Zhu, and Bo Zhang. Nonparametric max-margin matrix factorization for collaborative prediction. In *Advances in Neural Information Processing Systems*, pages 64–72, 2012.

[133] Minjie Xu, Jun Zhu, and Bo Zhang. Fast max-margin matrix factorization with data augmentation. In *Proceedings of the 30th International Conference on Machine Learning (ICML-13)*, pages 978–986, 2013.

[134] Wei Xu, Xin Liu, and Yihong Gong. Document clustering based on non-negative matrix factorization. In *Proceedings of the 26th annual international ACM SIGIR conference on Research and development in informaion retrieval*, pages 267–273. ACM, 2003.

[135] Ian En-Hsu Yen, Xiangru Huang, Pradeep Ravikumar, Kai Zhong, and Inderjit Dhillon. Pd-sparse: A primal and dual sparse approach to extreme multiclass and multilabel classification. In *International Conference on Machine Learning*, pages 3069–3077, 2016.

[136] Hsiang-Fu Yu, Prateek Jain, Purushottam Kar, and Inderjit Dhillon. Large-scale multi-label learning with missing labels. In *International Conference on Machine Learning*, pages 593–601, 2014.

[137] Tingzhao Yu and Wensheng Zhang. Semisupervised multilabel learning with joint dimensionality reduction. *IEEE Signal Processing Letters*, 23(6):795–799, 2016.

[138] Yang Yu and Viktor K Prasanna. Energy-balanced task allocation for collaborative processing in wireless sensor networks. *Mobile Networks and Applications*, 10(1-2):115–131, 2005.

[139] Lihi Zelnik-Manor and Pietro Perona. Self-tuning spectral clustering. In *NIPS*, volume 17, page 16, 2004.

[140] Chong Zhang and Yufeng Liu. Multicategory large-margin unified machines. *The Journal of Machine Learning Research*, 14(1):1349–1386, 2013.

[141] Lefei Zhang, Liangpei Zhang, Dacheng Tao, Xin Huang, and Bo Du. Compression of hyperspectral remote sensing images by tensor approach. *Neurocomputing*, 147:358–363, 2015.

[142] Lefei Zhang, Qian Zhang, Bo Du, Dacheng Tao, and Jane You. Robust manifold matrix factorization for joint clustering and feature extraction. In *AAAI*, pages 1662–1668, 2017.

[143] Lefei Zhang, Qian Zhang, Liangpei Zhang, Dacheng Tao, Xin Huang, and Bo Du. Ensemble manifold regularized sparse low-rank approximation for multiview feature embedding. *Pattern Recognition*, 48(10):3102–3112, 2015.

[144] Min-Ling Zhang. Lift: Multi-label learning with label-specific features. In *IJCAI*, pages 1609–1614, 2011.

[145] Min-Ling Zhang and Lei Wu. Lift: Multi-label learning with label-specific features. *IEEE transactions on pattern analysis and machine intelligence*, 37(1):107–120, 2015.

[146] Min-Ling Zhang and Kun Zhang. Multi-label learning by exploiting label dependency. In *Proceedings of the 16th ACM SIGKDD international conference on Knowledge discovery and data mining*, pages 999–1008. ACM, 2010.

[147] Min-Ling Zhang and Zhi-Hua Zhou. Multilabel neural networks with applications to functional genomics and text categorization. *IEEE transactions on Knowledge and Data Engineering*, 18(10):1338–1351, 2006.

[148] Min-Ling Zhang and Zhi-Hua Zhou. Ml-knn: A lazy learning approach to multi-label learning. *Pattern recognition*, 40(7):2038–2048, 2007.

[149] Min-Ling Zhang and Zhi-Hua Zhou. A review on multi-label learning algorithms. *IEEE transactions on knowledge and data engineering*, 26(8):1819–1837, 2014.

[150] Tong Zhang and Frank J Oles. Text categorization based on regularized linear classification methods. *Information retrieval*, 4(1):5–31, 2001.

[151] Zhenyue Zhang and Keke Zhao. Low-rank matrix approximation with manifold regularization. *IEEE transactions on pattern analysis and machine intelligence*, 35(7):1717–1729, 2013.

[152] Erheng Zhong, Wei Fan, and Qiang Yang. Contextual collaborative filtering via hierarchical matrix factorization. In *SDM*, pages 744–755. SIAM, 2012.

[153] Mingyuan Zhou, Chunping Wang, Minhua Chen, John Paisley, David Dunson, and Lawrence Carin. Nonparametric bayesian matrix completion. *Proc. IEEE SAM*, 2010.

[154] Yang Zhou and Ling Liu. Activity-edge centric multi-label classification for mining heterogeneous information networks. In *Proceedings of the 20th ACM SIGKDD international conference on Knowledge discovery and data mining*, pages 1276–1285. ACM, 2014.

[155] Shenghuo Zhu, Xiang Ji, Wei Xu, and Yihong Gong. Multi-labelled classification using maximum entropy method. In *Proceedings of the 28th annual international ACM SIGIR conference on Research and development in information retrieval*, pages 274–281. ACM, 2005.

Collaborative Filtering and Multi-Label Classification with Matrix Factorization

PRIMARY SOURCES

1 Vikas Kumar, Arun K. Pujari, Vineet Padmanabhan, Sandeep Kumar Sahu, Venkateswara Rao Kagita. "Multi-Label Classification Using Hierarchical Embedding", Expert Systems with Applications, 2017
Publication

18%

2 Vikas Kumar, Arun K. Pujari, Sandeep Kumar Sahu, Venkateswara Rao Kagita, Vineet Padmanabhan. "Collaborative filtering using multiple binary maximum margin matrix factorizations", Information Sciences, 2017
Publication

7%

3 Vikas Kumar, Arun K. Pujari, Sandeep Kumar Sahu, Venkateswara Rao Kagita, Vineet Padmanabhan. "Proximal maximum margin matrix factorization for collaborative filtering", Pattern Recognition Letters, 2017
Publication

5%

4 Kumar, Vikas, Arun K. Pujari, Sandeep Kumar Sahu, Venkateswara Rao Kagita, and Vineet

1%