Regularized models and algorithms for machine learning

Chenyang Shen
*Hong Kong Baptist University*

Follow this and additional works at: [https://repository.hkbu.edu.hk/etd_oa](https://repository.hkbu.edu.hk/etd_oa)

**Recommended Citation**

Shen, Chenyang, "Regularized models and algorithms for machine learning" (2015). *Open Access Theses and Dissertations*. 195. [https://repository.hkbu.edu.hk/etd_oa/195](https://repository.hkbu.edu.hk/etd_oa/195)

This Thesis is brought to you for free and open access by the Electronic Theses and Dissertations at HKBU Institutional Repository. It has been accepted for inclusion in Open Access Theses and Dissertations by an authorized administrator of HKBU Institutional Repository. For more information, please contact repository@hkbu.edu.hk.
Regularized Models and Algorithms for Machine Learning

SHEN Chenyang

A thesis submitted in partial fulfillment of the requirements
for the degree of
Doctor of Philosophy

Principal Supervisor: Prof. NG, Michael Kwok-Po

Hong Kong Baptist University

August 2015
DECLARATION

I hereby declare that this thesis represents my own work which has been done after registration for the degree of PhD at Hong Kong Baptist University, and has not been previously included in a thesis, dissertation submitted to this or other institution for a degree, diploma or other qualification.

Signature: ______________________

Date: August 2015
ABSTRACT

Multi-label learning (ML), multi-instance multi-label learning (MIML), large network learning and random under-sampling system are four active research topics in machine learning which have been studied intensively recently. So far, there are still a lot of open problems to be figured out in these topics which attract worldwide attention of researchers. This thesis mainly focuses on several novel methods designed for these research tasks respectively.

The main difference between ML learning and traditional classification task is that in ML learning, one object can be characterized by several different labels (or classes). One important observation is that the labels received by similar objects in ML data are usually highly correlated with each other. In order to explore this correlation of labels between objects which might be a key issue in ML learning, we consider to require the resulting label indicator to be low rank. In the proposed model, nuclear norm which is a famous convex relaxation of intractable matrix rank is introduced to label indicator in order to exploiting the underlying correlation in label domain. Motivated by the idea of spectral clustering, we also incorporate information from feature domain by constructing a graph among objects based on their features. Then with partial label information available, we integrate them together into a convex low rank based model designed for ML learning. The proposed model can be solved efficiently by using alternating direction method of multiplier (ADMM). We test the performance on several benchmark ML data sets and make comparisons with the state-of-art algorithms. The classification results demonstrate the efficiency and effectiveness of the proposed low rank based methods.

One step further, we consider MIML learning problem which is usually more complicated than ML learning; besides the possibility of having multiple labels, each object can be described by multiple instances simultaneously which may significantly increase the size of data. To handle the MIML learning problem we first propose and develop a novel sparsity-based MIML learning algorithm. Our idea is to formulate and construct a transductive objective function for label indicator to be learned by using the method of random walk with restart that exploits the relationships among instances and labels of objects, and computes the affinities among the objects. Then sparsity can be introduced in the labels indicator of the objective function such that relevant and irrelevant objects with respect to a given class can be distinguished. The resulting sparsity-based MIML model can be given as a constrained convex optimization problem, and it can be solved very efficiently by using the augmented Lagrangian method (ALM). Experimental results on benchmark data have shown that the proposed sparse-MIML algorithm is computationally efficient, and effective in label prediction for MIML data. We demonstrate that the performance of the proposed method is better than the other testing MIML learning algorithms.

Moreover, one big concern of an MIML learning algorithm is computational efficiency, especially when figuring out classification problem for large data sets. Most of the existing methods for solving MIML problems in literature may take a long computational time and have a huge storage cost for large MIML data sets. In this thesis, our main aim is to propose and develop an efficient Markov Chain based learning algorithm for MIML problems. Our idea is to perform labels classification among objects
and features identification iteratively through two Markov chains constructed by using objects and features respectively. The classification of objects can be obtained by using labels propagation via training data in the iterative method. Because it is not necessary to compute and store a huge affinity matrix among objects/instances, both the storage and computational time can be reduced significantly. For instance, when we handle MIML image data set of 10000 objects and 250000 instances, the proposed algorithm takes about 71 seconds. Also experimental results on some benchmark data sets are reported to illustrate the effectiveness of the proposed method in one-error, ranking loss, coverage and average precision, and show that it is competitive with the other methods.

In addition, we consider the module identification from large biological networks. Nowadays, the interactions among different genes, proteins and other small molecules are becoming more and more significant and have been studied intensively. One general way that helps people understand these interactions is to analyze networks constructed from genes/proteins. In particular, module structure as a common property of most biological networks has drawn much attention of researchers from different fields. However, biological networks might be corrupted by noise in the data which often lead to the miss-identification of module structure. Besides, some edges in network might be removed (or some nodes might be miss-connected) when improper parameters are selected which may also affect the module identified significantly. In conclusion, the module identification results are sensitive to noise as well as parameter selection of network. In this thesis, we consider employing multiple networks for consistent module detection in order to reduce the effect of noise and parameter settings. Instead of studying different networks separately, our idea is to combine multiple networks together by building them into tensor structure data. Then given any node as prior label information, tensor-based Markov chains are constructed iteratively for identification of the modules shared by the multiple networks. In addition, the proposed tensor-based Markov chain algorithm is capable of simultaneously evaluating the contribution from each network. It would be useful to measure the consistency of modules in the multiple networks. In the experiments, we test our method on two groups of gene co-expression networks from human beings. We also validate biological meaning of modules identified by the proposed method.

Finally, we introduce random under-sampling techniques with application to X-ray computed tomography (CT). Under-sampling techniques are realized to be powerful tools of reducing the scale of problem especially for large data analysis. However, information loss seems to be un-avoidable which inspires different under-sampling strategies for preserving more useful information. Here we focus on under-sampling for the real-world CT reconstruction problem. The main motivation is to reduce the total radiation dose delivered to patient which has arisen significant clinical concern for CT imaging. We compare two popular regular CT under-sampling strategies with ray random under-sampling. The results support the conclusion that random under-sampling always outperforms regular ones especially for the high down-sampling ratio cases. Moreover, based on the random ray under-sampling strategy, we propose a novel scatter removal method which further improves performance of ray random under-sampling in CT reconstruction.
ACKNOWLEDGEMENTS

Firstly, I would like to express my deep gratitude to my principal supervisor Prof. Ng, Michael Kwok-Po for all of his guidance, patience and encouragement in the last five years. I am really grateful for every opportunity he has offered me. It has been a great privilege to work with him and I will treasure all everything I have learned from him for my whole life.

Also, I would like to thank my co-supervisor Prof. Yuan Xiaoming for his helpful advises on both my study and research.

Moreover, I would like to express sincere appreciation to Dr. Jia Xun of the University of Texas, Southwestern Medical Center for his kindly support and guidance during my time there as an exchange student.

In addition, I have been grateful to the help I have received from all my colleagues and friends, especially from Mr. Chen Chuan, Dr. Chi Yujie, Mr. Ke Xiao, Dr. Li Xinxin, Dr. Liu Yang, Ms. Mo Lili, Dr. Qin Nan, Dr. Tam Chi-Pan, Mr. Tao Jinming, Dr. Wang Wei, Dr. Wu Qiong, Mr. Wu Xiaomeng, Dr. Yau Chin Ko and Mr. Zhao Chen. Without them, I would never finish composing this thesis.

Finally, my deepest thanks would go to my beloved family for their endless love, support and consideration. They have always been my source of strength which will encourage me forever.
# Table of Contents

Declaration i

Abstract i

Acknowledgements iii

Table of Contents iv

List of Figures vii

List of Tables x

Chapter 1  Overview of the Thesis 1

1.1 Overview of Multi-label Learning 1

1.2 Overview of Multi-instance Multi-label Learning 2

1.3 Overview of Fast Markov Chain Based Algorithm 3

1.4 Overview of Module Structure Learning 4

1.5 Overview of Random Under-sampling in Medical Imaging 6

1.6 Outline of Thesis 8

Chapter 2  Multi-label Learning 9

2.1 Brief Review on Existing Multi-label Learning Algorithms 9

2.2 Low-rank based graph method for Multi-label Learning 13

2.2.1 Label Prediction by Using Low-rank Property 13

2.2.2 Relaxed Spectral Clustering for Multi-label Learning 15

2.2.3 The Combined Model and its Algorithm 17

2.3 Experimental Results 19
<table>
<thead>
<tr>
<th>Section</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>2.3.1 Datasets</td>
<td>19</td>
</tr>
<tr>
<td>2.3.2 Methodology</td>
<td>20</td>
</tr>
<tr>
<td>2.3.3 Results and Discussion</td>
<td>22</td>
</tr>
<tr>
<td>2.4 Concluding Remarks on LRML Learning Method</td>
<td>25</td>
</tr>
<tr>
<td>Chapter 3 Multi-instance Multi-label Learning</td>
<td>27</td>
</tr>
<tr>
<td>3.1 Brief Review on Multi-instance Multi-label Learning Algorithms</td>
<td>28</td>
</tr>
<tr>
<td>3.2 A Sparsity-Based Multi-instance Multi-label Learning Algorithm</td>
<td>29</td>
</tr>
<tr>
<td>3.2.1 The Sparse Model</td>
<td>33</td>
</tr>
<tr>
<td>3.2.2 The Proposed Algorithm</td>
<td>34</td>
</tr>
<tr>
<td>3.3 Experimental Results</td>
<td>36</td>
</tr>
<tr>
<td>3.3.1 Classification Performance</td>
<td>37</td>
</tr>
<tr>
<td>3.3.2 Effect of Parameters</td>
<td>42</td>
</tr>
<tr>
<td>3.4 Concluding Remarks on Sparse-MIML Algorithm</td>
<td>42</td>
</tr>
<tr>
<td>Chapter 4 The Fast Markov-chain based Learning Method</td>
<td>46</td>
</tr>
<tr>
<td>4.1 A Fast Markov Chain Based Algorithm for Multi-instance Multi-label</td>
<td>46</td>
</tr>
<tr>
<td>4.1.1 The Markov-MIML Algorithm</td>
<td>47</td>
</tr>
<tr>
<td>4.1.2 Two Markov Chains</td>
<td>48</td>
</tr>
<tr>
<td>4.1.3 Optimization Model</td>
<td>51</td>
</tr>
<tr>
<td>4.1.4 Comparison of Computational Cost</td>
<td>52</td>
</tr>
<tr>
<td>4.2 Experimental Results</td>
<td>54</td>
</tr>
<tr>
<td>4.2.1 Experiment 1</td>
<td>55</td>
</tr>
<tr>
<td>4.2.2 Experiment 2</td>
<td>57</td>
</tr>
<tr>
<td>4.2.3 Experiment 3</td>
<td>64</td>
</tr>
<tr>
<td>4.2.4 Experiment 4</td>
<td>67</td>
</tr>
<tr>
<td>4.3 Concluding Remarks on OF-MIML Algorithm</td>
<td>70</td>
</tr>
<tr>
<td>Chapter 5 Application to Module Structure Learning</td>
<td>72</td>
</tr>
<tr>
<td>5.1 Multiple Networks Modules Identification by a Multi-dimensional Markov Chain Method</td>
<td>72</td>
</tr>
<tr>
<td>5.1.1 Multidimensional Markov chains</td>
<td>74</td>
</tr>
</tbody>
</table>
List of Figures

2.1 Comparison of six methods under varying the labeled data sizes on Corel5K and Mediamill. ........................................ 24
2.2 Comparison results under varying the ratio of missing entries in label matrix (Y) on CAL500 and MSRC. ........................ 25
3.1 Sparsity of MIML image data (left) and MIML text data (right). A red (blue) pixel refers that an object belongs (does not belong) to a class. ......................................................... 31
3.2 The performance of sparse-MIML with 10% training image data for different values $\alpha$ and $\lambda$. .............................. 43
3.3 The performance of sparse-MIML with 90% training image data for different values $\alpha$ and $\lambda$. .............................. 44
3.4 The performance of sparse-MIML with 90% training text data for different values $\alpha$ and $\lambda$. .............................. 45
4.1 The classification performance of different algorithms with respect to the percentage of training samples. .......................... 58
4.2 The histogram of features for 7 different classes. .................. 59
4.3 Test 3: the classification performance of different algorithms with respect to the percentage of training samples for text data set containing 3000 bags. ........................................... 62
4.4 The classification performance of different algorithms with respect to the percentage of training samples for text data set containing 4000 bags. ........................................... 63
4.5 Test 3: the computational time as well as the classification performance of different algorithms with respect to data sets of different sizes.
4.6 Test 3: the classification performance of different algorithms with respect to data sets of different sizes.
4.7 Test 3: the classification performance of different algorithms with respect to the percentage of training samples for 5-class data.
4.8 Test 3: the classification performance of different algorithms with respect to the percentage of training samples for 10-class data.

5.1 The subgraph formed by first 10 vertices in the 3-rd network synthetic data.
5.2 Value of $x$ with the 1-st vertex as prior for synthetic data.
5.3 Value of $x$ with gene AFFX-r2-Ec-bioB-M at as prior information for example 1 in the three gene co-expression networks of cancers.
5.4 Value of $x$ with gene 202708-s-at as prior information for example 2 in the three gene co-expression networks of cancers.
5.5 Subgraphs constructed by 12 genes selected for example 2 in the three gene co-expression networks of cancers, left: subgraph in GBM cancer network; middle: subgraph in LUCS cancer network; right: subgraph in OV cancer network.
5.6 Value of $x$ with gene 200606-at as prior information for example 3 in the three gene co-expression networks of cancers.
5.7 Subgraphs constructed by 12 genes selected for example 3 in the three gene co-expression networks of cancers, left: subgraph in GBM cancer network; middle: subgraph in LUCS cancer network; right: subgraph in OV cancer network.
5.8 Value of $x$ with gene SAA1 as prior information for example 1 in the three gene co-expression networks of different tissues of morbidly obese patients.
5.9 Value of $x$ with gene ALAS2 and HBA1 as prior information for example 2 in the three gene co-expression networks of different tissues of morbidly obese patients.
6.1 Comparison of rank ................................. 107
6.2 Comparison of singular values ......................... 108
6.3 Comparison of right singular vectors ................. 110
6.4 Effect of interpolation for scattering signals: original scatter (left),
    blocked scatter (middle) and interpolated scatter (right) ....... 111
6.5 Reconstruction results where 25% of projection data preserved: full
    data (a), scattered full data (b), regular ray under-sampling (c), reg-
    ular view under-sampling (d) and random ray under-sampling (e)... 112
6.6 Reconstruction results where 16.7% of projection data preserved: reg-
    ular ray under-sampling (a), regular view under-sampling (b) and ran-
    dom ray under-sampling (c) .................................... 113
6.7 Reconstruction results where 10% of projection data preserved: regular
    ray under-sampling (a), regular view under-sampling (b) and random
    ray under-sampling (c) .................................... 114
## List of Tables

2.1 Multi-label data set summary. ............................................. 20

2.2 Comparison of multi-label learning performance output by six algorithms on six real world multimedia data sets. ....................... 21

3.1 The description of image and text data for MIML learning. .......... 38

3.2 Improvement by percentage of sparse-MIML method compared with MIML-RWR method on the text data. ............................... 40

3.3 Improvement by percentage of sparse-MIML method compared with MIML-RWR method on the image data. ............................... 40

3.4 Improvement by percentage of sparse-MIML method compared with MIML-RWR method with different sizes training data on the text data. 41

3.5 Improvement by percentage of sparse-MIML method compared with MIML-RWR method with different sizes training data on the image data. ......................................................... 41

4.1 The classification performance of different algorithms with 90% training data. ................................................................. 57

4.2 The computational time (seconds) of different algorithms. .......... 57

4.3 The classification performance of different algorithms for enlarged text data set containing 3000 bags with 90% training samples. .... 60

4.4 The classification performance of different algorithms for enlarged text data set containing 4000 bags with 90% training samples. .... 60

4.5 The computational time (seconds) of different algorithms for text data set containing 3000 bags. ........................................... 61
4.6 The computational time (seconds) of different algorithms for text data set containing 4000 bags. ........................................ 61
4.7 The classification performance of different algorithms for 5-class data with 90% training samples. ........................................ 65
4.8 The classification performance of different algorithms for 10-class data with 90% training samples. ........................................ 65
4.9 The computational time (seconds) of different algorithms for 5-class data. ................................................................. 66
4.10 The computational time (seconds) of different algorithms for 10-class data. ................................................................. 66
4.11 The classification performance of different algorithms for 20-class data with 80 bags in each class. ................................. 69
4.12 The classification performance of different algorithms for 20-class data with 160 bags in each class. ................................. 69
4.13 The performance of Object-Feature MIML with different sizes of training data on complete PascalVOC2007 data set. .......... 70

5.1 The 11 largest values in $x$ respect to the vertices for synthetic networks 84
5.2 Contribution scores in $y$ for synthetic data. ................................. 84
5.3 The 8 largest values in $x$ with their gene IDs for example 1 in the three gene co-expression networks of cancers. ................. 87
5.4 Contribution scores in $y$ of three cancers for example 1. ................. 88
5.5 The 15 largest values in $x$ with their gene IDs for example 2 in the three gene co-expression networks of cancers. ................. 89
5.6 Contribution scores in $y$ of three cancers for example 2. ................. 89
5.7 Gene ontology enrichment of the module identified for example 2 in the three gene co-expression networks of cancers. ................. 91
5.8 Contribution scores in $y$ of three cancers for example 3. ................. 92
5.9 The 5 largest values in $x$ with their gene IDs for example 1 in the three gene co-expression networks of different tissues of morbidly obese patients. ........................................ 94
5.10 Contribution scores of three tissues in $y$ for example 1 ................. 95
5.11 The 12 largest values in $x$ with their gene IDs for example 2 in the three gene co-expression networks of different tissues of morbidly obese patients. ........................................ 96

5.12 Contribution scores of three tissues in $y$ for example 2. .......................... 96

6.1 MSE compared with grand truth ......................................................... 115
Chapter 1

Overview of the Thesis

In this thesis, we mainly focus on machine learning techniques, in particular multi-label (ML) learning, multi-instance multi-label (MIML) learning, module structure learning and random under-sampling methods with their applications to image and text classification, gene co-expression network analysis as well as medical imaging.

1.1 Overview of Multi-label Learning

The classical learning task requires one to categorize object with unknown label into one of the possible classes. Indeed, this is single-label (SL) learning problem which is modeled based on the assumption that an object only belongs to one class. However, this assumption may not hold in numerous real applications. Practically, an object is possible to be assigned by multiple labels simultaneously in many different areas. For instance, in image classification, an image containing both tree and buildings should be assigned with both tips (labels) ’natural’ and ’artificial’; several different topics (labels) can be discussed in one single article for text mining; and many genes usually correspond to multiple biological responses (labels) in gene function analysis. Therefore, so called multi-label (ML) learning algorithm seems to be necessary for these data analysis tasks. ML learning algorithm tackles the case that each object possibly belongs to more than one categories. Generally speaking, ML learning problem should be more suitable for handling practical problems and, off course more complicated than classical single-label learning problem.
The chapter 2 of this thesis will mainly concentrate on ML learning problem. To provide an overview of ML learning problem firstly, several state-of-art approaches for ML learning are briefly reviewed. Then we move onto details of the proposed approach. In order to tackling the ML learning problem, we design to combine low rank based methods and spectral clustering techniques to formulate a convex optimization problem, such that ML classification can be accomplished by iteratively seeking for optimal solutions of the proposed optimization model. Forcing low rank property to label indicator of objects helps to to exploit and utilize underlying correlation in label domain which may improve prediction results for unlabeled objects while performing spectral clustering on the graph constructed by measuring similarity among objects helps to incorporate the feature domain for label prediction. In the experimental study, we test the proposed algorithm on several bench mark data sets with comparisons with state-of-art methods are made. The classification results support that the proposed method outperforms the others.

1.2 Overview of Multi-instance Multi-label Learning

Consider the ML learning problem mentioned in previous section, each object may be assigned to multiple classes. Further on, if each object is described by several different instances simultaneously, the problem becomes multi-instance multi-label (MIML) problem. Compared to ML problem, the MIML problem further increases the difficulty of classification, however more appropriate to formulate classification of complicated objects. Recover in ML learning, an image contains multiple substances (e.g. an image contain both human and dog) should be classified into multiple classes (i.e. belongs to label of 'human' and 'animals' at the same time). More precisely, the image should consist of patches (at least a patch containing human and another patch containing dog) which can be categorized by different labels ('human' and 'animals'). Therefore, it might be more rational to represent an image (object) by multiple patches (instances). Similarly, an article which goes through several different topics (labels) should consist of paragraphs focus on different topics respectively. Thus it
is reasonable that each article (object) should be described by multiple paragraphs (instances). It is apparently that MIML learning problem is even more complicated compared to ML learning while at the same time, the size of data can be larger.

In chapter 3, detailed MIML learning problem and the proposed sparsity based MIML learning methods are addressed. After a brief review on existing approaches in literature, the sparsity based Markov chain model is developed for MIML learning which is motivated by the observation that label indicator of MIML problem is often sparse matrix. More specifically, we first study relationship between each object by Markov chain process which is formulated by a normalized affinity matrix among object transformed from a affinity matrix computed among instance level. Then a transductive objective function for label indicator is formulated to exploit the relationship between objects and label which contributes to label prediction incorporating with partial training label information. More importantly, the label indicator is constrained to be sparse matrix base on the observation mentioned. It turns out that the proposed method constructs a convex optimization problem for MIML learning task which is capable of classifying unlabeled objects more accurately. Numerical results on two bench mark MIML data sets illustrate the effectiveness and efficiency of the proposed method compared with state-of-art MIML learning algorithms.

### 1.3 Overview of Fast Markov Chain Based Algorithm

As mentioned in previous section, MIML learning problem usually involves large size of data. So far most of the existing algorithms fail to handle large MIML data due to both computational and storage limits and failure in dealing with large MIML data has become a bottle neck of MIML learning problem. Therefore, one main issue in the design of MIML algorithm nowadays is how to handle a large MIML data set effectively and efficiently such that both storage and computational cost can be reduced in the learning process. One of the main contributions of this thesis reported in chapter 4 is proposing an Object-Feature MIML learning algorithm to evaluate the importance of relationships between a set of labels associated with objects of multiple instances.
efficiently. The proposed algorithm computes the ranking score of importance of different labels (label-object ranking) as labeling indicator for each object. Instead of the construction of affinity matrix among objects, we make use of instance-feature matrix (the input MIML data) to construct to two Markov chains. One Markov chain is to determine the transition probabilities from features to instances, and the other Markov chain is to determine the transition probabilities from instances to features. The construction cost of the two Markov chains is significantly less than that of the MIML-Markov method [78]. By making use of the object-instance relation matrix, we can design a simple iterative scheme to govern ranking score of labels associated with objects and relevance score of features in the MIML problem. With label information from labeled objects, all objects then spread their label ranking scores to their neighbors based on the two Markov chains in a random walk. The spread process is repeated until a global steady state is reached. Experimental results on two MIML text and image data sets are also listed in chapter 4 which have illustrated that the proposed method is computationally fantastically efficient while the classification performance is still competitive with the other state-of-art MIML algorithms.

1.4 Overview of Module Structure Learning

Genes/proteins function inference from complex biological systems has become a very important problem. In the past few years, it has been studied intensively by people from multiple fields. Among a large number of diverse research tasks for genes/proteins, one important aspect is to figure out the complex relations between genes/proteins. Biological networks representing the co-expression, regulation, interaction and so on seem to have become a general tool for computational genes/proteins relation analysis. More specifically, the network is constructed by making use of genes (or proteins) as nodes while the edges as connections. By analyzing such genes (or proteins) network, it is possible for researchers to identify and interpret relations between genes (or proteins) such that some important phenomena in a complex biological systems can be inferred correspondingly, see [2, 5, 12–14, 23, 42, 63, 64, 74]. Among a large number of approaches, one substantial way in network analysis is to
explore a group of similar nodes which are closely related with each other. These nodes combining together form a densely connected subgraph which is usually called module. The module structure is common in biological networks and it is very useful especially in the study on function of genes and proteins in some complex biological systems. For instance, the large size of genes (or proteins) network can be reduced by dividing the network into different modules and thus much easier to handle. Once a module is identified, we may infer the function of unknown genes by other genes which we know well since the genes/proteins in the same module are more likely to play similar roles. In the literature, there are a lot of works accomplished for identifying the functional module for a single graph [16,21,24,27,52,53,59,60,85,86]. However, biological networks constructed by real data are often corrupted by noise occurring in data: some edges which should exist are removed due to the noise while in other place, some extra edges are introduced into the networks. If so, the modules are possibly to be miss-identified. Moreover, the selection of different thresholding values during the process of constructing networks may introduce the inconsistency of module structure. One intuitive idea to reduce the effect of both noise and parameters is integrating multiple networks composed of the same set of nodes, see [30–32,41,43,50].

In the chapter 5 of this thesis, we mainly consider the module identification problem based on multiple networks at the same time. Our main idea is to formulate multiple networks into order-three tensor for identifying common module structure embedded in these networks. The numerical algorithm is mainly motivated by [78]. The main difficulty lies in is that instead of formulating the problem under the matrix framework, we target at algorithm that can handle the tensor data. To overcome this difficulty, a novel two-stage iterative scheme is proposed. As preparation for the algorithm, two probability transition tensors should be generated. One of them should be normalized along the direction corresponding to nodes while the other is normalized with respect to different networks. Then in the first stage, we may multiply two probability transition tensors by the initial guess of label indicator vector such that two probability transition matrices are generated with one dimension of them corresponding to nodes and the other referring to multiple networks. Then in the second stage, by fixing these two matrices, we are able to form two Markov
chains: one is to determine the transition probabilities from multiple networks to nodes incorporating with prior information and the other computes the transition probabilities from nodes to different networks. After convergence is achieved in the second stage, we may make use of the converged label indicator for module to update the two transition probability matrices. Repeating the two-stage process until global steady state is reached, we identify the module structure in the label indicator and also, we are able to tell the contributions of different networks as well as the consistency of the modules across multiple networks. Later we illustrate the effect of the proposed tensor-based module identification method by the experimental results on two different gene data sets. It is easy to tell from the results that our proposed method can identify the valid consistent module structures in multiple networks.

1.5 Overview of Random Under-sampling in Medical Imaging

X-ray computed tomography (CT) as volumetric image guidance is broadly used in clinics for multiple targets. However, the high X-ray radiation dose to patients in CT scans often arise clinical concerns [34,39,40,49,51,56,69,77]. Under-sampling strategy which is commonly applied in machine learning might be useful in dose reduction for CT imaging. Recently, reconstruction from under-sampled data becomes an active research topic due to the desire of reducing dose [11,35,67,82]. As a straightforward under-sampling strategy, angular under-sampling has been widely employed and the reconstruction problem has been extensively studied. This is because CT scan is conventionally performed in a view-by-view fashion and angular under-sampling is expected to be easily achievable. In contrast, other type of under-sampling strategies have been less explored. This includes ray under-sampling within each view. In addition, within each view, one could also under-sample in a regular or random fashion. Compared to angular under-sampling, the authors believe that under-sampling strategies in each view should be more favorable mainly because these ray under-sampling methods provide us possibility of removing large amount of scattered photons which may cause largely decay in image quality of CT. As random or regular beam blocker
should place before the beam source which blocks each projection in order to achieve the ray down-sampling, only scatter signal should be measured in the deliberately created shadows, which allows us to further interpolate the signal to the entire plane base on smoothness assumption of scattered photons. Our main idea is to remove the interpolated scatter from the total signal in the blocked area, such that cleaned data is used for CT reconstruction. When ray under-sampling occurs in CT reconstruction task, conventional FDK type reconstruction [26] failed in this regime. A common approach is to use a iterative reconstruction method that solves an optimization problem. A data fidelity term, typically least square form, guarantees the resulting CT image to be reconstructed based on the under-sampling projection data while regularization term is also needed to remove artifacts caused by under-sampling based on a prior knowledge about the image properties. A widely used prior knowledge is based on the assumption that images of good quality should be piece-wise smooth while at the same time; sharp edges in image should be preserved. Among many possible approaches in literature, total variation (TV) [35, 68, 75] and tight frame [9,20,36] are two most successfully applied methods. In this work, we propose to employ the tight frame based model for under-sampling CT reconstruction task due to superior performance in enhancing image quality.

Later in chapter 6 of this thesis, we would like to first compare these three under-sampling strategies by analyzing numerical property of projection operators when solving the corresponding least square problem. Motivated by the results in [45, 46], in which the authors provide an efficient approach for characterizing singular value decomposition (SVD) of full projection operator, we propose a novel approach to perform SVD for projection operators respect to different under-sampling strategies. It is possible to tell the difference based on comparisons on both singular values and singular vectors between different under-sampling and the full projection operators. The numerical results may explain the reason that the strategy of random under-sampling in each view usually outperforms the others. Then in the Monte Carlo (MC) simulation study, we demonstrate the outstanding performance of ray random under-sampling method by comparing constructed image quality with both angular under-sampling as well as ray regular under-sampling. We also propose a novel scatter
removal algorithm and illustrate its effectiveness in the simulation study.

1.6 Outline of Thesis

This rest of the thesis will be organized as follows:

In chapter 2, ML learning with state-of-art ML algorithms is introduced followed by the proposed low rank based ML learning method. Experimental results are also reported to illustrate the classification performance of the proposed method.

Then we move on to introduce the MIML learning problem in chapter 3. Similarly, brief review of existing algorithm is listed before going to details of the proposed sparsity based Markov chain method. Then we provide numerical results of the tested MIML algorithms as demonstration for the sparsity based Markov chain method.

Chapter 4 is mainly based on the fast MIML algorithm designed for large MIML data. We derive the proposed fast MIML model as well as algorithm and make comparison with state-of-art MIML algorithms on both synthetic and real data sets.

Module structure learning problem and method are both introduced in chapter 5. In addition, theoretical convergence analysis of the proposed tensor based algorithm is given before we report the experimental results of the proposed method on two bench mark gene data sets.

The main goal of chapter 6 is to illustrate the superiority of random under-sampling with its application to medical imaging. We perform simulation study in order to illustrating the advantages of random under-sampling system. Also, reconstruction results of all the three considered under-sampling strategies are available with detail information of the proposed scatter removal algorithm in chapter 6.

Finally, conclusion and future work are both addressed in chapter 7.
Chapter 2

Multi-label Learning

It is usually more reasonable to formulate real problems such as text mining and image classification into multi-label (ML) learning problem due to the reason that each object (article or image) usually belongs to several different classes simultaneously. ML learning requires algorithm that is capable of categorizing any unlabeled object into multiple categories when most of the traditional classification algorithms fail to. Thus it is necessary to develop model and algorithm which target on tackling ML learning. In this chapter, we first briefly introduce several existing ML learning methods and then provide detail information of the proposed approach that develops for exploring and utilizing correlation of labels among objects. Finally numerical results are reported with comparisons between state-of-art ML learning methods followed by concluding remarks.

2.1 Brief Review on Existing Multi-label Learning Algorithms

In the literature, many algorithms have been designed recently to handle ML learning problem [6, 10, 44, 70, 83]. Let’s first introduce some preliminary notations before we start reviewing.

For the ML problem, assume that we have in total $N = N_1 + N_2$ objects (or data points). Given $N_1$ objects $\{(x_{it}, y_i)\}^{N_1}_{i=1}$, where $x_{it} \in \mathbb{R}^F$ and $y_i \in \mathbb{R}^K$. Then, let $X^{tr} = [x_1^{tr}, x_2^{tr}, \cdots, x_{N_1}^{tr}]^T$ and $Y^{tr} = [y_1, y_2, \cdots, y_{N_1}]^T$ to be the feature and label
matrices for training data respectively.

Similarly, with the number of testing data to be \( N_2 \), \( X^{tt} \in \mathbb{R}^{N_2 \times F} \) and \( Y^{tt} \in \mathbb{R}^{N_2 \times K} \) are used to denote the both feature and label matrices.

In addition, we use \( X \) and \( Y \) to denote the feature and label information for all data respectively, i.e.:

\[
X = \begin{bmatrix} X^{tr} \\ X^{tt} \end{bmatrix}, \quad Y = \begin{bmatrix} Y^{tr} \\ Y^{tt} \end{bmatrix}.
\]

**PLST** Tai and Lin proposed a principle label space transformation method (PLST) [70] by making use of the label information \( Y \). PLST is based on the hypercube sparsity assumption that the number of training data \( N_1 \) is much less than the number of vertices \( (2^K) \) of hypercube which is built via \( K \) labels (i.e., the cardinality of power set of \( K \) labels). The main idea is to learn subspace with much lower dimension compared with the original feature space. More precisely, the encoding factor \( P \) for PLST method can be extracted by solving the following optimization problem:

\[
\min_P \|Y^{tr}^T - PP^TY^{tt}^T\|_F^2 \quad \text{s.t.} \quad P^TP = I \quad (2.1)
\]

The optimal solution \( P \in \mathbb{R}^{M \times K} (M \ll K) \) of problem (2.1) is used to transform the original features to its low-dimensional subspace: \( h = Y^{tt}P^T \). Moreover, an prediction function \( r : \mathbb{R}^F \mapsto \mathbb{R}^M \) should also be learned using training data. Finally, the label of any testing sample \( x^{tt}_i \) can be predicted by \( r(x^{tt}_i)P \).

**CPLST** Note that PLST method only considers information in label domain, later, Chen and Lin [10] propose to combine both data information \( X \) and label information \( Y \) and presented a constrained PLST (CPLST) model to extract latent space for multi-label learning.

Inspired by the orthogonally constrained canonical correspondence analysis (CCA) [8], the feature information is incorporated in the following optimization problem:

\[
\min_{W_x, P} \|X^{tt}W_x - Y^{tr}P\|_F^2 \quad \text{s.t.} \quad P^TP = I \quad (2.2)
\]

When fixing \( P \), the optimal \( W_x \) can be computed by a closed form solution \( W_x = X^{tt\dagger}Y^{tr}P \), where \( X^{tt\dagger} \) is the pseudo inverse of \( X \).
Then CPLST combines (2.2) and (2.1) and the label prediction of CPLST basically follow the same procedures in PLST.

**FAIE** Both PLST and CPLST explicitly identify the encoding factor $P$ for mapping the label space ($K$-dimensional space) to latent space ($M$-dimensional space). As mentioned in [44], the optimal mapping may be complicated and even indescribable. Thus, Lin et al. [44] proposed a new feature-aware implicit label space encoding method (FAIE), which jointly maximizes the recoverability of the original label space and the predictability of the latent space from the original feature space:

$$\max_{C,D} \phi(Y^{tr}, C, D) + \varphi(X^{tr}, C)$$

where the function $\phi(Y^{tr}, C, D)$ targets on the recoverability and $\varphi(X^{tr}, C)$ is set for the predictability. In addition, the optimal $D$ can be derived as a closed-form expression with $D = (C^T C)^{-1} C^T Y^{tr}$ in the optimization problem. Thus the key issue for FAIE in learning the latent space with both recoverability and predictability is solving optimal $C$.

**LEML** The above models which are all tied to the $L_2$ loss function cannot directly handle missing labels as it requires SVD on fully observed $Y^{tr}$. Yu et al. [83] proposed an empirical risk minimization model for multi-label learning (LEML), which has ability to deal with data with missing labels via the following models,

$$\min_{W, P} \sum_{(i,j) \in \Omega} \ell(Y_{ij}^{tr}, X_{i}^{tr} W P_j^T) + \lambda \left( \|W\|_F^2 + \|P\|_F^2 \right)$$

(2.3)

where $\Omega \subseteq [n] \times [K]$ for $n \leq N_1$ is the index set that represents 'known' labels and $W \in R^{F \times L}$, $P \in R^{K \times L}$. Any loss function can be used for $\ell$ such as squared loss, logistic loss, squared hinge loss and so on.

**MC** The matrix completion method (MC) proposed in [6] targets on the weakly-supervised multi-label image classification. Instead of consider feature and label information separately, they construct a matrix $Z_{obs}$ as follows:

$$Z_{obs} = \begin{bmatrix} Y^{trT} & Y^{ttT} \\ X^{trT} & X^{ttT} \end{bmatrix} = \begin{bmatrix} Y^T \\ X^T \end{bmatrix}^T 1^n.$$

Due to the reason that $Y^{tt} = 0$ (label information of testing data is lost), the author propose to predict $Y^{tt}$ by employing the technique of matrix completion. The model
is built based on the assumption that matrix \( Z \) should low-rank or approximately low-rank and they aim at minimize the rank of matrix \( Z \) with two loss functions \( l_X(E_X) \) and \( l_Y(E_Y^\Omega) \) for errors in feature domain and label domain respectively:

\[
\min_{Z, E_X, E_Y^\Omega} \mu \|Z\|_* + l_X(E_X, X^\Omega) + \lambda l_Y(E_Y)
\]

\[
s.t. \quad Z = \begin{bmatrix} X^T & 0 \\ Y^T & E_X \\ 1^T & 0 \end{bmatrix}
\]

\( \| \star \|_* \) indicates the famous nuclear norm which has been studied intensively in literature, see [7, 61]. The nuclear norm \( \|Z\|_* \) here is a convex approximation for the untractable function \( \text{rank}(Z) \). We remark that MC algorithm can also work in the situation when there are missing labels in training data.

**SLRM** Recently, Yang et al. [38] have designed a low rank based mapping model for ML learning:

\[
\min_U \frac{1}{2} \|UX_\Omega - Y_\Omega\|_F^2 + \lambda \|U\|_* + \frac{\gamma}{2} \text{tr}((UX)^T L(UX)),
\]

which is aiming at seeking for an optimal mapping \( U \) from feature domain to label domain, such that one can determine the label of an object accurately based on its features. \( L \) indicates Laplacian matrix constructed by information in feature domain. \( X_\Omega \) and \( Y_\Omega \) denotes the feature and label indicator for labeled objects respectively. In this model, the correlation between objects is exploited by forcing low-rank property to mapping \( U \). This model is convex and can be solved efficiently by ADMM. Also SLRM is applicable when there are missing labels in training data.

Besides what we have mentioned, far more algorithms have been proposed for ML learning in literature. In the next chapter, the proposed model and algorithm are derived followed by numerical results and concluding remarks.
2.2 Low-rank based graph method for Multi-label Learning

In general, we believe that the similar objects should receive similar labels in classification problems. The labels should be highly correlated for those objects who are close to each other. This fact indicates the label indicator matrix for the objects should be low rank or approximately low-rank matrix. Our main idea comes to combine both low-rank property as well as information from feature domain together for label prediction of ML learning. More specifically, idea of spectral clustering [4, 47, 54] is utilized for exploring relationship among objects embedded in feature domain, while, partial known label information is employed as reference for label prediction. In addition, low rank property, on the other hand, is approximately achieved by minimizing nuclear norm for label indicator. All the missing labels are propagated based on the correlation to training label (low rank based method) in label domain as well as underlying structure embedded in feature domain (spectral clustering). The proposed model is formulated by a convex optimization problem for which ADMM schemes are designed to solve the model iteratively. We further remark here that it should be more efficient to solve the proposed model compared with MC method especially when dealing with large data. For both models, the main computational cost is performing singular value decomposition (SVD) at each iteration for the nuclear norm involved in the minimization problem. In MC algorithm, it is required to compute SVD for a much larger matrix (size of matrix $Z_{obs}$) compared with the proposed model (size of matrix $Y$). What is more, our model is also capable of handling label correction when some labels are miss-assigned in training data.

Nextly, details of the proposed low-rank based method (LRML) for multi-label learning are introduced. Let us start by presenting the main idea of low-rank method.

2.2.1 Label Prediction by Using Low-rank Property

$Y \in \mathbb{R}^{N \times F}$ is the label matrix and $Y_{i,j} = 1$ if $i$-th object belongs to $j$-th class and $Y_{i,j} = 0$ if not. In addition, only the label information at position $\Omega$ is available.
Missing labels locate at the position \( \tilde{\Omega} = \{(i, j) | i \in S_1 \subset [1, 2, \ldots, N], j \in S_2 \subset [1, 2, \ldots, K]\}\), i.e. \( Y_{\tilde{\Omega}} = 0 \) (training set \( \subset S_1 \)). The purpose of ML learning is to predict these missing labels and get the true label matrix \( Y^* \). Due to the assumption that rows in \( Y^* \) should be highly correlated to each other, \( \text{rank}(Y^*) \ll \min N, K \).

Therefore, we may employ the following model for this ML problem:

\[
\begin{align*}
\min_Q \text{rank}(Q) \\
s.\text{t.} Q_{\Omega} &= Y_{\Omega}.
\end{align*}
\]  
(2.4)

This model (2.5) actually formulates the matrix completion problem which has been discussed comprehensively in [7]. \( \text{rank}(Q) \) here is numerically untractable. Therefore nuclear norm can be employed instead as approximation to \( \text{rank}(Q) \):

\[
\begin{align*}
\min_Q \|Q\|_* \\
s.\text{t.} Q_{\Omega} &= Y_{\Omega}.
\end{align*}
\]  
(2.5)

\( \|Q\|_* = \sum \sigma_i \) where \( \sigma_i \) corresponds to \( i \)-th singular value of matrix \( Q \). In [7], the model (2.5) is very nearly minimized iteratively using singular value thresholding (SVT) technique. More clearly, by introducing the dual variable \( W \),

\[
\begin{align*}
Q^k &= \mathcal{D}_\tau(W^{k-1}), \\
W^k &= Q^k + \delta_k(Y_{\Omega} - Q_{\Omega}).
\end{align*}
\]

Here \( \mathcal{D}_\tau(*) \) denotes the SVT operator. Let \( W = U\Sigma V^T \) to be SVD of matrix \( W \), then \( \mathcal{D}_\tau(U) = U(\Sigma - \tau I)_+ V^T \) with

\[
(a)_+ = \begin{cases} 
  a, & a \geq 0 \\
  0, & \text{otherwise.}
\end{cases}
\]

Convergence of the algorithm (2.6) can be guaranteed for a parameter sequence \( \delta_k \in [0, 2] \) in [7]. The optimal solution \( Q^* \) is a low-rank matrix where \( Q^*_{\tilde{\Omega}} \) should be predicted only based on the low-rank assumption of \( Y^* \).

If we relax the constraint to seek for a low rank approximation of \( Y^* \), we may further revise the model (2.5) as follows.

\[
\begin{align*}
\min_Q \frac{1}{2} \|Q - \hat{Y}\|_F^2 + \tau \|Q\|_*. 
\end{align*}
\]  
(2.6)
∥*∥_F indicates the Frobenius norm of matrix and \(\tau\) here controls the rank of solution. \(\hat{Y}_\Omega = Y_\Omega\) and \(\hat{Y}_\Omega = 0\). Compared with (2.5), (2.6) only requires \(Y_\Omega \approx Q_\Omega\) and more importantly, it can be solved directly by close form: \(D_\tau(Y)\). Therefore, it can be more efficient to only compute the low-rank approximation rather than strictly require \(Q_\Omega = Y_\Omega\). One advantage the relaxed model is that it does not require the labels in training domain of the predicted label to be exactly same to original ones, such that may have chance to correct miss-labeling occurs in training data.

In both of above models, only information in label domain is involved while relationship between each object in feature domain is ignored. Next we will introduce the idea of predicting label based on features of objects.

### 2.2.2 Relaxed Spectral Clustering for Multi-label Learning

Motivated by the successful application of spectral clustering methods in many different fields [4,47,54], we consider to remove the constraint in spectral clustering such that the relaxed model can work for label learning of ML problem.

More precisely, let \(A \in \mathbb{R}^{N \times N}\) to be the symmetric affinity matrix constructed by measuring the similarity between different objects: \(A_{i,j} = \theta(x_i, x_j)\) where \(\theta(*)\) can be any function for computing similarity score. Based on \(A\), we are able to construct a graph \(G\) by identifying similar objects:

\[
G_{i,j} = \begin{cases} 
1, & A_{i,j} > t; \\
0, & \text{otherwise}.
\end{cases}
\]

where \(t\) is a predefined value. Some other methods such \(k\)-nearest neighbor can be also used for graph construction. Note here \(G = G^T\) is also a symmetric matrix. Then the Laplacian matrix for \(G\) should be computed as \(L = D - G\) with \(D\) is a diagonal matrix and \(D_{i,i} = \sum_j G_{i,j}\). It is easy to evaluate that Laplacian matrix \(L\) is a positive semi-definite matrix. Then according to [47], the spectral clustering can be represented by the following optimization problem:

\[
\min_{\hat{Q}} \text{tr}(\hat{Q}^T L \hat{Q}) \\
\text{s.t. } \hat{Q}^T e = 0.
\]

(2.7)

where optimal \(\hat{Q}\) indicates the clustering results for all the data points.
In our case, we incorporate partial label information in $Y_{\Omega}$: further seek for $Q$ as the label indicator which minimizes (2.7) while simultaneously preserve label information in $Y_{\Omega}$. Therefore, combination of both feature and label information should be given by:

$$
\min_Q \text{tr}(Q^T L Q) \\
\text{s.t. } Q_{\Omega} = Y_{\Omega}.
$$

(2.8)

Note that we remove the constraint that requires $Q^T e = 0$. Further, let $f_\alpha(Q) = \alpha \text{tr}(Q^T L Q) + ||Q||_F^2$, it is clear to see if $\alpha \to \infty$, it is equivalent to minimize:

$$
\min_Q f_\alpha(Q) \\
\text{s.t. } Q_{\Omega} = Y_{\Omega}.
$$

(2.9)

**Theorem 1.** Set $U^0 = 0$, $\alpha > 0$ and $\delta_k \in (0, 2)$, Let

$$
\begin{align*}
Q^k &= (\alpha L + I)^{-1} U^{k-1}, \\
U^k &= Q^k + \delta_k(Y_{\Omega} - Q^k_{\Omega}).
\end{align*}
$$

The sequence of iterates above converges to the unique solution of (2.9).

**Proof.** As (2.8) is convex, let $Q^*$ be the global optimal solution of (2.8). It is easy to get $||Y_{\Omega} - Q^*_{\Omega}|| = 0$. Further, use $\partial f_\alpha = \{Q | f'_\alpha(Q) = 0\}$ to denote the subgradient of $f_\alpha$, we have $Q^* \in \partial f_\alpha$, that is:

$$
Q^* = (\alpha L + I)^{-1} Q^*.
$$

Thus, for optimal solution $Q^*$, it should be true that

$$
Q^* = (\alpha L + I)^{-1}(Q^* + \delta(Y_{\Omega} - Q^*_{\Omega})),
$$

for any $\delta$. Thus $Q^* = (\alpha L + I)^{-1}(Q^* + \delta_k(Y_{\Omega} - Q^k_{\Omega}))$, hold for $\delta_k$, $k = 1, 2, 3, \cdots$. Let $||*||_\lambda$ indicate the spectral norm which gives absolute value of eigenvalue with the largest scale. Recover $L$ is positive semi-definite matrix, it is easy to observe
∥(αL + I)^{-1}∥_λ ≤ 1. Then we may have
\[ ∥Q^* - Q^k∥_F^2 = ∥(αL + I)^{-1}(Q^* + δ_k(YΩ - Q^*_Ω) - (Q^{k-1} + δ(YΩ - Q^{k-1}_Ω)))∥_F^2 \]
\[ ≤ ∥(αL + I)^{-1}∥_λ∥Q^* - Q^{k-1} + δ_k(Q^{k-1}_Ω - Q^*_Ω)∥_F^2 \]
\[ = ∥(αL + I)^{-1}∥_λ∥Q^*_Ω - Q^{k-1}_Ω + δ_k(Q^{k-1}_Ω - Q^*_Ω) + Q^*_Ω - Q^{k-1}_Ω∥_F^2 \]
\[ ≤ |1 - δ_k||Q^*_Ω - Q^{k-1}_Ω∥_F^2 + ∥Q^*_Ω - Q^{k-1}_Ω∥_F^2 \]
\[ < ∥Q^* - Q^{k-1}∥_F^2, \]
if δ_k ∈ (0, 2) which concludes the proof.

Practically, we may set the parameter α to be a large number, such that the optimal solution should nearly minimize the optimization problem (2.8).

Similar to the approach of constructing model (2.6) from (2.5), we may also relax and simplify (2.8) to:

\[ \min_Q \text{tr}(Q^T LQ) + \mu∥Q - Ỹ∥_F^2, \quad (2.10) \]

μ > 0 is a parameter set to balance the contribution of graph constructed on feature domain and the label information in label domain. Model (2.10) can be interpreted as learning missing labels by combining relationship embedded in graph as well as label information available in label domain. The model (2.10) is convex with the optimal condition as:

\[ (L + μI)Q = μỸ. \]

Since μ > 0 and L is positive semi-definite matrix, \((L + μI)^{-1}\) exists and the close form solution of 2.10 can be expressed as \(Q = μ(L + μI)^{-1}Ỹ\). Thus, it is clear that the computational cost of solving (2.10) can be significantly less than which of (2.9). Moreover, the relaxation also provides the opportunity of correcting the miss-labeling case if any appears in label of training data.

### 2.2.3 The Combined Model and its Algorithm

Our main idea is integrating low-rank property in label domain with graph information from feature domain together to perform classification for multi-label data. With both low-rank based model and graph based model are introduced, we are ready to
present the integrated model. By combining (2.6) and (2.10), we may introduce the proposed model as follows.

$$\min_{Q} \frac{1}{2} \text{tr}(Q^T LQ) + \frac{\mu}{2} \|Q - \hat{Y}\|_F^2 + \tau \|Q\|_*.$$ (2.11)

Again, this model is convex. Both (2.6) and (2.10) can be solved explicitly, model (2.11) however, does not have closed form solution. Fortunately, as the convexity of (2.11) is guaranteed, it can be solved efficiently by many existing iterative methods with convergence ensured. In thesis paper, we propose to employ the famous alternating direction methods with multipliers (ADMM) [55].

More specifically, let $J(Q)$ denotes the objective function of (2.11) and we start by introducing auxiliary variable $P$. The corresponding augmented Lagrangian function of $J(Q)$ should be given by:

$$\mathcal{L}_\beta(Q, P, \Gamma) = \frac{1}{2} \text{tr}(Q^T LQ) + \frac{\mu}{2} \|Q - \hat{Y}\|_F^2 + \tau \|P\|_* + \frac{\beta}{2} \|P - Q\|_F^2 + \langle \Gamma, P - Q \rangle.$$ (2.12)

where $\Gamma$ is the Lagrangian multipliers of same size of $Q$, $\langle \star \rangle$ denotes the inner product (entry-wise multiplication of two matrices), and $\beta$ is a positive penalty parameters which are used to control the convergence of the ADMM. The formulation of $\mathcal{L}_\beta(Q, P, \Gamma)$ can be split into several subproblems which allow one to solve $Q$, $P$ and $\Gamma$ at each stage alternatively.

$$\begin{cases}
Q^k = \arg\min_Q \mathcal{L}_\beta(Q, P^{k-1}, \Gamma^{k-1}), \\
P^k = \arg\min_P \mathcal{L}_\beta(Q^k, P, \Gamma^{k-1}), \\
\Gamma^k = \Gamma^{k-1} + \frac{\beta}{2} (P^k - Q^k).
\end{cases}$$

Note that first two subproblems corresponding to $Q$ and $P$ at each step can both be solved explicitly by close form solution respectively:

$$\begin{align*}
\arg\min_Q \mathcal{L}_\beta(Q, P^{k-1}, \Gamma^{k-1}) &= (L + (\mu + \beta)I)^{-1}(\mu \hat{Y} + \beta P^{k-1} + \Gamma^{k-1}); \\
\arg\min_P \mathcal{L}_\beta(Q^k, P, \Gamma^{k-1}) &= \mathcal{D}_\tau(Q^k - \frac{\Gamma^{k-1}}{\beta}).
\end{align*}$$ (2.13)
The main computational cost in each step is performing SVD for matrix of same size as label indicator $Y$. Consider in the MC [6] algorithm, SVD of larger matrix (same size as $Z_{obs} = [X, Y, 1]^T$) is required in each iteration. Therefore, we may expect the computational cost of the proposed algorithm to be lower especially when dealing with high dimensional data. What is more, the proposed algorithm is capable of handling larger scale problem which MC fails to due to the computational limitation of computing SVD for matrix of large size.

Now, we are ready to present the algorithm designed for proposed model.

**Algorithm 1**

| Input: $L$, $\hat{Y}$, $P^0$, $Q^0$, $\Gamma^0$, $\mu$, $\beta$ and tolerance $\varepsilon$ |
| Output: $Q^*$ |
| Procedure: |
| 1. Set $k = 1$; |
| 2. Compute $Q^k = (L + (\mu + \beta)I)^{-1}(\mu\hat{Y} + \beta P^{k-1} + \Gamma^{k-1})$; |
| 3. Compute $P^k = D_r(Q^k - \frac{\Gamma^{k-1}}{\beta})$; |
| 4. Compute $\Gamma^k = \Gamma^{k-1} + \frac{\beta}{2}(P^k - Q^k)$; |
| 5. If $\|Q^k - Q^{k-1}\| < \varepsilon$, set $Q^* = Q^k$; Otherwise set $k = k + 1$ and goto Step 2. |

We claim that the Algorithm 1 designed by following ADMM scheme is guaranteed to converge to the unique minimizer of (2.11) and the detail convergence analysis can be found in [55].

2.3 Experimental Results

2.3.1 Datasets

We evaluated the performance of the proposed method for multi-label classification on six data sets including MSRC$^1$ and five Mulan multimedia datasets$^2$ (Birds, CAL500, Emotion, Core5K, and Mediamill). More detailed information of each date set can be found in Table 2.1, where $N$ is the number of objects, $F$ is the number of features, $K$

---

2http://mulan.sourceforge.net/datasets-mlc.html
is the number of labels/classes. The last column *cardinality* is defined by the average number of labels per objects. Among these data sets, the number of candidate labels varies from 6 to 374, the *cardinality*, on the other hand, varies from 1.104 to 26.044. Another thing should be mentioned here is that the objects belong to different classes also varies in a large range (e.g. the class size in *CAL500* varies from 5 to 444). Based on these facts, it is challenging task to predict the label information for such multi-label data sets.

<table>
<thead>
<tr>
<th>Dataset</th>
<th>Domain</th>
<th>N</th>
<th>F</th>
<th>K</th>
<th>cardinality</th>
</tr>
</thead>
<tbody>
<tr>
<td>Emotion</td>
<td>music</td>
<td>593</td>
<td>72</td>
<td>6</td>
<td>1.869</td>
</tr>
<tr>
<td>Birds</td>
<td>audio</td>
<td>645</td>
<td>258</td>
<td>19</td>
<td>1.104</td>
</tr>
<tr>
<td>MSRC</td>
<td>image</td>
<td>591</td>
<td>512</td>
<td>23</td>
<td>2.508</td>
</tr>
<tr>
<td>Mediamill</td>
<td>video</td>
<td>43907</td>
<td>210</td>
<td>101</td>
<td>4.376</td>
</tr>
<tr>
<td>CAL500</td>
<td>music</td>
<td>502</td>
<td>68</td>
<td>174</td>
<td>26.044</td>
</tr>
<tr>
<td>Corel5K</td>
<td>image</td>
<td>5000</td>
<td>499</td>
<td>374</td>
<td>3.522</td>
</tr>
</tbody>
</table>

Table 2.1: Multi-label data set summary.

### 2.3.2 Methodology

In order to demonstrate the performance of the proposed LRML method, CPLST [10], FAIE [44], MLLOC [33], MC [6], MiML [73] and SLRM [38] are considered for comparison. Note that we do not include PLST [70] since it has been reported in [10] that CPLST outperforms PLST. Among all considered algorithms, the first three are supervised methods which train the classifiers only using the labeled data, while similar to LRML, the later three methods are semi-supervised which take advantage of label information as well as both labeled and unlabeled data. We run MLLOC with the Matlab codes provided by the authors³. All the other five methods are implemented in Matlab by ourselves. All the experiments are conducted on Windows with 4G memory and 2G CPU.

³<http://lamda.nju.edu.cn/code_MLLOC.ashx>

For CPLST and FAIE, the number of reduced latent space dimensions (*b* in Sec-
<table>
<thead>
<tr>
<th>Dataset</th>
<th>Evaluation</th>
<th>CPLST</th>
<th>FAIE</th>
<th>MLLOC</th>
<th>MC</th>
<th>MIML</th>
<th>SLRM</th>
<th>LRML</th>
</tr>
</thead>
<tbody>
<tr>
<td>Emotion</td>
<td>AUC</td>
<td>0.7513</td>
<td>0.7427</td>
<td>0.8021</td>
<td>0.7866</td>
<td>0.8082</td>
<td><strong>0.8155</strong></td>
<td>0.8083</td>
</tr>
<tr>
<td></td>
<td>Macro-F1</td>
<td>0.5986</td>
<td>0.5880</td>
<td>0.6567</td>
<td>0.5872</td>
<td>0.6619</td>
<td>0.6733</td>
<td><strong>0.6784</strong></td>
</tr>
<tr>
<td></td>
<td>Micro-F1</td>
<td>0.6009</td>
<td>0.5918</td>
<td>0.6892</td>
<td>0.6054</td>
<td>0.6734</td>
<td><strong>0.6988</strong></td>
<td>0.6916</td>
</tr>
<tr>
<td></td>
<td>Accuracy</td>
<td>0.5015</td>
<td>0.4904</td>
<td>0.5790</td>
<td>0.4949</td>
<td>0.5773</td>
<td><strong>0.5853</strong></td>
<td><strong>0.6010</strong></td>
</tr>
<tr>
<td></td>
<td>RunTime (s)</td>
<td>0.006</td>
<td>0.008</td>
<td>3.44</td>
<td>6.97</td>
<td>11.23</td>
<td>0.011</td>
<td>0.6288</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Birds</td>
<td>AUC</td>
<td>0.6735</td>
<td>0.6600</td>
<td>0.6738</td>
<td>0.6715</td>
<td>0.7115</td>
<td><strong>0.7236</strong></td>
<td>0.7123</td>
</tr>
<tr>
<td></td>
<td>Macro-F1</td>
<td>0.2297</td>
<td>0.2347</td>
<td>0.2309</td>
<td>0.2875</td>
<td>0.3013</td>
<td><strong>0.3284</strong></td>
<td>0.3018</td>
</tr>
<tr>
<td></td>
<td>Micro-F1</td>
<td>0.4059</td>
<td>0.4040</td>
<td>0.4000</td>
<td>0.3822</td>
<td>0.4138</td>
<td><strong>0.4574</strong></td>
<td><strong>0.4337</strong></td>
</tr>
<tr>
<td></td>
<td>Accuracy</td>
<td>0.3073</td>
<td>0.2919</td>
<td>0.2962</td>
<td>0.2797</td>
<td>0.2873</td>
<td><strong>0.3208</strong></td>
<td><strong>0.3271</strong></td>
</tr>
<tr>
<td></td>
<td>RunTime (s)</td>
<td>0.013</td>
<td>0.026</td>
<td>3.65</td>
<td>7.36</td>
<td>31.35</td>
<td>0.169</td>
<td>0.690</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>MSRC</td>
<td>AUC</td>
<td>0.7887</td>
<td>0.7780</td>
<td>0.5400</td>
<td>0.7857</td>
<td>0.8133</td>
<td><strong>0.8253</strong></td>
<td><strong>0.8138</strong></td>
</tr>
<tr>
<td></td>
<td>Macro-F1</td>
<td>0.3317</td>
<td>0.3467</td>
<td>0.1048</td>
<td>0.2541</td>
<td>0.4083</td>
<td><strong>0.4481</strong></td>
<td><strong>0.4806</strong></td>
</tr>
<tr>
<td></td>
<td>Micro-F1</td>
<td>0.5109</td>
<td>0.5357</td>
<td>0.3692</td>
<td>0.4196</td>
<td>0.5538</td>
<td><strong>0.5890</strong></td>
<td><strong>0.5909</strong></td>
</tr>
<tr>
<td></td>
<td>Accuracy</td>
<td>0.3281</td>
<td>0.3344</td>
<td>0.2070</td>
<td>0.2353</td>
<td>0.2801</td>
<td><strong>0.3866</strong></td>
<td><strong>0.4715</strong></td>
</tr>
<tr>
<td></td>
<td>RunTime (s)</td>
<td>0.059</td>
<td>0.141</td>
<td>33.49</td>
<td>35.55</td>
<td>687.78</td>
<td>0.731</td>
<td>0.491</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Mediamill</td>
<td>AUC</td>
<td>0.7938</td>
<td>0.7793</td>
<td>0.7918</td>
<td>0.7563</td>
<td>0.7705</td>
<td><strong>0.7969</strong></td>
<td>0.7849</td>
</tr>
<tr>
<td></td>
<td>Macro-F1</td>
<td>0.0982</td>
<td>0.1266</td>
<td>0.1399</td>
<td>0.1269</td>
<td>0.1298</td>
<td><strong>0.1413</strong></td>
<td><strong>0.2230</strong></td>
</tr>
<tr>
<td></td>
<td>Micro-F1</td>
<td>0.5785</td>
<td>0.6422</td>
<td>0.6381</td>
<td>0.6273</td>
<td>0.6412</td>
<td><strong>0.6476</strong></td>
<td><strong>0.6432</strong></td>
</tr>
<tr>
<td></td>
<td>Accuracy</td>
<td>0.4264</td>
<td>0.4265</td>
<td>0.4326</td>
<td>0.4509</td>
<td>0.4465</td>
<td><strong>0.4691</strong></td>
<td><strong>0.5156</strong></td>
</tr>
<tr>
<td></td>
<td>RunTime (s)</td>
<td>0.278</td>
<td>10.09</td>
<td>4928.37</td>
<td>2534.60</td>
<td>8953.65</td>
<td>0.790</td>
<td>94.259</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>CAL500</td>
<td>AUC</td>
<td>0.5471</td>
<td>0.5468</td>
<td>0.5155</td>
<td>0.5211</td>
<td>0.5454</td>
<td><strong>0.5585</strong></td>
<td>0.5456</td>
</tr>
<tr>
<td></td>
<td>Macro-F1</td>
<td>0.1547</td>
<td>0.1541</td>
<td>0.1309</td>
<td>0.1366</td>
<td>0.1399</td>
<td><strong>0.1655</strong></td>
<td><strong>0.1734</strong></td>
</tr>
<tr>
<td></td>
<td>Micro-F1</td>
<td>0.4401</td>
<td>0.4410</td>
<td>0.4626</td>
<td>0.4703</td>
<td>0.4704</td>
<td><strong>0.4818</strong></td>
<td><strong>0.4812</strong></td>
</tr>
<tr>
<td></td>
<td>Accuracy</td>
<td>0.3022</td>
<td>0.3024</td>
<td>0.3027</td>
<td>0.3074</td>
<td>0.3016</td>
<td><strong>0.3087</strong></td>
<td><strong>0.3157</strong></td>
</tr>
<tr>
<td></td>
<td>RunTime (s)</td>
<td>0.013</td>
<td>0.018</td>
<td>438.04</td>
<td>19.32</td>
<td>1343.89</td>
<td>0.318</td>
<td>2.403</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Corel5k</td>
<td>AUC</td>
<td>0.5534</td>
<td>0.5547</td>
<td>0.5786</td>
<td>0.5317</td>
<td>0.5573</td>
<td>0.5762</td>
<td><strong>0.5989</strong></td>
</tr>
<tr>
<td></td>
<td>Macro-F1</td>
<td>0.0383</td>
<td>0.0411</td>
<td>0.0273</td>
<td>0.0419</td>
<td>0.0422</td>
<td><strong>0.0497</strong></td>
<td>0.0373</td>
</tr>
<tr>
<td></td>
<td>Micro-F1</td>
<td>0.2241</td>
<td>0.2220</td>
<td>0.2230</td>
<td>0.2305</td>
<td>0.2322</td>
<td><strong>0.2700</strong></td>
<td>0.2255</td>
</tr>
<tr>
<td></td>
<td>Accuracy</td>
<td>0.1256</td>
<td>0.1162</td>
<td>0.1332</td>
<td>0.1447</td>
<td>0.1306</td>
<td><strong>0.1566</strong></td>
<td>0.1458</td>
</tr>
<tr>
<td></td>
<td>RunTime (s)</td>
<td>1.53</td>
<td>2.67</td>
<td>17021.46</td>
<td>1441.99</td>
<td>3957.35</td>
<td>15.36</td>
<td>10.491</td>
</tr>
</tbody>
</table>

Table 2.2: Comparison of multi-label learning performance output by six algorithms on six real world multimedia data sets.
tion 3.5) is selected from $\{\lfloor 0.05K \rfloor, \lfloor 0.1K \rfloor, \lfloor 0.2K \rfloor, \lfloor 0.3K \rfloor, \lfloor 0.4K \rfloor, \lfloor 0.5K \rfloor\}$ if $K$ is greater than 10, otherwise $b$ is tuned in range $[2, K]$ with each step increment by 1.

In the learning stage, both CPLST and FAIE are coupled with linear regression for label prediction. The regularization parameter in MLLOC and MC, and the trade-off parameter in FAIE, $\lambda$ and $\gamma$ in SLRM, and $\mu$, $\tau$ and $\beta$ in LRML are tuned from the candidate set $\{10^{-3}, 10^{-2}, 10^{-1}, 1, 10^1, 10^2, 10^3\}$. For generating the affinity matrix $A$ in both SLRM and LRML, the similarity scores among different objects are all calculated by Gaussian function. The number of nearest neighbors is set to 15, i.e., thresholding value $t$ for matrix $A$ is set to be the 16-th largest similarity score for each objects. The number of groups in MLLOC and the size of compressed label space in MiML are tuned with the same method as $b$ in CPLST. The hyper-parameters in MiML are assigned according to the experimental setting in [73]. The parameters of these methods are tuned by conducting 10-fold cross-validation on the training set.

Prediction performance of all methods are evaluated with the widely-used metrics in the field of multi-label classification, i.e., label-based Macro-F1, Micro-F1, instance-based Accuracy and AUC [89]. The first three evaluation metrics require pre-defining threshold to determine the number of labels for testing data. Here the number of labels for each testing instance is set according to its ground truth. In general, it is hard to set a proper threshold value in real applications. AUC (the area under the Receiver Operating Characteristic ROC curve) is also used while the ROC curve is plot with respect to different threshold values.

### 2.3.3 Results and Discussion

In the first experiment on Birds, Emotion, MSRC, Core5K and Mediamill data sets, the objects in each class are evenly and randomly divided into 10 parts, one part for training and the rest for testing. For the class with less than 10 objects, we randomly select one instance for training and the rest for testing. As the cardinality of CAL500 (26.044) is large, only 1% data in each category are randomly sampled for training and the rest for testing. We run each method for 10 times on each data set. The average results are listed in Table 2.2. The best results of each evaluation measure are marked in bold, and the second best are underlined. According to the results,
we may draw the conclusion that the proposed LRML methods outperforms CPLST, FAIE, MLLOC, MC and MiML across the 6 tested data sets in most cases while obtaining similar classification performance with SLRM.

In addition, the average running time of all the methods on all data sets are listed in Table 2.2. It is easy to observe that the MIML method is usually time consuming mainly because that MiML has to re-train the classifier after selecting the annotated points in each iteration. On the other hand, MC handles SVD for joint matrix concatenating the feature information $X$ and label information $Y$ which costs much more computational time. For SLRM, in each step it involves the SVD computation for mapping $U$ which can be smaller than label indicator $Y$ when the number of features are less than number of objects in data sets which happens to be true for all the tested data sets. Therefore, the computational efficiency of SLRM usually outperforms the proposed algorithm. However, if one is required to handle high dimensional data which has been more and more common in modern applications, the proposed algorithm might figure out the classification problem much faster. For three supervised methods, both CPLST and FAIE are efficient because they only consider the labeled data. The performance of these two methods degrades significantly when there are only a few training data (see the experimental results in Figure 2). When the data set is large, e.g., Mediamill. In addition, as MLLOC has to learn the local encoding for each class based on training data, it is slower than the other methods especially when the size of labels is large, e.g. Corel5K.

Next we studied the label prediction performance under different the labeled data sizes. In the experiment, 1% – 10% of data in each category are employed as training set. For a given percentage, a desired number of data are randomly sampled ten times, and the resulting average “instance-based Accuracy” and “label-based Macro-F1” on the unlabeled data are recorded. We plotted these results on two large data sets Corel5K and Mediamill in Figure 2.1. Obviously, the proposed method LRML outperforms the other methods in most cases. It is interesting that both LRML and SLRM performs well even there are very few labeled data. This might be useful especially for handling large real data sets when training objects available are often very limited.
Furthermore, we tested the performance of both LRML and SLRM on data with missing labels. The tested data sets are generated based on data sets CAL500 and MSRC where the training and testing sets are same with that in Table 2.2. For each training objects, we introduce different ratios of missing labels (including positive and negative). To avoid the appearance of empty category or the case that one object only maintains negative labels, at least one object is kept for each category and at least one positive label is kept for each object. Then the label vector for training instance is set via $Y_{i,j} = 0$ if the $(i, j)$th entry is missing, otherwise $Y_{i,j} = 1$ if the $i$th objects belongs to the $j$th class, and $Y_{i,j} = -1$ if the $i$th object is not in the $j$th class. Note that FAIE MC and SLRM are also capable of handling missing labels, and the results are given in Figure 2.2. LRML shows superior performance over other methods, especially on MSRC. We remark that handling MSRC is more difficult than handling CAL500 when partial labels are missing, because MSRC has less average labels in each instance than CAL500. For the experiments on CAL500,
we notice that SLRM slightly outperforms the proposed methods based on Macro-F1. However, LRML still works much better than FAIE as well as MC. On the other hand, LRML achieves best results in terms of Accuracy.

2.4 Concluding Remarks on LRML Learning Method

We have proposed a new method LRML which is designed specifically for ML learning problem. By assuming the label indicator matrix as low-rank or approximately low-rank, we are able capture the label correlations by enforcing nuclear norm regularization on label indicator. Inspired by successful application of spectral clustering methods, LRML also makes use of both labeled and unlabeled data by considering the intrinsic geometric structure among different objects. What is more, the proposed model also considered the labeled information already known by minimizing square loss function between a labeled data and resulting label indicator of the method. In
order to deal with large-scale data, an efficient algorithm based on ADMM framework is developed to solve the proposed model. A series of experiments shows that the proposed LRML method may obtain superior performance compared with state-of-the-art methods under varying the labeled data size and ratio of missing labels which demonstrates the effectiveness and efficiency of the proposed method.

Moreover, LRML should also be capable of label correction as SLRM does for data with miss-assigned labels. In the future, we might also test the performance on label correction of LRML and make a comprehensive comparison with SLRM.
Chapter 3

Multi-instance Multi-label Learning

In this chapter, we mainly focus on multi-instance multi-label (MIML) learning problems [91]. Let \( X \) be a set of objects and \( Y \) be a set of labels or classes. We denote the size of \( Y \) by \( c = |Y| \), and the size of \( X \) by \( m = m' + m'' = |X| \), where \( m' \) and \( m'' \) are the sizes of training data and testing data, respectively. In the single-instance single-label (SISL) learning setting, each object (i.e., only one instance) \( X \in X \) is assigned a single class \( Y \in Y \) while in ML learning, each object may be categorized by several labels. On the other hand, in the multi-instance multi-label setting, the training data set is \( \{(X_1,Y_1),(X_2,Y_2),\ldots,(X_m,Y_m)\} \), where the \( i \)-th object \( X_i = \{x_1^{(i)},\ldots,x_n^{(i)}\} \) contains a bag of \( n_i \) instances, and \( Y_i = \{y_1^{(i)},\ldots,y_l^{(i)}\} \subset \{1,2,\ldots,c\} \) is a set of labels assigned to \( X_i \). The testing data set is \( X_{m'+1},\ldots,X_{m'+m''} \) without labels information.

It is clear that the SISL scenario is a special case of MIML setting where \( n_i = 1 \) and \( l_i = 1 \) for all \( X_i \) and \( Y_i \). In addition, ML learning task can be also considered as special case of MIML learning problem with \( n_i = 1 \). Basically, the MIML learning task is to learn a classifier \( \Phi : X \rightarrow Y \) which minimizes the probability that \( \hat{Y} \neq \Phi(\hat{X}) \) on a testing object \( \hat{X} \) with its set of labels \( \hat{Y} \). Compared to both SISL and ML learning problems, the MIML learning problems are more practical to represent complicated objects which have multiple semantic meanings. For example, an large image can contain multiple patches where each can be represented by an instance, and the image can belong to several classes simultaneously; a long document can
contain multiple sections where each of which can be represented as an instance, and the document can belong to different categories.

In next section, Some of the state-of-art MIML methods are briefly reviewed. Then, we give technical details the method we have proposed as well as experimental results and concluding remarks respectively.

3.1 Brief Review on Multi-instance Multi-label Learning Algorithms

So far in the literature, there are several algorithms that have been developed for solving MIML problems. Let us briefly review them before go into detailed methods.

1. **MIML-SVM** [90,91]: This algorithm is based on support vector machine. The approach is motivated by solving Multi-label (ML) learning problems [19]. The main idea is to transform the MIML problem into single instance Multi-label problem by using the \( k \)-medoids algorithm, and then employ support vector machine (SVM) to predict classes for unlabeled objects. The computational complexity of the MIML-SVM method is huge when the number of training samples is large. However, when the number of training samples is small (i.e., there is limited information for learning), the classification accuracy may be low.

2. **MIML-\( k \)NN** [87]: This algorithm is based on \( k \)-nearest neighbor (\( k \)NN) method [76]. Similar to MIML-SVM, the approach is to consider ML learning problems. The MIML-\( k \)NN method employs Hausdorff metric [22] to measure the distance between each object which is represented by a bag of instances, and then solve the classification task by using \( k \)NN. Again the computational cost of the MIML-\( k \)NN is large when we deal a large MIML data set.

3. **MIML-Boost** [90,91]: This algorithm is different from the MIML-SVM and MIML-\( k \)NN. The approach is to decompose the MIML problem into several Multi-Instance (MI) learning problems [62]. For each MI learning problem, it can be solved by the MI-boosting algorithm [79] which is a supervised learning
algorithm. However, the boosting algorithm can take a huge amount of computational time when we deal with a large number of ML problems. In this paper, we do not compare proposed algorithm with MIML-Boost algorithm due to the time limit.

4. M³MIML [88]: This algorithm is based on optimization approach by exploiting the relationship between instances and labels. For each class label, the M³MIML algorithm first solves a linear regularized optimization problem such that a maximum prediction for all instances can be generated. Then the prediction results are combined together to construct the final classification results for all the objects. According to the numerical results, the M³MIML algorithm is more efficient than the MIML-Boost algorithm, but it is still more expensive than the MIML-SVM and MIML-kNN algorithms.

5. Markov-MIML [65,78]: This algorithm is based on Markov chain framework. In the algorithm, information of nearest neighbors for all instances is utilized to predict labels of testing samples. The training step is formulated as random walk process with restart. Experimental results on benchmark data sets have shown that both classification results and computational time by the Markov-MIML algorithm are competitive with the above mentioned MIML algorithms. However, the main bottleneck of the algorithm is required to build a huge affinity matrix for Markov chain construction. When the number of training and testing instances is large, the construction cost would be very expensive.

In the next stage, we focus on introducing the proposed sparsity based for MIML learning algorithms.

3.2 A Sparsity-Based Multi-instance Multi-label Learning Algorithm

Compared with the other existing algorithms, we are interested in classifiers which can generate a ranking of labels for a given object such that its correct labels receive higher ranking than the other irrelevant labels. Our aim is to compute label indicator
for a testing \( \tilde{X} \) with its set of labels \( \tilde{Y} \) such that the labels of \( \tilde{X} \) in \( \mathcal{Y} \) should be ordered according to \( F \), i.e., if \( F(\tilde{X}, y) > F(\tilde{X}, y') \), then the label \( y \) is ranked higher than the label \( y' \). The classifier is evaluated in terms of its ability to predict a good approximation for \( \tilde{Y} \) based on the ranking by \( F \). In particular, we would like to obtain the ranking of labels in \( \hat{Y} \) is higher than the ranking of those not in \( \tilde{Y} \).

Our main contribution is to propose and develop a novel sparsity-based MIML learning algorithm to compute a ranking of labels associated with objects of multiple instances. We exploit the relationships among instances and labels of objects so that the ranking of a class label to an object depends on the affinity metric between the bag of instances of this object and the bag of instances of the other objects, and the ranking of a class label of similar objects. By employing instance-to-object relation matrix and computing the affinity metric among the objects, a transductive objective function for labels indicator to be learned can be constructed. Moreover, sparsity can be introduced in the labels indicator of the objective function such that relevant and irrelevant objects with respect to a given class can be distinguished. For example, we show in Figure 3.1 the sparsity of the association of class labels and testing objects from the two benchmark MIML data sets [91]. We see from the figure that each object associates a few classes. The resulting sparsity-based MIML model can be formulated as a constrained convex optimization problem, and it can be solved very efficiently by using the augmented Lagrangian method based on separable structure of the objective function. Experimental results on benchmark data will show that the proposed sparse-MIML algorithm is computationally efficient, and effective in label prediction for MIML data, and its performance is better than the other MIML learning algorithms.

The main idea of the method of random walk with restart is to set up the affinities among objects on MIML data, and initialize label information from labeled objects [72]. As there are multiple instances among objects in MIML data, we evaluate the affinity between the two objects which may have different set of instances. An affinity matrix is constructed to represent affinity among instances, and then employ instance-to-object-relation matrix to transfer the affinity information among instances to the label information among objects. We consider that the instances are ordered
Figure 3.1: Sparsity of MIML image data (left) and MIML text data (right). A red (blue) pixel refers that an object belongs (does not belong) to a class.

as follows:

\[
\begin{align*}
\begin{array}{cccc}
  x_1^{(1)}, & \ldots, & x_{n_1}^{(1)}, & x_1^{(2)}, \ldots, x_{n_2}^{(2)}, \ldots, x_1^{(m)}, \ldots, x_{n_m}^{(m)} \\
\end{array}
\end{align*}
\]

For simplicity, we set \( n \) to be the total number of instances in the MIML data, i.e., \( n = \sum_{i=1}^{m} n_i \). Let \( a_{i,j,s,t} \) be the affinity between the \( s \)-th instance of the \( i \)-th object and the \( t \)-th instance of the \( j \)-th object. Here we employ the Gaussian kernel as the affinity function similar to that used in other MIML learning algorithms [88,90,91]:

\[
a_{i,j,s,t} = \exp \left[ -\frac{\|x_s^{(i)} - x_t^{(j)}\|_2^2}{2\sigma^2} \right],
\]

where \( \| \cdot \|_2 \) is the Euclidean distance. An \( n \)-by-\( n \) block matrix \( A = [A_{i,j}] \) where the \((i,j)\)-th block is an \( n_i \)-by-\( n_j \) matrix \( A_{i,j} = [a_{i,j,s,t}] \) with \( s = 1, \ldots, n_i \) and \( t = 1, \ldots, n_j \) is obtained. Note that \( A \) is a symmetric matrix. Moreover, a block diagonal matrix \( B = [B_{i,j}] \) where the \((i,j)\)-th block is a zero matrix except \( i = j \). For the \((i,i)\)-th block, \( B_{i,i} \) is a 1-by-\( n_i \) matrix where all its entries are equal to 1. This block indicates the relation between the \( i \)-th object and its association instances. The size of \( B \) is \( m \)-by-\( n \), and it refers to be an object-to-instance relation matrix that can be used to transfer from the affinity information at the instance level to the object level. The resulting \( m \)-by-\( m \) matrix

\[
S = BAB^T
\]

represents the affinities among objects.
For label prediction problem with partial labeled data, the method of random walk with restart has probability \( \alpha \) \((\alpha \geq 0)\) to return to the labeled objects. It can be interpreted that during each time step each object receives the label information from its neighbors via the Markov chain, and also retains its initial label information. The parameter \( \alpha \) in between 0 and 1 specifies the relative amount of the information from its neighbors and its initial label information. In this approach, the steady state probabilities give ranking of labels to indicate the importance of a set of labels to a unlabeled object \[72\]. The steady state probabilities can be computed via the following recursive procedure:

\[
F_{(t+1)} = (1 - \alpha)SD^{-1}F_{(t)} + \alpha P,
\]

where \( D \) is a diagonal matrix with its main diagonal entry given by \([D]_{i,i} = \sum_{k=1}^{m}[S]_{k,i}\). Here \( F = [f_1, f_2, \cdots, f_c] \) is the labels indicator to be learned, where \( f_l \) is a vector of size \( m \) corresponding to the \( l \)-th class label, and \( F_{(t)} \) refers to the iterate of \( F \) at the \( t \)-th iteration. Also \( P = [p_1, p_2, \cdots, p_c] \) is the given labels of training objects, where \( p_l \) is a vector of size \( m \) corresponding to the \( l \)-th class label. One way to construct \( p_l \) is by using an uniform distribution on the objects with the label class \( l \). More precisely,

\[
[p_l]_i = \begin{cases} 
1/e_l, & \text{if } l \in Y_i \\
0, & \text{otherwise}
\end{cases}
\]

where \( e_l \) is the number of objects with the label class \( l \) in the training data. Note that the summation of entries of \( p_l \) is equal to 1, and the summation of entries of \( f_l \) is also equal to 1 via the iteration in (3.2).

By using the iteration in (3.2), we have

\[
F_{(t)} = (1 - \alpha)SD^{-1}F_{(0)} + \alpha \sum_{i=\infty}^{t-1}((1 - \alpha)SD^{-1})^iP,
\]

where \( F_{(0)} \) is the initial estimate of the labels indicator. Since \( \alpha \) is in between 0 and 1 and the spectral radius of \( SD^{-1} \) is equal to 1\(^1\), we have \( \lim_{t\to\infty}((1 - \alpha)SD^{-1})^tF_{(0)} = 0 \), and \( \lim_{t\to\infty} \sum_{i=0}^{t-1}((1 - \alpha)SD^{-1})^iP = (I - (1 - \alpha)SD^{-1})^{-1}P \), where \( 0 \) is a zero matrix.

\(^1\)Each column sum of \( SD^{-1} \) is equal to 1 and the results follow by using the Perron Frobenius Theorem.
and $I$ is an $m$-by-$m$ matrix. Hence, the sequence $\{F(t)\}$ converges to $F^* = \alpha(I - (1 - \alpha)SD^{-1})^{-1}P$ or
\[
\hat{F}^* = \alpha(I - (1 - \alpha)D^{-1/2}SD^{-1/2})^{-1}\hat{P},
\] (3.4)
where $\hat{F}^* = D^{-1/2}F^*$ and $\hat{P} = D^{-1/2}P$. Indeed, this solution can be obtained by minimizing the following objective function:
\[
J_1(\hat{F}) = \text{tr}(\hat{F}^T(I - D^{-1/2}SD^{-1/2})\hat{F}) + \mu\|\hat{F} - \hat{P}\|_F^2.
\] (3.5)
with $\mu = \frac{\alpha}{1 - \alpha}$. Let $L = I - D^{-1/2}SD^{-1/2}$, note that the matrix $L$ can be interpreted as the graph Laplacian, see [15]. As $I - D^{-1/2}SD^{-1/2}$ is similar to $I - SD^{-1}$ and the spectral radius of $SD^{-1}$ is also equal to 1. Therefore, $I - D^{-1/2}SD^{-1/2}$ is positive semi-definite, and (3.5) is a convex optimization problem. The optimization problem in (3.5) makes use of relationships among instances and labels of objects. An object, which contains a bag of instances that are highly similar to bags of instances of the other objects with a particular label, receives a high ranking of this label.

3.2.1 The Sparse Model

We note from (4.1) that the calculation of each column of $F$ is independent, and the solution $f_l^*$ is dependent on the iteration matrix $(1 - \alpha)SD^{-1}$ and the $l$-th label vector $p_l$. The disadvantage of this approach is that we do not make use of given label information from different classes to compute labels indicator. The main contribution of this paper is to explore the property of $F$ (or $\hat{F} = D^{-1/2}F$) to enhance classification accuracy by using given label information from different classes. Our idea is to use the sparsity of $F$ in the objective function. The aim is to distinguish the relevant and irrelevant objects with respect to a given class, i.e., each column vector of $F$: $[[F]_{1,l}, [F]_{2,l}, \cdots, [F]_{n,l}]$ should be sparse. By combining these two sparse constraints in the objective function, we can reformulate $J_1$ in (3.5) as follows:
\[
\min_{\hat{F}} J(\hat{F}) = \text{tr}(\hat{F}^T L\hat{F}) + \mu\|\hat{F} - \hat{P}\|_F^2 + \lambda \sum_{l=1}^{c} \sqrt{\sum_{i=1}^{n} [\hat{F}]^2_{i,l}}.
\] (3.6)
Since the element $[F]_{i,l}$ is in between 0 and 1, $[\hat{F}]_{i,l}$ is in between 0 and $b = \sqrt{\sum_{i=1}^{n} [S]_{k,l}}$.

The first term of the right-hand side in the above cost function is the smoothness constraint, which means that a proper labels indicator $\hat{F}$ should not change too much.
between neighbor objects. To minimize the first term, we expect that if two objects \( X_i \) and \( X_j \) are close (i.e., \([S]_{i,j}\) is large or \([L]_{i,j}\) is small), \([\hat{F}]_i\), and \([\hat{F}]_j\) are also close to each other. The second term is the fitting constraint, which means that a good labels indicator should not change too much from the initial label assignment. The third term is the sparsity requirement, which means that each object labels indicator should be corresponding to a few categories. Because of the sparsity constraint, both intra-class (column vector of \( \hat{F} \)) and inter-class (row vector of \( \hat{F} \)) label rankings are considered together in the model. The trade-off between these three terms is captured by two positive parameters \( \mu \) and \( \lambda \).

After solving \( \hat{F} \) (i.e., \( F \)) by using (3.6), we generate a ranking of the possible labels for a testing object \( X_i \) by ordering the values of each row of \( F: [F]_{i,1}, [F]_{i,2}, \cdots, [F]_{i,c} \) where \( c \) refers to the number of classes, such that its correct labels receive higher ranking than the other irrelevant labels.

### 3.2.2 The Proposed Algorithm

The objective function in (3.6) is a constrained nonlinear optimization problem with respect to \( \hat{F} \). Note that \( J(\hat{F}) \) is a quadratic and continuously differentiable function and the constraints are bounded and closed, and therefore there is one and only one minimizer of \( J \). To handle the constrains of \( \hat{F} \) efficiently, we introduce two new variables \( \hat{G} \) and \( \hat{H} \) and set \( \hat{F} = \hat{G} \) and \( \hat{G} = \hat{H} \). The corresponding augmented Lagrangian function \( J_a \) of (3.6) is given by

\[
J_a(\hat{F}, \hat{G}, \hat{H}, \Gamma^{(1)}, \Gamma^{(2)})
= \text{tr}(\hat{F}^T L \hat{F}) + \mu \| \hat{G} - \hat{P} \|_F^2 + \lambda \sum_{i=1}^{c} \sqrt{\sum_{i=1}^{n} [\hat{F}]_{i,i}^2} + \\
\langle \Gamma^{(1)}, (\hat{F} - \hat{G}) \rangle + \langle \Gamma^{(2)}, (\hat{G} - \hat{H}) \rangle + \beta_1 \| \hat{F} - \hat{G} \|_F^2 + \beta_2 \| \hat{G} - \hat{H} \|_F^2,
\]

with \( 0 \leq [\hat{H}]_{i,i} \leq b \), where \( \Gamma^{(1)} \) and \( \Gamma^{(2)} \) are the Lagrangian multipliers of same size of \( \hat{F} \) and \( \beta_1, \beta_2 \) are positive penalty parameters which are used to control the convergence of the augmented Lagrangian method (ALM). Now \( (\hat{F}, \hat{H}) \) and

\( ^2 \)Theoretically, the ALM method is convergent for any constant \( \beta_1, \beta_2 > 0 \). If necessary, we can adjust their values of \( \beta_1 \) and \( \beta_2 \) dynamically in order to achieve better numerical performance, see [55] for a detailed discussion.
\( \hat{G} \) in (3.7) can be minimized separately, and each minimization subproblem has a closed-form solution. More specifically, with a given initial \((\hat{G}_{(0)}, \Gamma_{(0)}^{(1)}, \Gamma_{(0)}^{(2)})\), the ALM approaches the solutions of (3.7) via the following iterative scheme (\( t \) refers to the iteration index):

\[
\hat{F}^{t+1} = \arg \min_{\hat{F}} J_a(\hat{F}, \hat{G}(t), \hat{H}(t), \Gamma(t)^{(1)}, \Gamma(t)^{(2)}) = (L + \beta_1 I + \mu I)^{-1}(\mu \hat{P} + \beta_1 \hat{G}^t - \Gamma(t)^{(1)}); \quad (3.8)
\]

for \( 1 \leq l \leq c \) (for each column),

\[
\hat{H}_{t+1}^{l} = \arg \min_{\hat{H}} J_a(\hat{H}(t+1), \hat{G}(t), \hat{H}, \Gamma(t)^{(1)}, \Gamma(t)^{(2)}) \text{ subject to } 0 \leq [\hat{H}]_{t,l} \leq b
\]

\[
= \frac{\hat{G}^{t+1}_{t,l} + \frac{1}{\beta_2}(\Gamma^{(2)}(t);t)}{\|\hat{G}^{t+1}_{t,l} + \frac{1}{\beta_2}(\Gamma^{(2)}(t);t)\|_2} \max \left\{ \left\| \hat{G}^{t+1}_{t,l} + \frac{1}{\beta_2}(\Gamma^{(2)}(t);t) \right\|_2 - \frac{\lambda}{\beta_2}, 0 \right\}; \quad (3.9)
\]

\[
\hat{G}^{t+1} = \max \left\{ \frac{\Gamma^{(1)} - \Gamma^{(2)} + \beta_1 \hat{F}^{t+1} + \beta_2 \hat{H}^t}{\beta_1 + \beta_2}, 0 \cdot E \right\} \quad (3.10)
\]

where \( E \) is a matrix of all entries being equal to one, and the maximum operation is entrywise based comparison;

\[
\Gamma^{(1)}_{t+1} = \Gamma^{(1)} + \frac{\beta_1}{2}(\hat{F}^{t+1} - \hat{G}^{t+1}); \quad (3.11)
\]

\[
\Gamma^{(2)}_{t+1} = \Gamma^{(2)} + \frac{\beta_2}{2}(\hat{G}^{t+1} - \hat{H}^{t+1}). \quad (3.12)
\]

We summarized the developed algorithm as following.

**Algorithm 2**

**Input:** \( L, \hat{P}, F^0, H^0, G^0, \Gamma_{(0)}^1, \Gamma_{(0)}^2, \mu, \lambda, \beta_1, \beta_2 \) and tolerance \( \varepsilon \)

**Output:** \( G^* \)

**Procedure:**

1. Set \( k = 1 \);
2. Compute \( F^k \) following (3.8);
3. Compute \( H^k \) following (3.9);
4. Compute \( G^k \) following (3.10);
5. Compute \( \Gamma^1_k \) following (3.11);
5. Compute \( \Gamma^2_k \) following (3.12);
6. If $\|G^k - G^{k-1}\| < \varepsilon$, set $G^* = G^k$, Otherwise set $k = k + 1$ and goto Step 2.

Theoretically, the convergence of the iteration steps in Algorithm 2 can be guaranteed and the iterate converges to the unique minimizer of $J$ in (3.6).

We remark that all the computations of $\hat{G}_{(t+1)}$, $\hat{H}_{(t+1)}$, $\Gamma^{(1)}_{(t+1)}$, $\Gamma^{(2)}_{(t+1)}$ can be carried out independently at each entry. These computation tasks can be done very efficiently. As for calculation of $\hat{F}_{(t+1)}$, a $m$-by-$m$ linear system is required to solved. The main computational cost and storage is to build and store the whole affinity matrix $S$ or $A$ in (3.1). In the implementation, we figure out that there are many entries of the affinity matrix $A$ being close to zero. Therefore it may not be necessary to store these entries such that the computational cost and the storage can be reduced. For example, we can generate a sparse and symmetric affinity matrix $A_\kappa$ as follows [78]:

$$a_{i,j,s,t} = \begin{cases} 
\exp[-\|x_{s}^{(i)} - x_{t}^{(j)}\|_2^2/2\sigma^2], & \text{if } x_{t}^{(j)} \in \mathcal{N}_\kappa, \\
0, & \text{otherwise}.
\end{cases}$$

where $\mathcal{N}_\kappa$ is the set of $\kappa$ nearest neighbors of instance $x_{s}^{(i)}$. The $\kappa$ nearest neighbors can be efficient searched by using $kd$-tree implementation. When $A_\kappa$ is sparse, the resulting affinity matrix $S_\kappa = BA_\kappa B^T$ is also sparse and the corresponding sparse linear system can be solved very efficiently.

### 3.3 Experimental Results

To evaluate the performance of the proposed sparse-MIML algorithm, we conduct experiments on two benchmark MIML data sets. The first one is the image classification task given in [90] to study the MIML framework. In summary, this data set contains 2,000 scene images taken from five possible class labels (desert, mountains, sea, sunset and trees) and each image is represented as a bag of 9 instances in 15-dimensional using the SBN image bag generator [48]. Another data set is the widely used Reuters-21578 text collection in text categorization. In summary, this data set contains 2,000 documents with multiple classes and represented as a bag of instances based on the techniques of sliding windows, where each instance corresponds to a

---


4 available at http://lamda.nju.edu.cn/datacode/miml-text-data.htm
The description of these two data sets is listed in Table 3.1, see [88] for the detailed information. The text and image data are both preprocessed in our experiments. Each instance is normalized such that its Euclidean norm is equal to 1. Also we follow the setting in [88], we normalize the image data on each dimension in the range between $[0, 5]$ in the $M^3$MIML algorithm. In the tests, we compare sparse-MIML with MIML-RWR. All the comparisons are performed in a computer running a server environment with 2.66GHz CPU and 3.5GB memory.

The performance of multi-label prediction is evaluated by four multi-label ranking metrics: one-error, ranking loss, coverage and average precision. One-error computes how many times the top-ranked label is not relevant. Ranking loss computes the average fraction of label pairs that are not correctly ordered. Coverage determines how far one needs to go in the list of labels to cover all the relevant labels of an instance. This measure is loosely related to the precision at the level of perfect recall. Average precision determines the percentage of relevant labels among all labels that are ranked above. For their detailed definitions, we refer to [62].

### 3.3.1 Classification Performance

We compare the average performance sparse-MIML with MIML-RWR for the two MIML data sets. Tables 3.2 and 3.3 show the performance of different MIML learning algorithms on the text and image data set respectively. The results in these two tables can be divided into two parts. The upper part shows how much the performance improved by sparse-MIML varies with different values of $\kappa$ respectively in the construction of sparse affinity matrix $S_\kappa$ in Section 3. When $\kappa = \text{all}$, it means that all the entries are used. The value following “±” gives the standard deviation. For the sparse-MIML, we fix $\alpha = 0.99$ and $\lambda = 0.11$ for the text data and $\alpha = 0.64$ and $\lambda = 0.17$ for the image data. For the the MIML-RWR algorithm, we choose its corresponding optimal parameters. We see from Table 3.2 and 3.3 that sparse-MIML consistently achieves highly competitive performance with MIML-RWR learning al-

---

5MIML-Boost, MIML-SVM, M$^3$MIML and MIML-kNN are not tested as its classification performance is poorer than MIML-RWR as reported in [78].
<table>
<thead>
<tr>
<th>Data set</th>
<th>Objects</th>
<th>Instances</th>
<th>Features</th>
<th>classes</th>
<th>Instance per bag</th>
<th>Labels per object (c)</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>min</td>
<td>max</td>
</tr>
<tr>
<td>Text</td>
<td>2,000</td>
<td>7,119</td>
<td>243</td>
<td>7</td>
<td>2</td>
<td>26</td>
</tr>
<tr>
<td>Image</td>
<td>2,000</td>
<td>18,000</td>
<td>15</td>
<td>5</td>
<td>9</td>
<td>9</td>
</tr>
</tbody>
</table>

Table 3.1: The description of image and text data for MIML learning.
algorithm across all evaluation metrics and data sets in most cases. We also note that
the computational times of sparse-MIML and MIML-RWR algorithms are about the
same (sparse-MIML: about 200 seconds and MIML-RWR: about 184 seconds), and
both increase when $\kappa$ increases. We would like to report that the proposed algo-

\[ \| \hat{F}_{t+1} - \hat{F}_t \|^2_F / \| \hat{F}_t \|^2_F \leq 1 \times 10^{-15}. \]

In the next experiment, we test the performance of sparse-MIML algorithm with
respect to the number of training examples. We randomly pick up 10, 20, 30, 40,
50, 60, 70, 80, 90 percentages of the data set as training data. The remaining data
set is used for testing data. The performance is measured by averaging 10 trials
by randomly selected data using this procedure. Tables 3.4 and 3.5 show the im-

\[ \text{Tables 3.4 and 3.5 show the im-} \]

provement of classification performance of sparse-MIML than MIML-RWR learning
algorithms on the text and image data set respectively when $\kappa = 100$. We note that
the classification performance when $\kappa = 100$ is higher than those in the other values
of $\kappa$ in Tables 3.2 and 3.3. For the text data, here it is optimal to use the same set
of parameters in Table 3.2. For the image data, we use the same set of parameters
in Table 3.3 when the percentages of the data are in between 30 and 90. However,
when the percentages of the image data is 10 and 20, we need to tune the parameters
to be $\alpha = 0.99$ and $\lambda = 0.1$. We see from Tables 4 and 5 that the performance of
sparse-MIML is usually better than that of MIML-RWR\(^6\).

Note that in both two sets of experiments, we tune the parameters for sparse-
MIML method mainly based on the lowest one-error which is the first ranking metric
we mentioned. However, in some cases, the other ranking metrics may not achieve
optimal value at the same time with one-error. Therefore we may find that in the
tables, we get some negative values for sparse-MIML method.

\(^6\)Because of the limited space, we have not reported the classification performance of the other
algorithms. Indeed, the performance of sparse-MIML on considered data sets is better than those
of MIML-SVM, MIML-Boost and $M^3$MIML which have been reported in [78].
Table 3.2: Improvement by percentage of sparse-MIML method compared with MIML-RWR method on the text data.

<table>
<thead>
<tr>
<th>sparse-MIML</th>
<th>one-error</th>
<th>coverage</th>
<th>rankLoss</th>
<th>1-avgPrec</th>
</tr>
</thead>
<tbody>
<tr>
<td>50</td>
<td>16.33%</td>
<td>4.27%</td>
<td>12.9%</td>
<td>15%</td>
</tr>
<tr>
<td>100</td>
<td>12.63%</td>
<td>3.45%</td>
<td>10%</td>
<td>12.07%</td>
</tr>
<tr>
<td>200</td>
<td>11.58%</td>
<td>3.16%</td>
<td>10%</td>
<td>10.37%</td>
</tr>
<tr>
<td>500</td>
<td>7.37%</td>
<td>0.87%</td>
<td>6.67%</td>
<td>6.90%</td>
</tr>
<tr>
<td>1000</td>
<td>1.06%</td>
<td>-1.74%</td>
<td>0%</td>
<td>0%</td>
</tr>
<tr>
<td>1500</td>
<td>6.38%</td>
<td>1.45%</td>
<td>6.67%</td>
<td>5.26%</td>
</tr>
<tr>
<td>all</td>
<td>2.15%</td>
<td>-3.2%</td>
<td>0%</td>
<td>1.75%</td>
</tr>
</tbody>
</table>

Table 3.3: Improvement by percentage of sparse-MIML method compared with MIML-RWR method on the image data.

<table>
<thead>
<tr>
<th>sparse-MIML</th>
<th>one-error</th>
<th>coverage</th>
<th>rankLoss</th>
<th>1-avgPrec</th>
</tr>
</thead>
<tbody>
<tr>
<td>50</td>
<td>9.78%</td>
<td>2.15%</td>
<td>4.88%</td>
<td>5%</td>
</tr>
<tr>
<td>100</td>
<td>8.05%</td>
<td>2.94%</td>
<td>5.36%</td>
<td>6.70%</td>
</tr>
<tr>
<td>200</td>
<td>7.51%</td>
<td>3.8%</td>
<td>5.78%</td>
<td>6.51%</td>
</tr>
<tr>
<td>500</td>
<td>9.88%</td>
<td>5.02%</td>
<td>7.30%</td>
<td>8.56%</td>
</tr>
<tr>
<td>1000</td>
<td>9.66%</td>
<td>6.15%</td>
<td>9.19%</td>
<td>4.80%</td>
</tr>
<tr>
<td>1500</td>
<td>9.64%</td>
<td>5.76%</td>
<td>8.47%</td>
<td>8.94%</td>
</tr>
<tr>
<td>all</td>
<td>10.81%</td>
<td>5.48%</td>
<td>8.76%</td>
<td>9.17%</td>
</tr>
</tbody>
</table>
### Table 3.4: Improvement by percentage of sparse-MIML method compared with MIML-RWR method with different sizes training data on the text data.

<table>
<thead>
<tr>
<th>sparse-MIML</th>
<th>one-error</th>
<th>coverage</th>
<th>rankLoss</th>
<th>1-avgPrec</th>
</tr>
</thead>
<tbody>
<tr>
<td>10%</td>
<td>1.9%</td>
<td>1.02%</td>
<td>0%</td>
<td>2.02%</td>
</tr>
<tr>
<td>20%</td>
<td>10.45%</td>
<td>1.15%</td>
<td>0%</td>
<td>4.82%</td>
</tr>
<tr>
<td>30%</td>
<td>5.83%</td>
<td>-1.71%</td>
<td>0%</td>
<td>4%</td>
</tr>
<tr>
<td>40%</td>
<td>10.71%</td>
<td>0%</td>
<td>-2.78%</td>
<td>1.43%</td>
</tr>
<tr>
<td>50%</td>
<td>2.88%</td>
<td>-2.70%</td>
<td>-6.06%</td>
<td>0%</td>
</tr>
<tr>
<td>60%</td>
<td>4.90%</td>
<td>-2.79%</td>
<td>-3.13%</td>
<td>0%</td>
</tr>
<tr>
<td>70%</td>
<td>12.24%</td>
<td>-1.14%</td>
<td>0%</td>
<td>6.67%</td>
</tr>
<tr>
<td>80%</td>
<td>7.45%</td>
<td>-3.49%</td>
<td>3.33%</td>
<td>3.45%</td>
</tr>
<tr>
<td>90%</td>
<td>10.75%</td>
<td>0.59%</td>
<td>3.57%</td>
<td>7.27%</td>
</tr>
</tbody>
</table>

### Table 3.5: Improvement by percentage of sparse-MIML method compared with MIML-RWR method with different sizes training data on the image data.

<table>
<thead>
<tr>
<th>sparse-MIML</th>
<th>one-error</th>
<th>coverage</th>
<th>rankLoss</th>
<th>1-avgPrec</th>
</tr>
</thead>
<tbody>
<tr>
<td>10%</td>
<td>1.72%</td>
<td>0.94%</td>
<td>1.33%</td>
<td>1.5%</td>
</tr>
<tr>
<td>20%</td>
<td>0.81%</td>
<td>1.58%</td>
<td>1.5%</td>
<td>1.65%</td>
</tr>
<tr>
<td>30%</td>
<td>1.12%</td>
<td>2.36%</td>
<td>-0.52%</td>
<td>0.43%</td>
</tr>
<tr>
<td>40%</td>
<td>2.03%</td>
<td>0.5%</td>
<td>0%</td>
<td>1.79%</td>
</tr>
<tr>
<td>50%</td>
<td>2.67%</td>
<td>1.01%</td>
<td>1.12%</td>
<td>2.27%</td>
</tr>
<tr>
<td>60%</td>
<td>3.90%</td>
<td>2.23%</td>
<td>3.41%</td>
<td>3.69%</td>
</tr>
<tr>
<td>70%</td>
<td>5.45%</td>
<td>3.47%</td>
<td>5.11%</td>
<td>5.09%</td>
</tr>
<tr>
<td>80%</td>
<td>4.97%</td>
<td>2.22%</td>
<td>4.17%</td>
<td>4.31%</td>
</tr>
<tr>
<td>90%</td>
<td>8.05%</td>
<td>2.94%</td>
<td>5.36%</td>
<td>6.70%</td>
</tr>
</tbody>
</table>

Table 3.4: Improvement by percentage of sparse-MIML method compared with MIML-RWR method with different sizes training data on the text data.

Table 3.5: Improvement by percentage of sparse-MIML method compared with MIML-RWR method with different sizes training data on the image data.
3.3.2 Effect of Parameters

In this subsection, we study the effect of parameters $\alpha$ and $\lambda$ to the classification performance of the sparse-MIML algorithm. In Figures 3.2 and 3.3, we show the performance of the proposed method with 10% and 90% training image data for different values of $\alpha$ and $\lambda$. When 10% training data is used, the optimal parameters are $\alpha = 0.99$ and $\lambda = 0.1$. We need to make use of affinity information from unlabeled data ($\alpha$ is close to 1) to determine the ranking labels. When 90% training data is used, the optimal parameters are $\alpha = 0.64$ and $\lambda = 0.17$. Because there is more labeled data, the usage of affinity information from unlabeled data can be reduced and the algorithm make use of the sparsity to enhance classification performance. It is interesting to note that the optimal values of parameters are quite consistent for all four evaluation metrics. In Figure 3.4, we also show the performance of the proposed method with 90% training text data for different values of $\alpha$ and $\lambda$. Indeed, the graphs are the about the same as those in Figure 3.4 when 10% data is used, and the optimal parameters are the same for 10% and 90% training data to be used. It seems that this MIML problem for image example may be more difficult than that for text example.

3.4 Concluding Remarks on Sparse-MIML Algorithm

We have developed a convex sparse-MIML\footnote{For more details, please refer to [65].} model which can be solved iteratively to determine the rank of class labels for objects in multi-instance multi-label data. The experimental results have demonstrated that the proposed algorithm is efficient and effective. In the future, we would like to explore how to determine the optimal parameters in the proposed algorithm, and apply the algorithm to the other large scale MIML problems.
Figure 3.2: The performance of sparse-MIML with 10% training image data for different values $\alpha$ and $\lambda$. 
Figure 3.3: The performance of sparse-MIML with 90% training image data for different values $\alpha$ and $\lambda$. 
Figure 3.4: The performance of sparse-MIML with 90% training text data for different values $\alpha$ and $\lambda$. 
Chapter 4

The Fast Markov-chain based Learning Method

In this chapter, we mainly focus on introducing the proposed fast MIML algorithm which is designed to handle a large MIML data set effectively and efficiently. Our main idea is to construct two Markov chains which exploits relationship between objects and labels, features and labels respectively with the construction cost of the two Markov chains is significantly less than that of Sparse-MIML [65] method which has been introduced in the previous chapter.

4.1 A Fast Markov Chain Based Algorithm for Multi-instance Multi-label Learning

Similarly to the previous chapter, suppose there are multiple instances among objects in MIML data and $x^{(j)}_i$ is the d-dimensional feature vector for the $j$-th instance of the $i$-th object. The MIML data can be represented by an $d$-by-$n$ instance-feature matrix as follows:

$$X = \begin{bmatrix}
    x^{(1)}_1 \cdots x^{(1)}_{n_1} & x^{(2)}_1 \cdots x^{(2)}_{n_2} & \cdots & x^{(m)}_1 \cdots x^{(m)}_{n_m}
\end{bmatrix},$$

where $n$ is the total number of instances in the MIML data, i.e., $n = \sum_{i=1}^{m} n_i$. Each column contains the features of each instance.
4.1.1 The Markov-MIML Algorithm

In the Markov-MIML algorithm [78], the first step is to construct the affinity matrix among instances. In particular, the affinity between the \( s \)-th instance of the \( i \)-th object and the \( t \)-th instance of the \( j \)-th object is calculated as follows:

\[
a_{i,j,s,t} = \exp\left(-\frac{||x_i^{(s)} - x_j^{(t)}||^2}{2\sigma^2}\right).
\]

Here \(|| \cdot ||_2\) is the Euclidean distance and \( \sigma^2 \) is the Gaussian kernel parameter. Then the \( n \)-by-\( n \) affinity matrix \( A = [A_{ij}] \) where the \((i,j)\)-th block is an \( n_i \)-by-\( n_j \) matrix \( A_{i,j} = [a_{i,j,s,t}] \) with \( s = 1, \ldots, n_i \) and \( t = 1, \ldots, n_j \) is obtained. The instance-to-object-relation matrix is also a block diagonal matrix \( B = [B_{ij}] \) where the \((i,j)\)-th block is a zero matrix except \( i = j \). For the \((i,i)\)-th block, \( B_{i,i} \) is a 1-by-\( n_i \) matrix where all its entries are equal to 1. This block indicates the relation between the \( i \)-th object and its association instances. The size of \( B \) is \( m \)-by-\( n \). The next step is to transfer the affinity information among instances into object level by utilizing instance-to-object-relation matrix:

\[
S = BAB^T.
\]

Here \( S \) is the similarity matrix among objects which can be treated as nearest neighbors affinity for all the objects. By normalizing each column sum of \( S \) to be one, a transition probability matrix \( M \) is obtained. The Markov-MIML iterative algorithm is given as follows:

\[
P(t + 1) = (1 - \alpha)MP(t) + \alpha U, \quad t = 0, 1, 2, \cdots, \quad (4.1)
\]

where \( U = [u_1, u_2, \cdots, u_c] \), \( \alpha \) is a parameter to control the balance between the label information from neighbors and the given label training data \( U \), and

\[
P(t) = [p_1(t), p_2(t), \cdots, p_c(t)]
\]

is the label-indicator \((m \text{-by-} c) \text{ matrix}\) at the \( t \)-th iteration. Here we assume that there are \( c \) classes to be classified. Also \( u_l \) is an \( m \)-by-1 vector containing the label information of objects associated to the \( l \)-th class, and \( p_l(t) \) is an \( m \)-by-1 vector.
containing the association of objects to the $l$-th class label. Suppose the $i$-th object is labeled and it belongs to the $l$-th class, we set

$$[u_l]_i = \frac{1}{e_l}$$

(4.2)

(otherwise, it is set to be zero), and $e_l$ is the number of labeled objects belonging to the $l$-th class. The convergence analysis given in [78] demonstrate that the iterate $P(t)$ converges to a limiting vector $\hat{P}$ which can be used for MIML testing object prediction. In [78], it has been shown this algorithm is very effective, but the bottleneck of this algorithm is the construction of affinity matrix $A$. For instance, when we consider MIML data with 20,000 instances and 1024 features, the construction of affinity matrix takes a few thousand seconds.

### 4.1.2 Two Markov Chains

The main idea of the proposed method is to construct two Markov chains to represent transition probabilities from features to instances and from instances to features. More precisely, we employ object-to-instance transition matrix $B$ to transfer features from instance level to object level $Y = BX^T$ and then make use of each column sums and each row sums of $Y$ to construct an $d$-by-$d$ diagonal matrix $D_F$ and an $m$-by-$m$ diagonal matrix $D_O$ respectively. We can obtain two normalized matrices:

$$M_F = YD_F^{-1} \quad \text{and} \quad M_O = (D_O^{-1}Y)^T.$$  

(4.3)

Note that both $M_F$ is $m$-by-$d$ matrix and $M_O$ is $d$-by-$m$ matrix where each column sum of both $M_F$ and $M_O$ is equal to 1. We can interpret the $(i,j)$-th entry of $M_F$ is the transition probability of giving the $i$-th object assuming that the $j$-th feature is used, and the $(i,j)$-th entry of $M_O$ is the transition probability of using the $i$-th feature assuming that the $j$-th object is considered.

Next we derive the iterative algorithm for updating the label-indicator $P(t)$ as follows:

$$P(t + 1) = (1 - \alpha)M_FF(t) + \alpha U$$  

(4.4)

$$F(t + 1) = M_O P(t + 1)$$  

(4.5)
Here we introduce an $d$-by-$c$ relevance-indicator matrix
\[ F(t) = [f_1(t), f_2(t), \cdots, f_c(t)] \]
where $f_i(t)$ is an $d$-by-1 vector containing the relevance of features to the $l$-th class label. Note that $X$ is the matrix of original data, hence the computational cost of the construction of $M_F$ and $M_O$ is significantly lower than that of the construction of the pairwise affinity matrix $A$ in Markov-MIML algorithm. Below is the proposed Object-Feature MIML algorithm.

**Algorithm 3**

**Input:** $M_F, M_O, U, F(0)$ (or $P(0)$), $\alpha \in (0, 1)$ and tolerance $\varepsilon > 0$

**Output:** $P^*$ and $F^*$

**Procedure**

1. Set $t = 1$;
2. Compute $P(t) = (1 - \alpha)M_F F(t - 1) + \alpha U$, and $F(t) = M_O P(t)$;
3. If $\|P(t) - P(t - 1)\| < \varepsilon$ and $\|F(t) - F(t - 1)\| < \varepsilon$, set $P^* = P(t)$, $F^* = F(t)$; Otherwise set $t = t + 1$ and goto Step 2.

Next we show the convergence of $\{P(t)\}$ in the Object-Feature MIML algorithm.

**Theorem 2.** Suppose $M_F \in \mathbb{R}^{m \times d}$ and $M_O \in \mathbb{R}^{d \times m}$ are two transition probability matrices and $U$ is given by (4.2), Then there exists one unique nonnegative matrix $P^*$ such that
\[ P^* = (1 - \alpha)M_F M_O P^* + \alpha U, \]
Moreover, iterative scheme given in (4.4) and (4.5) converges for any initial matrix $P(0)$ where each column is a probability distribution vector.

**Proof.** As both $M_F$ and $M_O$ are transition probability matrices with $\sum_i [M_F]_{i,j} = 1$ and $\sum_j [M_O]_{i,j} = 1$. Let $\bar{M} = M_F M_O$, and let $[M_O]_{i,j}$ be the $j$-th column of $M_O$ and let $[M_F]_{i,:}$ be the $i$-th row of $M_F$.
\[
\sum_i [\bar{M}]_{i,j} = \sum_i [M_F]_{i,:}[M_O]_{i,j} = (\sum_i [M_F]_{i,:})[M_O]_{i,j} = 1^T [M_O]_{i,j} = 1
\]
Thus, the summation of each column of $\bar{M}$ is equal to 1. Then, by Perron-Frobenius Theorem [29,58], we can get that spectral radius of $\bar{M}$ is equal to 1 which shows the
largest eigenvalue of \((1 - \alpha)\bar{M}\) is equal to \((1 - \alpha)\). Therefore, the matrix \(I - (1 - \alpha)\bar{M}\) is invertible and \(P^* = (I - (1 - \alpha)\bar{M})^{-1}\alpha U\) is well defined. This shows that there exist one unique \(P^*\) satisfying \(P^* = (1 - \alpha)\bar{M}F + \alpha P^*\). Also consider the series of expansion of \((I - (1 - \alpha)\bar{M})^{-1}\) as follows:

\[
(I - (1 - \alpha)\bar{M})^{-1} = \sum_{k=0}^{\infty} (1 - \alpha)^k \bar{M}^k,
\]

and all the entries of \(\bar{M}\) are non-negative since both \(M_F\) and \(M_O\) are nonnegative, we get \(P^*\) are non-negative. Moreover, when \(\bar{M}\) is irreducible, there exists a positive integer \(k'\) such that the entries \(\bar{M}^{k'}\) are even positive. Since \(\sum_{k=0}^{\infty} (1 - \alpha)^k \alpha = 1\) and each column of \(\bar{M}^{k}U\) consists of probability distribution vectors, each column of \(P^*\) is also a probability distribution vector. Moreover, according to the iterative scheme in (4.4) and (4.5),

\[
\|P^* - P(t)\| = \|(1 - \alpha)\bar{M}(P^* - P(t - 1))\| < \|P^* - P(t - 1)\|
\]

due to the reason that spectral radius of \(\bar{M}\) is equal to 1, the iterative scheme converges for any initial \(P(0)\) satisfied the requirement stated in the theorem.

According to Theorem 2, the iterative scheme in (4.4) and (4.5), \(P\) will converge to a unique matrix with each column is a probability distribution vector. Similarly, the convergence of \(\{F(t)\}\) can be obtained.

**Theorem 3.** Suppose \(M_F \in \mathbb{R}^{m \times d}\) and \(M_O \in \mathbb{R}^{d \times m}\) are two transition probability matrices and \(U\) is given by (4.2), Then there exists one unique nonnegative matrix \(F^*\) such that

\[
F^* = (1 - \alpha)M_OM_FF^* + \alpha M_OU,
\]

Moreover, iterative scheme given in (4.4) and (4.5) converges for any initial matrix \(F(0)\) where each column is a probability distribution vector.

The proof of this Theorem 3 basically follows the same way of proof of Theorem 2. Theorem 3 guarantees the convergence of the iterative scheme for solving \(F\) as well as the uniqueness and nonnegativity of \(F\).

By using the Object-Feature MIML algorithm, we obtain the optimal solution \(P^*\). We can make use of \(P^*\) to rank class labels associated to each testing object. More
precisely, for the $i$-th testing object (without label information), we consider the $i$-th row of $P^*$:
\[ [P^*]_{i,1}, [P^*]_{i,2}, \cdots, [P^*]_{i,c}. \]
The larger the value of $P^*(i, k)$ is, the higher the relevancy of the $k$-th class label associated to the $i$-th object. Therefore, we can perform classification for each testing object.

### 4.1.3 Optimization Model

In this subsection, we present the above proposed Object-Feature MIML algorithm based on a convex optimization model. Firstly, we define a 2-by-2 block matrix $H$ as follows:
\[
H = \begin{bmatrix}
0 & Y \\
\frac{1}{1-\alpha} Y^T & 0
\end{bmatrix}
\]
and a diagonal matrix $D$ as follows:
\[
D = \begin{bmatrix}
D_O & 0 \\
0 & D_F
\end{bmatrix},
\]
Now we set $V$ and $\hat{V}$ by using label-indicator $P$ and relevance-indicator $F$ as follows:
\[
V = \begin{bmatrix}
P \\
F
\end{bmatrix}, \quad \hat{V} = D^{-1/2}V.
\]
Similarly, by using given labels, we set
\[
T = \begin{bmatrix}
U \\
0
\end{bmatrix}, \quad \hat{T} = D^{-1/2}T.
\]
Now we construct the following optimization problem:
\[
\min_{\hat{V}} \text{tr}(\hat{V}^T (I - (1-\alpha)D^{-\frac{1}{2}}HD^{-\frac{1}{2}}) \hat{V}) + \frac{\alpha}{1-\alpha} \| \hat{V} - \hat{T} \|_F^2 \tag{4.6}
\]
We note that the objective function of (4.6) is convex as the matrix $(I - (1-\alpha)D^{-\frac{1}{2}}HD^{-\frac{1}{2}})$ is positive semidefinite for $0 \leq \alpha \leq 1$. Also the optimal solution of (4.6) is given by
\[
\hat{V}^* = \alpha(I - (1-\alpha)D^{-\frac{1}{2}}HD^{-\frac{1}{2}})^{-1}\hat{T},
\]
i.e.,

\[ V^* = \alpha(I - (1 - \alpha)HD^{-1})^{-1}T \]
\[ V^* = (1 - \alpha)HD^{-1}V^* + \alpha T. \]

It is easy to see that when we iterate

\[ V(t + 1) = (1 - \alpha)HD^{-1}V(t) + \alpha T, \quad (4.7) \]

we obtain the above optimal solution. Indeed, (4.7) is the resulting iterative step by combining two equations in the second step of the Object-Feature MIML algorithm.

Let us study the case where \( \alpha = 0 \), i.e., there is no given label data. This is an unsupervised learning setting. We see that the objective function becomes:

\[ \min_{\hat{V}} \text{tr}(\hat{V}^T(I - D^{-\frac{1}{2}}HD^{-\frac{1}{2}})\hat{V}). \quad (4.8) \]

We note that

\[ I - D^{-\frac{1}{2}}HD^{-\frac{1}{2}} = D^{-\frac{1}{2}}(D - H)D^{-\frac{1}{2}} \]
\[ = D^{-\frac{1}{2}} \begin{bmatrix} D_O & -Y \\ -Y^T & D_F \end{bmatrix} D^{-\frac{1}{2}}. \]

Here the resulting matrix is just the normalized Laplacian matrix in spectral co-clustering algorithm [17, 18, 37]. We remark in the spectral co-clustering algorithm that the optimal solution \( \hat{V} \) is further required to be orthogonal.

### 4.1.4 Comparison of Computational Cost

We remark that the main motivation of proposed Object-Feature MIML algorithm is to solve the MIML problem more efficiently. Here we compare the computational cost of the proposed algorithm with the other existing algorithms.

1. **Object-Feature MIML:** We first compute \( Y = BX^T \), and two diagonal matrices \( D_F \) and \( D_0 \). Then two matrices \( M_F \) and \( M_0 \) are constructed. The computational cost is of \( O((m + n)d) \), where \( m \) is the number of objects, \( n \) is the number of instances and \( d \) is the number of features. According to Algorithm 3, the main cost is to perform matrix multiplications \( M_F F(t - 1) \) and \( M_O P(t) \)
in each iteration. The cost is of $O(mdc)$, where $c$ is the number of classes. Therefore, the computational cost is of $O((m + n)d + mdc \times \text{Iter})$ and $\text{Iter}$ is the number of iterations for convergence.

2. Markov-MIML: Markov-MIML algorithm also requires to construct the affinity matrix among instances with the cost of $O(n^2d)$. At each iteration of Markov-MIML algorithm, the main cost is to perform a multiplication between an $m$-by-$m$ object affinity matrix and an $m$-by-$c$ label-indicator matrix in each iteration, and it is $O(m^2c)$. We note that the cost is quadratically with $n$ and $m$. When the number of instances and the number of objects are large, the computational cost would be significantly larger than that of the proposed algorithm.

3. MIML-SVM: The $k$-medoids algorithm is applied to the set of training objects. The cost of each iteration of the $k$-medoids algorithm is of $O(r^2(m_{\text{train}} - k)^2kd)$, where we assume that $m_{\text{train}}$ is the number of labeled objects, and $r$ is the number of instances for each object. Here the work in each iteration is to calculate the Hausdroff metrics among labeled objects. After the $k$ medoids are determined, we compute the Hausdroff metrics between the medoids and the unlabeled objects, and the cost is of $O(r^2(m - m_{\text{train}})dk)$. Each object can be represented by these $k$ distances to medoids. The SVM can be employed to perform classification for unlabeled objects based on labeled objects. The training cost is of $O(m_{\text{train}}^3k)$ and the cost for classification of unlabeled objects is of $O((m - m_{\text{train}})k)$. Therefore, the cost MIML-SVM depends on the number of instances and the number of labeled objects.

4. MIML-$k$NN: Similarly, the $k$-medoids algorithm is applied to the set of training objects, and the Hausdroff metrics between the medoids and the unlabeled objects are calculated. The $k'$NN algorithm is employed for classification instead. The classification cost is of $O((m - m_{\text{train}})m_{\text{train}}k')$. Therefore, the cost of MIML-$k$NN depends on the number of instances and the number of labeled objects.
5. **MIML-Boost**: The idea of MIML-Boost is to decompose the MIML learning problem into multi-label learning problem. Therefore, MI-Boost is performed for each of possible labeled classes respectively. In the each round of boosting, the computational cost of training process is at least $O(n_{\text{train}}^3)$, where $n_{\text{train}}$ is the number of instances corresponding to labeled objects. Therefore, the boosting algorithm can take a huge amount of computational time when we deal with a large scale MIML data set.

6. **M$^3$MIML**: The algorithm involves $O(n_{\text{train}}^3)$ operations in each iteration in order to solve the optimization problem for the instance level. Thus the computational cost for training of M$^3$MIML algorithm grows in cubic order along with the increasing of the number of training instances.

According to the above discussion, we see that the proposed Object-Feature MIML algorithm is quite efficient as its cost depends on the number of instances, the number of objects and the number of features. In the next section, we will test MIML data sets to demonstrate the effectiveness and efficiency of the proposed method.

### 4.2 Experimental Results

In this section, we report the performance of the proposed algorithm and make comparisons with other algorithms. All the comparisons are performed in a computer running a server environment with 3.40GHz CPU and 8 GB memory. Since MIML algorithms make multi-label prediction, the performance is evaluated by multi-label ranking metrics. Suppose $\{x_1, x_2, \cdots, x_{m'}\}$ is a set of testing objects and $\{y_1, y_2, \cdots, y_{m'}\}$ is the set of corresponding true labels of $\{x_1, x_2, \cdots, x_{m'}\}$. As $x_i$ may belong to several classes, $y_i$ is a subset of $\{1, 2, \cdots, c\}$. The performance of multi-label prediction is evaluated by four multi-label ranking metrics: one-error, ranking loss, coverage and average precision [62]. Let $f(x_i, l)$ denote the ranking assigned to the $l$-th label for $x_i$.

(i) One-error computes how many times the top-ranked label is not relevant:

$$\text{one-error}(f) = \frac{1}{m'} \sum_{i=1}^{m'} \left| \arg \max_{1 \leq l \leq c} f(x_i, l) \notin y_i \right|,$$
where \(|.|\) equals 1 if \(\pi\) holds and 0 otherwise;

(iii) Coverage determines how far one needs to go in the list of labels to cover all the relevant labels of an instance. This measure is loosely related to the precision at the level of perfect recall:

\[
\text{coverage}(f) = \frac{1}{m'} \sum_{i=1}^{m'} \max_{l \in y_i} f(x_i, l) - 1;
\]

(iv) Average precision determines for each relevant label \(l \in y_i\) the percentage of relevant labels among all labels that are ranked above it, and averages these percentages over all relevant labels:

\[
\text{avgprec}(f) = \frac{1}{m'} \sum_{i=1}^{m'} \frac{1}{|y_i|} \sum_{l \in y_i} \frac{|\{l'|f(x_i, l') \leq f(x_i, l), l' \in y_i\}|}{f(x_i, l)}.
\]

For the metrics (i), (ii) and (iii), the smaller the value is and the better the performance is. For the metric (iv), the larger the value is and the better the performance is. All the classification results and computational time below are given in the average of ten runs under the same experimental setting.

### 4.2.1 Experiment 1

The first experiment is on a text data set which is widely used Reuters-21578 text collection in text categorization \(^1\). The data set contains 2,000 documents with seven classes. Each document is represented as a bag of instances based on the techniques of sliding windows and each instance corresponds to a text segment enclosed in a sliding window \([1]\). There are 7119 instances in total, and each instance has 243 features and its feature vector is normalized to be 1 \([88]\). Also there are 1701, 290 and 9 objects belonging to one, two and being more than three classes respectively.

\(^1\)available at http://lamda.nju.edu.cn/datacode/miml-text-data.htm
In this thesis, we give the comparisons of Object-Feature MIML (OF-MIML) with Markov-MIML, M3MIML, MIML-SVM and MIML-kNN. Note that MIML-Boost takes a huge amount of computational time in order to obtain the results, we do not make a comparison with it. Neighbor size of 50 is set in Markov-MIML and parameter which is used to control the significance of training label information is set to be 0.99 as recommended in [78]. For the M3MIML algorithm, the value of $C$ (cost parameter) and $\gamma$ of M3MIML are set to the default value of 1.0 as given in [88]. It is reported in [88] that M3MIML shows similar performance with $\gamma$ in [0.6, 1.4], and the mean value over this range is used as the default value. The number of iterations to find the solution of dual variables in M3MIML algorithm by Franke and Wolfe method [28] is set to be 50. For MIML-SVM, the optimal parameter settings as reported in [90] are used. More precisely, the Gaussian kernel parameter of MIML-SVM is set to be 0.2. We also evaluate the performance of $k$ nearest neighbor type algorithms for MIML learning. By considering replace the SVM in MIML-SVM algorithm by ML-kNN [87], we extend ML-kNN to MIML-kNN. Here, ML-kNN is parameterized by the size of neighborhood, and the number of nearest neighbors is set to be 10 which is the recommended value given in [87] to obtain the best performance. Experimental results across different sizes of training data (from 10% to 90%) for all the algorithms mentioned. The convergence criteria we used in the experiments is $\|P(t + 1) - P(t)\|_F^2/\|P(t)\|_F^2 \leq 1 \times 10^{-15}$.

In Table 4.1, we report the performance of proposed method as well as the other methods when 90% of training data is used. The classification performance of the OF-MIML algorithm is quite good even it is not best. However, the computational time of the proposed algorithm is significantly less than those by the other methods. In Figure 4.1, we further plot the classification results of different algorithms when the percentages of training data are from 10% to 80%. We see that the results by the OF-MIML algorithm are competitive with those by the other methods. However, the computational time required by the proposed algorithm is significantly less than those by the other methods, see Table 4.2. Also it is interesting to note that the computational time of OF-MIML and Markov-MIML are independent of the number of training samples. The initialization cost of Markov-MIML is very large as the
<table>
<thead>
<tr>
<th>Algorithm</th>
<th>one-error</th>
<th>coverage</th>
<th>rank-loss</th>
<th>1-avgprec</th>
<th>time (seconds)</th>
</tr>
</thead>
<tbody>
<tr>
<td>OF-MIML</td>
<td>0.089</td>
<td>0.357</td>
<td>0.032</td>
<td>0.056</td>
<td><strong>0.30</strong></td>
</tr>
<tr>
<td>Markov-MIML</td>
<td>0.093</td>
<td><strong>0.338</strong></td>
<td><strong>0.028</strong></td>
<td>0.055</td>
<td>170.5</td>
</tr>
<tr>
<td>M$^3$MIML</td>
<td><strong>0.079</strong></td>
<td>0.352</td>
<td><strong>0.028</strong></td>
<td><strong>0.051</strong></td>
<td>11390.0</td>
</tr>
<tr>
<td>MIML-SVM</td>
<td>0.096</td>
<td>0.350</td>
<td>0.030</td>
<td>0.058</td>
<td>135.2</td>
</tr>
<tr>
<td>MIML-KNN</td>
<td>0.138</td>
<td>0.464</td>
<td>0.049</td>
<td>0.087</td>
<td>126.8</td>
</tr>
</tbody>
</table>

Table 4.1: The classification performance of different algorithms with 90% training data.

<table>
<thead>
<tr>
<th>training data</th>
<th>OF-MIML</th>
<th>Markov-MIML</th>
<th>M$^3$MIML</th>
<th>MIML-SVM</th>
<th>MIML-KNN</th>
</tr>
</thead>
<tbody>
<tr>
<td>10%</td>
<td><strong>0.3</strong></td>
<td>170.5</td>
<td>367.0</td>
<td>7.9</td>
<td>8.3</td>
</tr>
<tr>
<td>20%</td>
<td><strong>0.3</strong></td>
<td>170.5</td>
<td>954.0</td>
<td>17.2</td>
<td>17.4</td>
</tr>
<tr>
<td>30%</td>
<td><strong>0.3</strong></td>
<td>170.5</td>
<td>1737.0</td>
<td>27.6</td>
<td>27.7</td>
</tr>
<tr>
<td>40%</td>
<td><strong>0.3</strong></td>
<td>170.5</td>
<td>2398.0</td>
<td>42.7</td>
<td>41.0</td>
</tr>
<tr>
<td>50%</td>
<td><strong>0.3</strong></td>
<td>170.5</td>
<td>3987.0</td>
<td>58.9</td>
<td>54.5</td>
</tr>
<tr>
<td>60%</td>
<td><strong>0.3</strong></td>
<td>170.5</td>
<td>5756.0</td>
<td>76.0</td>
<td>77.2</td>
</tr>
<tr>
<td>70%</td>
<td><strong>0.3</strong></td>
<td>170.5</td>
<td>7043.0</td>
<td>97.5</td>
<td>84.9</td>
</tr>
<tr>
<td>80%</td>
<td><strong>0.3</strong></td>
<td>170.5</td>
<td>8480.0</td>
<td>120.3</td>
<td>107.6</td>
</tr>
</tbody>
</table>

Table 4.2: The computational time (seconds) of different algorithms.

The affinity matrix is required to construct. The training cost of the M$^3$MIML, MIML-SVM and ML-$k$NN depends on the number of training samples.

### 4.2.2 Experiment 2

The data used in the previous experiment is a small data set. We would like to test the performance of the proposed algorithm as well as making comparison with the other state-of-art methods for larger data sets. Our idea is to simulate a large data set by using the data used in Experiment 1. More specifically, we combine all the bags together, and construct histograms of the features in each class. Figure 4.2 shows these histograms of seven classes. Then we extract twenty features (about 8%) where their numbers are the highest among all the features in the histogram. The next step
Figure 4.1: The classification performance of different algorithms with respect to the percentage of training samples.

is to generate new multi-label data bags which contain multiple instances. In the original text data, each bag at most can be categorized into 3 different classes. Here we randomly assign at most 3 labels to each bag. The number of instances contained in each bag are chosen randomly from 2 to 26 (2 is the minimum instance number of the text data while 26 is the maximum number of the text data). For each instance in the data bag, we need to generate the values of the selected features of the assigned labels in this bag. Note that the integer value can be randomly generated in the range being the same as those range of feature values in the original data set. For the other features, we randomly select 20% of them and assign their integer values randomly in between 1 and 50. According to such simulation process, we can generate additional bags for the performance evaluation.

In this experiment, we first simulate two data sets of containing 1,000 bags and 2,000 bags respectively. Then we add them into the original data set containing 2,000 bags, and therefore we have two generated data sets of containing 3,000 bags with 11,154 instances and 4,000 bags with 15,004 instances. In Tables 4.3 and 4.4,
<table>
<thead>
<tr>
<th>Feature</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>Class One</td>
<td>0, 1000, 2000, 3000, 4000, 5000, 6000, 7000</td>
</tr>
<tr>
<td>Class Two</td>
<td>0, 500, 1000, 1500, 2000, 2500</td>
</tr>
<tr>
<td>Class Three</td>
<td>0, 200, 400, 600, 800, 1000, 1200, 1400, 1600, 1800, 2000</td>
</tr>
<tr>
<td>Class Four</td>
<td>0, 200, 400, 600, 800, 1000, 1200, 1400, 1600, 1800, 2000</td>
</tr>
<tr>
<td>Class Five</td>
<td>0, 200, 400, 600, 800, 1000, 1200, 1400, 1600, 1800, 2000</td>
</tr>
<tr>
<td>Class Six</td>
<td>0, 200, 400, 600, 800, 1000, 1200, 1400, 1600, 1800, 2000, 2500, 3000</td>
</tr>
<tr>
<td>Class Seven</td>
<td>0, 500, 1000, 1500, 2000, 2500, 3000</td>
</tr>
</tbody>
</table>

Figure 4.2: The histogram of features for 7 different classes.
Table 4.3: The classification performance of different algorithms for enlarged text data set containing 3000 bags with 90% training samples.

<table>
<thead>
<tr>
<th>Algorithm</th>
<th>one-error</th>
<th>coverage</th>
<th>rankLoss</th>
<th>1-avgprec</th>
<th>time (seconds)</th>
</tr>
</thead>
<tbody>
<tr>
<td>OF-MIML</td>
<td>0.079</td>
<td>0.326</td>
<td>0.029</td>
<td>0.053</td>
<td>0.7</td>
</tr>
<tr>
<td>Markov-MIML</td>
<td>0.090</td>
<td>0.333</td>
<td>0.029</td>
<td>0.056</td>
<td>479.4</td>
</tr>
<tr>
<td>M₃MIML</td>
<td>0.126</td>
<td>0.423</td>
<td>0.046</td>
<td>0.083</td>
<td>22220.8</td>
</tr>
<tr>
<td>MIML-SVM</td>
<td>0.083</td>
<td>0.333</td>
<td>0.030</td>
<td>0.055</td>
<td>161.4</td>
</tr>
<tr>
<td>MIML-KNN</td>
<td>0.110</td>
<td>0.394</td>
<td>0.039</td>
<td>0.071</td>
<td>187.9</td>
</tr>
</tbody>
</table>

Table 4.4: The classification performance of different algorithms for enlarged text data set containing 4000 bags with 90% training samples.

<table>
<thead>
<tr>
<th>Algorithm</th>
<th>one-error</th>
<th>coverage</th>
<th>rankLoss</th>
<th>1-avgprec</th>
<th>time (seconds)</th>
</tr>
</thead>
<tbody>
<tr>
<td>OF-MIML</td>
<td>0.053</td>
<td>0.260</td>
<td>0.019</td>
<td>0.037</td>
<td>1.3</td>
</tr>
<tr>
<td>Markov-MIML</td>
<td>0.065</td>
<td>0.263</td>
<td>0.021</td>
<td>0.042</td>
<td>780.1</td>
</tr>
<tr>
<td>M₃MIML</td>
<td>0.137</td>
<td>0.287</td>
<td>0.032</td>
<td>0.075</td>
<td>39927.1</td>
</tr>
<tr>
<td>MIML-SVM</td>
<td>0.060</td>
<td>0.269</td>
<td>0.021</td>
<td>0.040</td>
<td>281.4</td>
</tr>
<tr>
<td>MIML-KNN</td>
<td>0.082</td>
<td>0.336</td>
<td>0.031</td>
<td>0.055</td>
<td>312.5</td>
</tr>
</tbody>
</table>

we report the performance of proposed algorithm when 90% data are employed for training to make the comparison with other state-of-art algorithms. It is clear that the proposed OF-MIML algorithm improve the classification results while reducing the running time significantly. The running time required for other methods is at least 161.4 seconds and 281.4 seconds with respect to two simulated data sets while the proposed OF-MIML only takes 0.7 second and 1.3 seconds accordingly. In Figures 4.3 and 4.4, we plot the classification results for different methods with the percentages of training data varying from 10% to 80%. The proposed OF-MIML algorithm usually outperforms the others for the simulated data set of 4,000 bags. The results are still competitive on the simulated data set of 3,000 bags. In Tables 4.5 and 4.6, we show the computational time required for different methods with respect to different percentages of training data. We see that the computational time for OF-MIML is independent of the number of training data and the results further illustrate the efficiency of proposed algorithm.
<table>
<thead>
<tr>
<th>training data</th>
<th>OF-MIML</th>
<th>Markov-MIML</th>
<th>M³MIML</th>
<th>MIML-SVM</th>
<th>MIML-KNN</th>
</tr>
</thead>
<tbody>
<tr>
<td>10%</td>
<td>0.7</td>
<td>479.4</td>
<td>408.4</td>
<td>9.1</td>
<td>9.8</td>
</tr>
<tr>
<td>20%</td>
<td>0.7</td>
<td>479.4</td>
<td>1375.7</td>
<td>19.0</td>
<td>21.0</td>
</tr>
<tr>
<td>30%</td>
<td>0.7</td>
<td>479.4</td>
<td>2675.1</td>
<td>32.3</td>
<td>34.7</td>
</tr>
<tr>
<td>40%</td>
<td>0.7</td>
<td>479.4</td>
<td>4475.1</td>
<td>48.3</td>
<td>53.2</td>
</tr>
<tr>
<td>50%</td>
<td>0.7</td>
<td>479.4</td>
<td>6683.3</td>
<td>65.4</td>
<td>70.6</td>
</tr>
<tr>
<td>60%</td>
<td>0.7</td>
<td>479.4</td>
<td>9933.6</td>
<td>84.5</td>
<td>95.1</td>
</tr>
<tr>
<td>70%</td>
<td>0.7</td>
<td>479.4</td>
<td>13521.5</td>
<td>107.3</td>
<td>112.5</td>
</tr>
<tr>
<td>80%</td>
<td>0.7</td>
<td>479.4</td>
<td>17379.9</td>
<td>131.9</td>
<td>148.3</td>
</tr>
</tbody>
</table>

Table 4.5: The computational time (seconds) of different algorithms for text data set containing 3000 bags.

<table>
<thead>
<tr>
<th>training data</th>
<th>OF-MIML</th>
<th>Markov-MIML</th>
<th>M³MIML</th>
<th>MIML-SVM</th>
<th>MIML-KNN</th>
</tr>
</thead>
<tbody>
<tr>
<td>10%</td>
<td>1.3</td>
<td>780.1</td>
<td>810.8</td>
<td>10.5</td>
<td>13.0</td>
</tr>
<tr>
<td>20%</td>
<td>1.3</td>
<td>780.1</td>
<td>2513.6</td>
<td>24.7</td>
<td>29.9</td>
</tr>
<tr>
<td>30%</td>
<td>1.3</td>
<td>780.1</td>
<td>5188.9</td>
<td>46.6</td>
<td>54.3</td>
</tr>
<tr>
<td>40%</td>
<td>1.3</td>
<td>780.1</td>
<td>8572.1</td>
<td>70.8</td>
<td>82.8</td>
</tr>
<tr>
<td>50%</td>
<td>1.3</td>
<td>780.1</td>
<td>12931.1</td>
<td>102.0</td>
<td>115.1</td>
</tr>
<tr>
<td>60%</td>
<td>1.3</td>
<td>780.1</td>
<td>17793.7</td>
<td>140.8</td>
<td>156.2</td>
</tr>
<tr>
<td>70%</td>
<td>1.3</td>
<td>780.1</td>
<td>23400.5</td>
<td>180.6</td>
<td>200.7</td>
</tr>
<tr>
<td>80%</td>
<td>1.3</td>
<td>780.1</td>
<td>30260.0</td>
<td>229.1</td>
<td>259.4</td>
</tr>
</tbody>
</table>

Table 4.6: The computational time (seconds) of different algorithms for text data set containing 4000 bags.
Next we conduct experiments on larger data sets. Here we simulate 6,000 bags (23,242 instances), 8,000 bags (31,141 instances), 10,000 bags (38,984 instances), 15,000 bags (59,184 instances), 20,000 bags (79,216 instances), 30,000 bags (119,188 instances) and 40,000 bags (159,268 instances). We find that the computational time required for $M^3$MIML exceeds 12 hours when the data is set more than 4,000 bags. For Markov-MIML, the storage requirement of the algorithm is also beyond the ability of our computer when handling data being more than 6,000 bags. Both MIML-SVM and MIML-KNN cannot handle the data sets being more than 20,000 bags due to the computer storage limit or time limit (12 hours). It is interesting to note that the proposed algorithm takes around 50 and 107 seconds for data sets with 30,000 bags and 40,000 bags respectively. The detailed results can be found in Figures 4.5 and 4.6. Also the proposed algorithm outperforms the other methods across different evaluation metrics in most cases.
Figure 4.4: The classification performance of different algorithms with respect to the percentage of training samples for text data set containing 4000 bags.

Figure 4.5: Test 3: the computational time as well as the classification performance of different algorithms with respect to data sets of different sizes.
Figure 4.6: Test 3: the classification performance of different algorithms with respect to data sets of different sizes.

4.2.3 Experiment 3

In this subsection, we test a large MIML data to evaluate the performance of different MIML algorithms. We generate MIML image data from PascalVOC2007 [25]. The raw image data contains 20 classes and 9963 images with multi-label. We divide each image (object) into 25 patches (instances), and the SIFT technique [80, 81] is used to extract a 1024 feature vector of each patch. Next construct two MIML data sets as follows: 5-class data with each class having 160 objects and 10-class data with each class having 80 objects. We keep the number of instances in these two MIML data sets to be 20,000. We compare the classification performance and computational time of different algorithms. Here we set the value of $C$ (cost parameter) is set to be $1 \times 10^{-2}$ to make sure the convergence for the M$^2$MIML algorithm.

Tables 4.7 and 4.8 shows the classification results of different algorithms for these two MIML data sets. We see the OF-MIML algorithm performs quite well, and the computational time required is also less than those by the other methods. Tables 4.9
Table 4.7: The classification performance of different algorithms for 5-class data with 90% training samples.

<table>
<thead>
<tr>
<th>Algorithm</th>
<th>one-error</th>
<th>coverage</th>
<th>rankLoss</th>
<th>1-avgprec</th>
<th>time (seconds)</th>
</tr>
</thead>
<tbody>
<tr>
<td>OF-MIML</td>
<td>0.530</td>
<td>1.359</td>
<td>0.267</td>
<td>0.328</td>
<td>0.62</td>
</tr>
<tr>
<td>Markov-MIML</td>
<td>0.679</td>
<td>1.869</td>
<td>0.400</td>
<td>0.441</td>
<td>6022.0</td>
</tr>
<tr>
<td>M³MIML</td>
<td>0.589</td>
<td>2.093</td>
<td>0.430</td>
<td>0.437</td>
<td>8372.0</td>
</tr>
<tr>
<td>MIML-SVM</td>
<td>0.540</td>
<td>1.578</td>
<td>0.316</td>
<td>0.362</td>
<td>822.0</td>
</tr>
<tr>
<td>MIML-KNN</td>
<td>0.541</td>
<td>1.598</td>
<td>0.320</td>
<td>0.362</td>
<td>803.0</td>
</tr>
</tbody>
</table>

Table 4.8: The classification performance of different algorithms for 10-class data with 90% training samples.

<table>
<thead>
<tr>
<th>Algorithm</th>
<th>one-error</th>
<th>coverage</th>
<th>rankLoss</th>
<th>1-avgprec</th>
<th>time (seconds)</th>
</tr>
</thead>
<tbody>
<tr>
<td>OF-MIML</td>
<td>0.613</td>
<td>2.820</td>
<td>0.245</td>
<td>0.454</td>
<td>0.65</td>
</tr>
<tr>
<td>Markov-MIML</td>
<td>0.766</td>
<td>3.672</td>
<td>0.344</td>
<td>0.564</td>
<td>6168.0</td>
</tr>
<tr>
<td>M³MIML</td>
<td>0.622</td>
<td>5.103</td>
<td>0.469</td>
<td>0.580</td>
<td>15139.0</td>
</tr>
<tr>
<td>MIML-SVM</td>
<td>0.584</td>
<td>3.072</td>
<td>0.262</td>
<td>0.458</td>
<td>811.0</td>
</tr>
<tr>
<td>MIML-KNN</td>
<td>0.598</td>
<td>3.109</td>
<td>0.273</td>
<td>0.472</td>
<td>801.0</td>
</tr>
</tbody>
</table>

and 4.10 further compare the computational time of different algorithms for different percentages of training samples. We see that the OF-MIML algorithm is very efficient. For example, it takes less than one second for processing 5-class and 10-class MIML data sets, while the Markov-MIML and M³MIML takes a few thousand seconds the MIML-SVM and MIML-kNN requires a few hundred seconds. The computational time of OF-MIML algorithm is again independent of number of training samples. Also the OF-MIML algorithm is effective for classification of MIML data with different percentages of training samples, see Figures 4.7 and 4.8. In most cases (except one-error in 10-class data set), the performance of the OF-MIML algorithm is better than those of the other methods in the four evaluation metrics.
### Table 4.9: The computational time (seconds) of different algorithms for 5-class data.

<table>
<thead>
<tr>
<th>training data</th>
<th>OF-MIML</th>
<th>Markov-MIML</th>
<th>M^2MIML</th>
<th>MIML-SVM</th>
<th>MIML-KNN</th>
</tr>
</thead>
<tbody>
<tr>
<td>10%</td>
<td>0.62</td>
<td>6022.0</td>
<td>503.0</td>
<td>41.0</td>
<td>42.0</td>
</tr>
<tr>
<td>20%</td>
<td>0.62</td>
<td>6022.0</td>
<td>1078.0</td>
<td>64.0</td>
<td>98.0</td>
</tr>
<tr>
<td>30%</td>
<td>0.62</td>
<td>6022.0</td>
<td>1733.0</td>
<td>160.0</td>
<td>166.0</td>
</tr>
<tr>
<td>40%</td>
<td>0.62</td>
<td>6022.0</td>
<td>2511.0</td>
<td>234.0</td>
<td>245.0</td>
</tr>
<tr>
<td>50%</td>
<td>0.62</td>
<td>6022.0</td>
<td>3367.0</td>
<td>319.0</td>
<td>335.0</td>
</tr>
<tr>
<td>60%</td>
<td>0.62</td>
<td>6022.0</td>
<td>3898.0</td>
<td>409.0</td>
<td>444.0</td>
</tr>
<tr>
<td>70%</td>
<td>0.62</td>
<td>6022.0</td>
<td>6008.0</td>
<td>514.0</td>
<td>556.0</td>
</tr>
<tr>
<td>80%</td>
<td>0.62</td>
<td>6022.0</td>
<td>7467.0</td>
<td>643.0</td>
<td>679.0</td>
</tr>
</tbody>
</table>

### Table 4.10: The computational time (seconds) of different algorithms for 10-class data.

<table>
<thead>
<tr>
<th>training data</th>
<th>OF-MIML</th>
<th>Markov-MIML</th>
<th>M^2MIML</th>
<th>MIML-SVM</th>
<th>MIML-KNN</th>
</tr>
</thead>
<tbody>
<tr>
<td>10%</td>
<td>0.65</td>
<td>6168.0</td>
<td>712.0</td>
<td>43.0</td>
<td>43.0</td>
</tr>
<tr>
<td>20%</td>
<td>0.65</td>
<td>6168.0</td>
<td>1463.0</td>
<td>99.0</td>
<td>99.0</td>
</tr>
<tr>
<td>30%</td>
<td>0.65</td>
<td>6168.0</td>
<td>3831.0</td>
<td>168.0</td>
<td>165.0</td>
</tr>
<tr>
<td>40%</td>
<td>0.65</td>
<td>6168.0</td>
<td>4438.0</td>
<td>248.0</td>
<td>244.0</td>
</tr>
<tr>
<td>50%</td>
<td>0.65</td>
<td>6168.0</td>
<td>6031.0</td>
<td>343.0</td>
<td>334.0</td>
</tr>
<tr>
<td>60%</td>
<td>0.65</td>
<td>6168.0</td>
<td>7498.0</td>
<td>440.0</td>
<td>434.0</td>
</tr>
<tr>
<td>70%</td>
<td>0.65</td>
<td>6168.0</td>
<td>10167.0</td>
<td>547.0</td>
<td>547.0</td>
</tr>
<tr>
<td>80%</td>
<td>0.65</td>
<td>6168.0</td>
<td>13361.0</td>
<td>672.0</td>
<td>671.0</td>
</tr>
</tbody>
</table>

Table 4.9: The computational time (seconds) of different algorithms for 5-class data.

Table 4.10: The computational time (seconds) of different algorithms for 10-class data.
Figure 4.7: Test 3: the classification performance of different algorithms with respect to the percentage of training samples for 5-class data.

4.2.4 Experiment 4

We report experimental results on two larger sub data sets of PascalVOC2007 in this subsection. The first sub data set includes 20 classes and 80 bags for each class. The second sub data set is even larger which includes 160 bags for each of the 20 classes. For simplicity, we fix 10% and 90% training data for these two data sets and only report results of the algorithms which can solve the label prediction problem within 1000 seconds. In the experiments, storage requirement of affinity matrix in Markov-MIML algorithm is beyond capacity of memory, thus it's not possible to utilize Markov-MIML algorithm to deal these two data sets. Moreover, even only 10% of data are selected for training, M³MIML fails to get any result within 1000 seconds for either of 2 data sets we consider in this subsection. Thus, it is also not practical to use M³MIML to handle such kind of large problems. Hence, we only show classification results MIML-SVM and MIML-kNN when using 10% of data for training. Moreover, we also report the classification performance of proposed
Figure 4.8: Test 3: the classification performance of different algorithms with respect to the percentage of training samples for 10-class data.

algorithm selecting varies percentage of training from 10%-90% in Table 4.11 and Table 4.12. It’s easy to see that proposed method outperforms the other methods on evaluation metric like coverage, ranking loss and average precision. More importantly, the computational time of OF-MIML is much less than the other methods.

According to previous experimental results, when about 800 bags (or objects) and 20,000 instances involved in the problem, the computational time for Markov-MIML and M^3MIML requires more than 5,000 seconds. Moreover, for M^3MIML, MIML-SVM and MIML-kNN, costs for ranking problem grow significantly when larger number of samples are selected as training data. Thus it may not be practical to use these methods to handle larger data set considering the time issue. When dealing with complete PascalVOC2007 data set which consists of approximately 10,000 bags and 250,000 instances, it’s too time consuming to test the performance of all these algorithms. Therefore, we only report the performance of proposed Object-Feature MIML method on PascalVOC2007 data set. The results reported in Table 4.13 give the performance evaluation of proposed method on the whole PascalVOC2007 data.
### Table 4.11: The classification performance of different algorithms for 20-class data with 80 bags in each class.

<table>
<thead>
<tr>
<th>training data</th>
<th>algorithm</th>
<th>one-error</th>
<th>coverage</th>
<th>rankLoss</th>
<th>1-avgprec</th>
<th>time (seconds)</th>
</tr>
</thead>
<tbody>
<tr>
<td>10%</td>
<td>MIML-SVM</td>
<td>0.714</td>
<td>9.464</td>
<td>0.379</td>
<td>0.658</td>
<td>165</td>
</tr>
<tr>
<td>10%</td>
<td>MIML-KNN</td>
<td><strong>0.708</strong></td>
<td>9.192</td>
<td>0.358</td>
<td>0.628</td>
<td>169</td>
</tr>
<tr>
<td>10%</td>
<td>OF-MIML</td>
<td>0.786</td>
<td>6.820</td>
<td>0.276</td>
<td>0.628</td>
<td><strong>2.0</strong></td>
</tr>
<tr>
<td>20%</td>
<td>OF-MIML</td>
<td>0.772</td>
<td>6.542</td>
<td>0.259</td>
<td>0.616</td>
<td><strong>2.0</strong></td>
</tr>
<tr>
<td>30%</td>
<td>OF-MIML</td>
<td>0.764</td>
<td>6.454</td>
<td>0.255</td>
<td>0.607</td>
<td><strong>2.0</strong></td>
</tr>
<tr>
<td>40%</td>
<td>OF-MIML</td>
<td>0.766</td>
<td>6.355</td>
<td>0.250</td>
<td>0.605</td>
<td><strong>2.0</strong></td>
</tr>
<tr>
<td>50%</td>
<td>OF-MIML</td>
<td>0.757</td>
<td>6.327</td>
<td>0.248</td>
<td>0.600</td>
<td><strong>2.0</strong></td>
</tr>
<tr>
<td>60%</td>
<td>OF-MIML</td>
<td>0.763</td>
<td>6.368</td>
<td>0.248</td>
<td>0.602</td>
<td><strong>2.0</strong></td>
</tr>
<tr>
<td>70%</td>
<td>OF-MIML</td>
<td>0.749</td>
<td>6.254</td>
<td>0.244</td>
<td><strong>0.594</strong></td>
<td><strong>2.0</strong></td>
</tr>
<tr>
<td>80%</td>
<td>OF-MIML</td>
<td>0.776</td>
<td>6.328</td>
<td>0.250</td>
<td>0.609</td>
<td><strong>2.0</strong></td>
</tr>
<tr>
<td>90%</td>
<td>OF-MIML</td>
<td>0.753</td>
<td>6.169</td>
<td><strong>0.242</strong></td>
<td>0.596</td>
<td><strong>2.0</strong></td>
</tr>
</tbody>
</table>

### Table 4.12: The classification performance of different algorithms for 20-class data with 160 bags in each class.

<table>
<thead>
<tr>
<th>training data</th>
<th>algorithm</th>
<th>one-error</th>
<th>coverage</th>
<th>rankLoss</th>
<th>1-avgprec</th>
<th>time (seconds)</th>
</tr>
</thead>
<tbody>
<tr>
<td>10%</td>
<td>MIML-SVM</td>
<td>0.662</td>
<td>8.445</td>
<td>0.329</td>
<td>0.620</td>
<td>606</td>
</tr>
<tr>
<td>10%</td>
<td>MIML-KNN</td>
<td><strong>0.639</strong></td>
<td>8.384</td>
<td>0.326</td>
<td>0.613</td>
<td>632</td>
</tr>
<tr>
<td>10%</td>
<td>OF-MIML</td>
<td>0.774</td>
<td>6.543</td>
<td>0.263</td>
<td>0.613</td>
<td><strong>6.6</strong></td>
</tr>
<tr>
<td>20%</td>
<td>OF-MIML</td>
<td>0.760</td>
<td>6.318</td>
<td>0.252</td>
<td>0.600</td>
<td><strong>6.6</strong></td>
</tr>
<tr>
<td>30%</td>
<td>OF-MIML</td>
<td>0.750</td>
<td>6.236</td>
<td>0.247</td>
<td>0.595</td>
<td><strong>6.6</strong></td>
</tr>
<tr>
<td>40%</td>
<td>OF-MIML</td>
<td>0.746</td>
<td>6.174</td>
<td>0.244</td>
<td>0.590</td>
<td><strong>6.6</strong></td>
</tr>
<tr>
<td>50%</td>
<td>OF-MIML</td>
<td>0.748</td>
<td>6.195</td>
<td>0.245</td>
<td>0.502</td>
<td><strong>6.6</strong></td>
</tr>
<tr>
<td>60%</td>
<td>OF-MIML</td>
<td>0.748</td>
<td>6.168</td>
<td>0.242</td>
<td>0.590</td>
<td><strong>6.6</strong></td>
</tr>
<tr>
<td>70%</td>
<td>OF-MIML</td>
<td>0.752</td>
<td>6.173</td>
<td>0.245</td>
<td>0.594</td>
<td><strong>6.6</strong></td>
</tr>
<tr>
<td>80%</td>
<td>OF-MIML</td>
<td>0.749</td>
<td>6.105</td>
<td><strong>0.240</strong></td>
<td><strong>0.580</strong></td>
<td><strong>6.6</strong></td>
</tr>
<tr>
<td>90%</td>
<td>OF-MIML</td>
<td>0.757</td>
<td>6.210</td>
<td>0.245</td>
<td>0.597</td>
<td><strong>6.6</strong></td>
</tr>
</tbody>
</table>

69
<table>
<thead>
<tr>
<th>training data</th>
<th>one-error</th>
<th>coverage</th>
<th>rankLoss</th>
<th>1-avgprec</th>
<th>time (seconds)</th>
</tr>
</thead>
<tbody>
<tr>
<td>10%</td>
<td>0.842</td>
<td>6.499</td>
<td>0.280</td>
<td>0.671</td>
<td>71.5</td>
</tr>
<tr>
<td>20%</td>
<td>0.813</td>
<td>6.262</td>
<td>0.266</td>
<td>0.647</td>
<td>71.5</td>
</tr>
<tr>
<td>30%</td>
<td>0.816</td>
<td>6.275</td>
<td>0.266</td>
<td>0.649</td>
<td>71.5</td>
</tr>
<tr>
<td>40%</td>
<td>0.814</td>
<td>6.232</td>
<td>0.264</td>
<td>0.649</td>
<td>71.5</td>
</tr>
<tr>
<td>50%</td>
<td>0.813</td>
<td>6.224</td>
<td>0.264</td>
<td>0.648</td>
<td>71.5</td>
</tr>
<tr>
<td>60%</td>
<td>0.821</td>
<td>6.296</td>
<td>0.268</td>
<td>0.656</td>
<td>71.5</td>
</tr>
<tr>
<td>70%</td>
<td>0.801</td>
<td>6.169</td>
<td>0.260</td>
<td>0.639</td>
<td>71.5</td>
</tr>
<tr>
<td>80%</td>
<td>0.809</td>
<td>6.082</td>
<td>0.256</td>
<td>0.639</td>
<td>71.5</td>
</tr>
<tr>
<td>90%</td>
<td>0.822</td>
<td>6.113</td>
<td>0.263</td>
<td>0.650</td>
<td>71.5</td>
</tr>
</tbody>
</table>

Table 4.13: The performance of Object-Feature MIML with different sizes of training data on complete PascalVOC2007 data set.

It’s easy to find the computational times of our Object-Feature MIML method are still acceptable (about 71 seconds).

### 4.3 Concluding Remarks on OF-MIML Algorithm

In this work, we have proposed a novel Object-Feature MIML algorithm to determine the rank of labels for multi-label multi-instance data. The experimental results demonstrate that proposed algorithm is very efficient than the other MIML algorithms with competitive classification results. For many practical problems with large MIML data sets, it is not feasible to use existing MIML learning methods due to a large computational time or a huge memory storage. However, by employing the idea of Markov chain learning, Object-Feature MIML learning algorithm instead can handle such large MIML data successfully. In the future, we may consider some meaningful regularization in order to improve the classification performance of Object-Feature MIML algorithm. For instance, similar to Sparse-MIML (see [65]) introduced above, we may employ sparsity constraints. Here sparsity constraints can be introduced in the labels indicator of the objective function such that relevant and irrelevant objects with respect to a given class can be distinguished. The resulting sparsity-based MIML model can be given as a constrained convex optimization...
problem:

$$\min_{\hat{V}} \quad \text{tr}(\hat{V}^T(I - D^{-\frac{1}{2}}HD^{-\frac{1}{2}})\hat{V}) + \mu\|\hat{V} - \hat{T}\|_F^2 + \lambda \sum_{i=1}^{c} \sqrt{\sum_{i=1}^{m+d} [\hat{V}]_{i,l}^2}$$

where $\mu$ and $\lambda$ are two positive regularization parameters. The first term of the right-hand side in the above cost function is the smoothness constraint, which means that a proper labels indicator $\hat{V}$ should not change too much between neighbor objects. To minimize the first term, we expect that if the $i$-th and $j$-th objects are close $[\hat{V}]_{i,\cdot}$ and $[\hat{V}]_{j,\cdot}$ are also close to each other. The second term is the fitting constraint, which means that a good labels indicator should not change too much from the initial label assignment. The third term is the sparsity requirement, which means that each object labels indicator should be corresponding to a few categories. Because of the sparsity constraint, both intra-class (column vector of $\hat{V}$ and inter-class (row vector of $\hat{V}$) label rankings are considered together in the model. The trade-off between these three terms is captured by two positive parameters $\mu$ and $\lambda$. We remark that the optimization problem [65] can be solved very efficiently by using the augmented Lagrangian method.
Chapter 5

Application to Module Structure Learning

The main contribution of this chapter is to propose and develop a novel tensor based algorithm which integrates multiple networks together for module learning task. In the following sections, detailed methodology of the proposed approach is introduced. In addition, we conduct experiments of the proposed method on real world gene data sets. Finally, concluding remarks on the proposed methods are made.

5.1 Multiple Networks Modules Identification by a Multi-dimensional Markov Chain Method

The main reason that we try to identify module structure in multiple networks is the uncertainty in constructing network. Although biological network is obviously among the most powerful tools for analyzing relations between small biological molecules, several drawbacks of it should also be big concern of us. In general, one substantial step in construction of network should be measuring similarity among different molecules. Even a large amount of approaches for measuring similarity have been proposed so far, most of them fail to be robust to parameters: distinct parameter selections may bring in huge difference for the result. Then, based on the similarity scores, one is required to build the network by determine where to allocate the edges. Basically, thresholding and $k$NN are two most popular approaches for as-
assigning edges. Thresholding is defined by some specific value such that edge should be only assigned between two molecules which receive similarity score higher than thresholding value. Hence, different thresholding values correspond to totally different networks. Similarly, the selection of parameter $k$ for $k$NN method also controls the resulting network. Therefore, parameter selections may influence on network significantly. In addition, sensitivity to noise in biological data can be one obvious drawback of network. Recall the previous process of constructing network, it is easy to see that the effect of noise may break edges between molecules which should have been connected while miss-connecting some others which should be irrelevant.

Therefore, for module structure learning, the crucial point is to seek for a approach that is robust to the parameter selection and corruption occurred in networks. One intuitive idea to reduce the effect of both parameter selections and noise is integrating multiple networks composed of the same set of nodes, see [41] [31] [32] [43] [50] [30].

Identifying module structure from multiple networks simultaneously is also the main motivation of our work. The main idea is to formulate multiple networks into order-three tensor: a cubic-like three dimensional data array with each layer (which is actually a matrix) indicating one of the multiple networks. Base on the formulation of order three tensor data, two of the three dimensions indicate the molecules considered which exactly follow the multiple networks, while the last dimension refers to distinct networks enrolled. Then, our goal is to identify the common module structure embedded in multiple networks.

The proposed method is mainly inspired by Markov-MIML method in [78]. They formulate a Markov chain by normalizing the adjacency matrix among objects and iteratively learn a label indicator of objects for multi-instance multi-label learning (MIML) tasks. Remark each column of the label indicator is guaranteed to be a probability distribution vector. As far as we concern, module identification problem can be also formulated in the classification point of view: by considering each node (molecule) in network as an object to be classified, our goal is to separate all the nodes into two categories based on the given prior information: in or out of the module(s) containing prior node(s). Therefore, we determine to formulate this module identification problem on tensor data into two high order Markov chains. The main
difficulty here is that instead of formulating the problem under the matrix framework, we target at algorithm that can handle tensor data. To overcome this difficulty, a novel two-stage iterative algorithm is proposed: in the first stage, we multiply two probability transition tensors by the initial guess of label indicator vector such that two probability transition matrices are generated with one dimension of corresponding to nodes and the other referring to multiple networks. Then in the second stage, by fixing these two matrices, we are able to formulate two Markov chains: one is to determine the transition probabilities from multiple networks to nodes incorporating with prior information and the other computes the transition probabilities from nodes to different networks. It is much easier to solve the solutions of these two Markov chains compared to the original high order Markov chains formulated by order tree tensor. Then by repeating the two stages, we are able to compute both the label indicator as well as contribution score of each network iteratively. The label indicator provides module structure which includes prior node(s) while the contribution scores not only evaluate the contribution of different networks to the detected module structure, but also give a criteria of measuring consistency of the modules across multiple networks. More importantly, we may also tell if one network is corrupted by noise or poor parameter selections based on the contribution scores of multiple networks. We show the effectiveness as well as efficiency of the proposed tensor-based module identification method by the experimental results on three different gene data sets. Moreover, results from biological analysis also validate the module identified by the proposed algorithm.

5.1.1 Multidimensional Markov chains

Suppose we have a number of $n_2$ co-expression networks constructed by the same set of $n_1$ nodes which can be genes, proteins and some other small molecules. The $k$-th network $A_k$ ($k = 1, 2, \cdots, n_2$) is expressed by $n_1$-by-$n_1$. Our idea of integrating the $n_2$ networks together is combining the multiple networks into order three tensor data. More precisely, let $A \in \mathbb{R}^{n_1 \times n_1 \times n_2}$ be the order three tensor data, each layer of $A$ shall be formed by the $k$-th network $A_k$. Therefore $[A]_{i,j,k}$ which indicates $(i,j)$-th element at $k$-th layer of tensor should be $[A_k]_{i,j}$ which is the $(i,j)$-th element of $k$-th
Our idea is formulating Markov chains to extract consistent module structure embedded in the multiple networks, thus we may first normalize original order three tensor data $A$. In fact, in our case it is possible to normalize $A$ along different dimensions and then we have $A^{(1)}$ and $A^{(2)}$ as transition probability tensors:

$$
\sum_{i=1}^{n_1} [A^{(1)}]_{i,j,k} = 1, \sum_{k=1}^{n_2} [A^{(2)}]_{i,j,k} = 1,
$$

Note that $A^{(1)}$ is normalized along the dimension corresponding to nodes (molecules) while $A^{(2)}$ is normalized along the multiple networks. To formulate multidimensional Markov chains, we should first define $p$-mode multiplication for tensors. Suppose $B \in \mathbb{R}^{n_1 \times n_2 \times \cdots \times n_m}$ is an order $m$ tensor while $b \in \mathbb{R}^{n_p}$ is a column vector with $i_p = 1, 2, \cdots, m$. Then $p$-mode multiplication can be defined as follows:

$$
D = B \times_p b
$$

where $D \in \mathbb{R}^{n_1 \times \cdots \times n_{p-1} \times n_{p+1} \times \cdots \times n_m}$ is order $m-1$ tensor. Elements of $D$ should be calculated by:

$$
[D]_{j_1, \cdots, i_{p-1}, i_{p+1}, \cdots, i_m} = \sum_{j=1}^{n_p} [B]_{j_1, \cdots, i_{p-1}, j, i_{p+1}, \cdots, i_m} [b]_j.
$$

Then incorporating with prior information, we can one step further formulate the problem by multidimensional Markov Chains:

$$
x = (1 - \alpha) A^{(1)} \times_2 x \times_3 y + \alpha p; \quad (5.1)
$$

$$
y = A^{(2)} \times_1 x \times_2 x. \quad (5.2)
$$

Here, our aim is to seek for $x$ and $y$ satisfying (5.1) and (5.2) where $p$ indicates the label information while in $x$ nodes which are closely related to prior $p$ receives high values, such that identify the nodes that belong to the same module with the prior nodes. $y$ offers evaluations of contributions from different networks to the module. $\alpha \in [0, 1]$ here is set to balance the contribution between multiple networks and prior information.
We remark that proposed tensor based model is unsupervised, such that label information of the prior nodes is not required. After reach the steady state of multidimensional Markov chains, if any nodes are connected with prior while at the same time, closely related to each other, they should receive much high values in \( x \) compared to the others. These nodes together should construct densely connected subgraphs which is module structure we required to seek for.

Moreover, any nodes in networks can be selected as prior, for instance, if a set of nodes \( S = \{s_1, s_2, s_3, \cdots, s_l\} \subset 1, 2, \cdots n_1 \) are chosen to be prior in the Markov chains, then set \([p]_S = 1/l\) and all the other entries to be zero. The resulting \( x \) will identify the modules containing nodes in set \( S \). One step further, if all the module structures are required to be detected, we may go through all the nodes in the multiple networks to be prior information. More precisely, denote \( I_{n_1} \in \mathbb{R}^{n_1 \times n_2} \) as the identical matrix, we are able to extract all the modules embedded in the networks by setting \( p \) to be each column of \( I_{n_1} \) one by one.

In addition, \( y \) provides us an opportunity to evaluate the consistency of identified module. The more balanced values we get in \( y \), the more consistent module structure across \( k \) networks we detect. Furthermore, if contribution scores of most networks are quite balanced while big differences are observed for very few of them, the conclusion can be drawn that these small number of networks with unbalanced contribution scores should be corrupted seriously by noise or bad parameter selections.

### 5.1.2 Numerical algorithm

In order to reach the global steady state of system in (5.1) and (5.2), we propose a two-stage iterative scheme. More precisely, we first fix \( M_1 = \mathcal{A}^{(1)} \times_2 x \) and \( M_2 = (\mathcal{A}^{(2)} \times_2 x)^T \). Then we get the following linear system:

\[
\begin{align*}
x &= (1 - \alpha)M_1 y + \alpha p; \quad (5.3) \\
y &= M_2 x. \quad (5.4)
\end{align*}
\]

\( M_1 \) and \( M_2 \) are two transition probability matrices, thus (5.3) and (5.4) give two standard Markov chains which can be solved iteratively according to following
recursive formulas:

\[ x_{k+1} = (1 - \alpha)M_1 y_k + \alpha p; \quad (5.5) \]
\[ y_{k+1} = M_2 x_k. \quad (5.6) \]

Once (5.5), (5.6) converge with \( \hat{x} \) and \( \hat{y} \), we may update the probability transition matrices by:

\[ \hat{M}_1 = A^{(1)} \times x; \quad \hat{M}_2 = (A^{(2)} \times x)^T. \]

Replacing \( M_1, M_2 \) in (5.5) and (5.6) by \( \hat{M}_1, \hat{M}_2 \), we are capable to once again update \( x, y \) accordingly. Then by repeating the above process until steady state, the nonlinear systems in (5.1), (5.2) can be solved eventually. Note the initial guess of \( x, y \) and \( p \) in our algorithm is required to be probability distribution vectors.

Now, we are ready to present the proposed algorithm (see Algorithm 4) for solving \( x \) and \( y \) in (5.1)(5.2).

\begin{algorithm}
\textbf{Algorithm 4}
\begin{itemize}
    \item \textbf{Input:} \( A^{(1)}, A^{(2)}, x(0), y(0), p, \alpha \) and tolerance \( \varepsilon \)
    \item \textbf{Output:} \( x^* \) and \( y^* \)
    \item \textbf{Procedure}
    \begin{enumerate}
        \item Set \( t = 1 \);
        \item Compute \( M_1(t-1) = A^{(1)} \times x(t-1); M_2(t-1) = (A^{(2)} \times x(t-1))^T \);
        \item Set \( k = 0, x_0 = x(t-1), y_0 = y(t-1) \);
        \item Compute \( x_k = (1 - \alpha)M_1(t-1)y_k + \alpha p, \) and \( y_k = M_1(t-1)x_k \);
        \item If \( \|x_k - x_{k-1}\| < \varepsilon \) and \( \|y_k - y_{k-1}\| < \varepsilon \), set \( x(t) = x_k, y(t) = y_k \);
        \item Otherwise set \( k = k + 1 \) and goto Step 4.
        \item If \( \|x(t) - x(t-1)\| < \varepsilon \) and \( \|y(t) - y(t-1)\| < \varepsilon \), set \( x^* = x(t), y^* = y(t) \);
        \item Otherwise set \( t = t + 1 \) and goto Step 2.
    \end{enumerate}
\end{itemize}
\end{algorithm}
5.1.3 Convergence and Complexity Analysis

In order to illustrating the stability as well as efficiency of the tensor-based Markov chain module identification (TMI) method, we provide convergence and complexity analysis of the proposed algorithm. It can be shown that the convergence of Algorithm 1 can be guaranteed when $\alpha > \frac{2}{3}$. In addition, if $n_1$ which indicates the total number of nodes (molecules) enrolled in networks is a large number, we may expect Algorithm 1 to be convergent for $\alpha \geq \frac{1}{2}$.

5.1.3.1 Convergence Analysis of the Proposed Algorithm

Let us first consider the following lemma:

Lemma 4.

$$\sum_{i=1}^{n_1} [M_1]_{i,k} = 1, \quad \sum_{i=1}^{n_2} [M_2]_{i,k} = 1$$

Proof.

$$\sum_{i=1}^{n_1} [M_1]_{i,k} = [A^{(l)} \times x]_{ik} = \sum_{j=1}^{n_1} a_{i,j,k} x_j$$

$$\sum_{i=1}^{n_1} [M_1]_{i,k} = \sum_{i=1}^{n_1} \sum_{j=1}^{n_1} a_{i,j,k} x_j = \sum_{i=1}^{n_1} \left( \sum_{j=1}^{n_1} a_{i,j,k} x_j \right) = \sum_{j=1}^{n_1} x_j = 1$$

we also easy get $\sum_{i=1}^{n_2} [M_2]_{i,k} = 1$ via the similar way.

This Lemma 4 guarantees that both $M_1$ and $M_2$ are transition probability matrices. Then we may show that both $x$ and $y$ satisfying (5.3) and (5.4) are unique:

Lemma 5. For $\alpha \in (0, 1)$, there exist a unique $x$ and $y$ that satisfy (5.3) and (5.4).

Proof. From lemma 4, we get that

$$\|M_1\|_1 = \|M_2\|_1$$

therefore

$$\|M_1 M_2\|_1 \leq \|M_1\|_1 \|M_2\|_1$$
the radius of $M_1M_2$ is not larger than its norm, we get $\rho(M_1M_2) \leq 1$. From (5.3) and (5.4),
\[
x = (1 - \alpha)M_1M_2x + \alpha p \\
y = (1 - \alpha)M_2M_1y + \alpha p
\]
then
\[
(I - (1 - \alpha)M_1M_2)x = \alpha p \\
(I - (1 - \alpha)M_2M_1)y = \alpha p
\]
where $I$ is identity matrix. because
\[
\rho((1 - \alpha)M_1M_2) = (1 - \alpha)\rho(M_1M_2) \leq (1 - \alpha) < 1 \quad \text{and} \quad \rho((1 - \alpha)M_2M_1) \leq 1 - \alpha < 1.
\]
Therefore $I - (1 - \alpha)M_1M_2$ and $I - (1 - \alpha)M_2M_1$ nonsingular,
\[
x = \alpha[I - (1 - \alpha)M_1M_2]^{-1}p \\
y = \alpha[I - (1 - \alpha)M_2M_1]^{-1}p
\]
\[\square\]

Now consider the the iteration in (5.5) and (5.6):
\[
x_{k+1} = (1 - \alpha)M_1y_k + \alpha p \\
y_{k+1} = M_2x_{k+1}
\]

**Lemma 6.** The iteration guarantees the sequence $\{x_k\}$, $\{y_k\}$ satisfy $\|x_k\|_1 = 1$ and $\|y_k\|_1 = 1$.

**Proof.** The iteration can be written as the following
\[
x_{k+1} = (1 - \alpha)M_1M_2x_k + \alpha p \\
y_{k+1} = (1 - \alpha)M_2M_1y_k + \alpha p
\]
let $e = [1, 1, \cdots, 1]^T$ then
\[
e^T x_{k+1} = (1 - \alpha)e^T M_1M_2x_k + e^T \alpha p \\
= (1 - \alpha)e^T M_2x_k + \alpha = (1 - \alpha)e^T x_k + \alpha
\]
If $\|x_k\|_1 = 1$, then $\|x_{k+1}\|_1 = 1$. The same consequence be get for sequence $\{y_k\}$. \[\square\]
Based on previous lemmas, we may give the following remarks.

Remarks:
1. If we set $\|x_0\|_1 = \|y_0\|_1 = 1$, then the $\| \cdot \|_1 = 1$ can keep during the iteration.
2. $\|M_1\|_1 = \|M_2\|_1 = 1$ is established under the condition $\|x\|_1 = 1$ which the iteration promised.

Then we are ready to present the following theorem:

**Theorem 7.** The iteration (5.5) and (5.6) is convergent.

Theorem 7 can be proved easily via the similar proof of lemma 2. This theorem demonstrates the convergence of the iterative scheme in the second stage of Algorithm 4. Then according to Algorithm 4, it is easy to get:

\[
x(t) = (1 - \alpha)M_1(t - 1)M_2(t - 1)x(t) + \alpha p
\]
\[
y(t) = (1 - \alpha)M_2(t - 1)M_1(t - 1)y(t) + \alpha p
\]

then

\[
x(t) = [I - (1 - \alpha)M_1(t - 1)M_2(t - 1)]^{-1}\alpha p
\]
\[
y(t) = [I - (1 - \alpha)M_2(t - 1)M_1(t - 1)]^{-1}\alpha p
\]

It is not difficult to see that the convergence of $\{y(t)\}$ is guaranteed if $\{x(t)\}$ converges. Therefore, it is enough that we only consider the convergence of $\{x(t)\}$ here. For simplicity, let

\[
x(t + 1) = F(x(t))
\]

where $F(x(t)) = (I - (1 - \alpha)M_1(t)M_2(t))^{-1}\alpha p$.

Let us consider the mapping

\[
F : x \to F(x)
\]

where $F(x) = (I - (1 - \alpha)M_1(x)M_2(x))^{-1}\alpha p$. $M_1(x) = A^{(1)} \times_2 x$ and $M_2(x) = (A^{(2)} \times_2 x)^T$.

Denote $Q(x) = (I - (1 - \alpha)M_1(x)M_2(x))^{-1}$, $G(x) = (Q(x))^{-1} = I - (1 - \alpha)M_1(x)M_2(x)$ and let $L(x) = M_1(x)M_2(x)$, we have:

\[
\frac{dF(x)}{dx} = \frac{d(\alpha Q(x)p)}{dx} = \alpha \frac{dQ(x)}{dx} - p
\]
\[
\frac{dQ(x)}{dx} = -Q(x)\frac{dG(x)}{dx}Q(x)
\]

\[
\frac{dG(x)}{dx} = \frac{d(I - (1 - \alpha)M_1M_2)}{dx} = -(1 - \alpha)\frac{dL(x)}{dx}
\]

\[
\frac{dL(x)}{dx} = \frac{dM_1(x)M_2(x)}{dx} = \frac{dM_1(x)}{dx}M_2(x) + M_1(x)\frac{dM_2(x)}{dx}
\]

In addition, if rewrite

\[
M_1(x) = \mathcal{A}^{(1)} \times_2 x
\]

\[
= \begin{pmatrix}
\sum_{j=1}^{n_1} a_{1,j,1}^{(1)} x_j & \cdots & \sum_{j=1}^{n_1} a_{1,j,n_2}^{(1)} x_j \\
\vdots & \ddots & \vdots \\
\sum_{j=1}^{n_1} a_{n_1,j,1}^{(1)} x_j & \cdots & \sum_{j=1}^{n_1} a_{n_1,j,n_2}^{(1)} x_j
\end{pmatrix} (x \otimes I_{n_2})
\]

\[
= \mathcal{A}^{(1)}(1)(x \otimes I)
\]

and

\[
M_2(x) = (\mathcal{A}^{(2)} \times_2 x)^T
\]

\[
= \begin{pmatrix}
a_{1,1,1}^{(2)} & \cdots & a_{1,n_1,1}^{(2)} & \cdots & a_{n_1,1,1}^{(2)} \\
\vdots & \ddots & \vdots & \ddots & \vdots \\
a_{1,1,n_2}^{(2)} & \cdots & a_{1,n_1,n_2}^{(2)} & \cdots & a_{n_1,1,n_2}^{(2)}
\end{pmatrix} (x \otimes I_{n_1})
\]

\[
= \mathcal{A}^{(2)}(3)(x \otimes I)
\]

where \(\mathcal{A}^{(1)}(1)\) is unfolding of tensor \(\mathcal{A}^{(1)}\) from 1st direction, \(\mathcal{A}^{(2)}(3)\) is the unfolding of tensor \(\mathcal{A}^{(2)}\) from 3rd direction, we may get

\[
\frac{dM_1(x)}{dx} = \mathcal{A}^{(1)}(1)I
\]

\[
\frac{dM_2(x)}{dx} = \mathcal{A}^{(2)}(3)I
\]

Then

\[
\frac{\|dL(x)\|_1}{dx} \leq \frac{\|dM_1(x)\|}{x} \|M_2(x)\|_1 + \|M_1(x)\| \|\frac{dM_2(x)}{dx}\|_1
\]

\[
= \frac{\|dM_1(x)\|}{dx} \|1\| + \|\frac{dM_2(x)}{dx}\|_1 = 2
\]
In addition, we may also get

\[
\|Q(x)\|_1 = \|(I - (1 - \alpha)M_1(x)M_2(x))^{-1}\|_1 \\
\leq \frac{1}{1 - \|(1 - \alpha)M_1(x)M_2(x)\|_1} \\
= \frac{1}{1 - (1 - \alpha)}
\]

Therefore, we have

\[
\left\| \frac{dF(x)}{dx} \right\|_1 = \|(1 - \alpha)Q(x)\frac{dL(x)}{dx} Q(x)p\|_1 \\
= (1 - \alpha)\|Q(x)\frac{dL(x)}{dx} Q(x)p\|_1 \\
\leq 2(1 - \alpha)\alpha \frac{1}{[1 - (1 - \alpha)]^2} = \frac{2 - 2\alpha}{\alpha}
\]

In conclusion, when \(\alpha > \frac{2}{3}\), \(\|\frac{dF(x)}{dx}\|_1 < 1\), it is easy to see that the mapping \(F\) is a contract mapping. From the Banach fixed point theorem, the equation (5.7) converge to a unique fixed point \(x^*\) which means the algorithm 1 converges. Then we may give the following thereom:

**Theorem 8.** If \(\alpha > \frac{2}{3}\), the algorithm converges.

In general, convergence of the proposed algorithm can be established when \(\alpha > 2/3\) with initial guess of \(x(0)\) and \(y(0)\) are probability distribution vectors. Empirically, when \(\alpha \geq 1/2\), we may also observe the convergence of the proposed algorithm especially for large number of data points are considered. In addition, the converged \(x^*\) and \(y^*\) should be unique probability distribution vectors.

### 5.1.4 Complexity of the proposed algorithm

We provide the computational complexity of the proposed algorithm listed in Algorithm 4. In the first stage of each iteration, we are required to compute \(M_1\) and \(M_2\) with complexity of \(O(n_1^2n_2)\). Then, during the second stage, it takes \(O(n_1n_2)\) to update \(x\) and \(y\) at each step. Suppose that for fixed \(M_1\) and \(M_2\), it requires \(k = Iter_1\) steps for both \(x\) and \(y\) to converge and we further assume that when \(t\) reaches the number of \(Iter_2\), the whole algorithm converges. The total computational cost of Algorithm 4 should be \(O(Iter_2 \ast (n_1^2n_2 + Iter_1 \ast (n_1n_2)))\).
5.2 Experimental Results

We mainly focus on the performance on identifying common module of the proposed algorithm in this section. Both synthetic data and real data are considered in order to show the efficiency as well as the effectiveness of the proposed tensor based method. Although there are methods proposed to address the common module identification problem [71,84], they are developed under the assumption that the underlying modules (clusters) are the same across different networks (data sets). In our considered networks, this assumption does not hold. We only report the experimental results of our proposed tensor based Markov Chain algorithm.

5.2.1 Module identification on synthetic data

In this subsection, we illustrate the efficiency as well as the effectiveness of the proposed algorithm by conducting experiments on synthetic data. As we have mentioned before, the proposed tensor based algorithm is capable of handling any number of multiple networks simultaneously. In this case, we implement proposed method 5 different synthetic networks corresponding to a same set of 100 vertices. In each network, the subgraph formed by 10 vertices is more densely connected compared with the others. Thus, when considering the multiple networks together, this subgraph should be exactly the common module structure which proposed algorithm targets on. Practically, we set the subgraph constructed by the first 10 vertices to be complete graph for all the networks except the third one. In the third networks, we try to add noise to the networks by removing some of the edges in the subgraph of the first 10 vertices (see Figure 5.1) such that we can test if the proposed algorithm is capable of addressing the noised networks successfully. For the rest vertices the networks, we randomly put edges to connect vertices and finally, the average degree of each vertex in networks achieves around 18.

In the experiment, we set $\alpha = 0.5$ and employ the first vertex as prior information. The running time of the algorithm is around 0.02 second. Figure 5.2 gives the value of $x$ generated by performing the proposed algorithm on the synthetic networks. Obviously, the values received by first 10 vertices in $x$ are larger than the others.
Table 5.1: The 11 largest values in $x$ respect to the vertices for synthetic networks

<table>
<thead>
<tr>
<th>Vertex</th>
<th>Value of $x$</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>0.5085</td>
</tr>
<tr>
<td>10</td>
<td>0.0374</td>
</tr>
<tr>
<td>4</td>
<td>0.0354</td>
</tr>
<tr>
<td>2</td>
<td>0.0318</td>
</tr>
<tr>
<td>8</td>
<td>0.0315</td>
</tr>
<tr>
<td>7</td>
<td>0.0306</td>
</tr>
<tr>
<td>5</td>
<td>0.0304</td>
</tr>
<tr>
<td>3</td>
<td>0.0295</td>
</tr>
<tr>
<td>6</td>
<td>0.0276</td>
</tr>
<tr>
<td>9</td>
<td>0.0274</td>
</tr>
<tr>
<td>73</td>
<td>0.0078</td>
</tr>
</tbody>
</table>

Table 5.2: Contribution scores in $y$ for synthetic data.

<table>
<thead>
<tr>
<th>Networks</th>
<th>1-st</th>
<th>2-nd</th>
<th>3-rd</th>
<th>4-th</th>
<th>5-th</th>
</tr>
</thead>
<tbody>
<tr>
<td>Value of $y$</td>
<td>0.2192</td>
<td>0.2225</td>
<td>0.1046</td>
<td>0.2231</td>
<td>0.2306</td>
</tr>
</tbody>
</table>

More specifically, in Table 5.1 we list the largest 11 entries of resulting $x$. We may also check the contribution of each network by checking value of each entry in $y$, see Table 5.2. Recover that the third network is corrupted by noise. It is easy to find out that the contribution score of the third network comes out to be the smallest while the contribution of the others is almost on the same level. Base on the these observations, we may draw the conclusion that proposed algorithm not only identifies common module embedding in multiple networks correctly, but also addresses the noised networks successfully.

5.2.2 Module identification on three cancer gene co-expression networks

In this subsection, we report the experimental results conducted on three gene data sets downloaded from The Cancer Genome Atlas (TCGA). The three cancers are ovar-
ian cancer (OV), glioblastoma multiforme (GBM), and lung squamous cell carcinoma (LUSC), respectively. All the data are generated with Affymetrix HT HG-U133A by Broad Institute. There are 558 OV samples, 594 GBM samples and 134 LUSC samples in total 22,277 different genes. For each cancer, we compute the variance of all the genes across the samples. In the experiments, we only select 1500 genes with largest variance for each cancer. Combining all 3 cancers, we actually select the number of 2756 different genes for further study.
The next step, for each of the three cancers, we calculate the Pearson correlation coefficients across all the genes and construct the affinity matrices by taking the hard thresholding. More precisely, if the Pearson correlation coefficient of two genes is greater than some pre-defined value, we assign an edge to link them up; otherwise, there should be no edge between them. Practically, we set thresholding 0.65, 0.60, 0.52 for OV, LUSC and GBM respectively such that all these networks have approximately scale free property. Moreover, the average degree of each gene across all 3 networks is about 18. Then, by removing the genes that have no connection to any other gene in all the 3 cancer networks, we finally construct three networks of number of 2297 common genes.

We apply our proposed TMI method to the networks to identify the module structures. With different nodes being selected as prior information, we are able to generate a probability distribution vector in which the nodes similar to them receive clearly much higher value than the others. Note that the parameter $\alpha$ is tuned based on seeking for an obvious jump between them. Moreover, another probability distribution vector $y$ is also generated to measure the contribution of different networks. Based on $y$, we may also check the consistency of the modules across multiple networks. When the differences in contribution scores of multiple networks are relatively small, we may conclude that module structure is consistent. Otherwise, the module structure may only appear in some of the networks. Thus, we may identify the consistent modules based on both converged $x$ and $y$. In order to illustrate the effectiveness of the proposed method more clearly, we state and discuss several examples in the following part.

5.2.2.1 Example 1

We select the control probe AFFX-r2-Ec-bioB-M as prior information $p$ in this example. When we set $\alpha$ to be 0.5, our algorithm converges within about 19.8 seconds and the resulting $x$ can be seen in the Figure 5.3. The gene receiving largest value is AFFX-r2-Ec-bioB-M itself. In addition, it is clear to find out in the figure that some genes receive larger values compared with the others. To make it easier to check, we sort $x$ in descending order and list the largest 8 values in Table 5.3 with
Figure 5.3: Value of $x$ with gene AFFX-r2-Ec-bioB-M at as prior information for example 1 in the three gene co-expression networks of cancers.

<table>
<thead>
<tr>
<th>Gene IDs</th>
<th>Value of $x$</th>
</tr>
</thead>
<tbody>
<tr>
<td>AFFX-r2-Ec-bioB-M-at</td>
<td>0.5386</td>
</tr>
<tr>
<td>AFFX-BioB-5-at</td>
<td>0.0769</td>
</tr>
<tr>
<td>AFFX-BioB-M-at</td>
<td>0.0769</td>
</tr>
<tr>
<td>AFFX-BioC-3-at</td>
<td>0.0769</td>
</tr>
<tr>
<td>AFFX-BioC-5-at</td>
<td>0.0769</td>
</tr>
<tr>
<td>AFFX-BioDn-5-at</td>
<td>0.0769</td>
</tr>
<tr>
<td>AFFX-r2-Ec-bioB-M-at</td>
<td>0.0769</td>
</tr>
<tr>
<td>201348-at</td>
<td>$7.2426 \times 10^{-18}$</td>
</tr>
</tbody>
</table>

Table 5.3: The 8 largest values in $x$ with their gene IDs for example 1 in the three gene co-expression networks of cancers.

the gene IDs accordingly. Obviously, a jump can be found between the value of gene AFFX-BioDn-5-at (0.0769) and gene 201348-at ($7.2426 \times 10^{-18}$) and the first 7 genes receives much higher value than the others.

Moreover, we also concern about the converged $y$ which is the contribution scores of multiple cancers as listed in Table 5.4. We may see for this example, the contribution of each cancer is on the same level. We get back to check the networks and find that the subgraphs constructed by the 7 identified genes are all complete graphs (thus exactly the same with each other). Hence it is reasonable that the contribution scores of the 3 networks are the same. In addition, the complete subgraphs extracted
Table 5.4: Contribution scores in $y$ of three cancers for example 1.

<table>
<thead>
<tr>
<th>Cancers</th>
<th>CBM</th>
<th>LUSC</th>
<th>OV</th>
</tr>
</thead>
<tbody>
<tr>
<td>Value of $y$</td>
<td>0.3333</td>
<td>0.3333</td>
<td>0.3333</td>
</tr>
</tbody>
</table>

Figure 5.4: Value of $x$ with gene 202708-s-at as prior information for example 2 in the three gene co-expression networks of cancers.

also explain the reason why all the other nodes identified receive the same score in $x$.

Based on the results in Table 5.4 and Table 5.4, we may conclude that AFFX-r2-Ec-bioB-M-at, AFFX-BioB-5-at, AFFX-BioB-M-at, AFFX-BioC-3-at, AFFX-BioC-5-at, AFFX-BioDn-5-at and AFFX-r2-Ec-bioB-M-at should form a consistent module across networks of three cancers. Biologically, these nodes selected in this module are all known to be control probes which should be closely related with each other. This evidence supports that the module identified is meaningful, which validates the effectiveness of the proposed algorithm.

5.2.2.2 Example 2

In this example, we employ the gene 202708-s-at as prior information. $\alpha$ here is again set to be 0.5 and the computational time is around 20.0 seconds. In Figure 5.4, we plot the converged $x$ to make it clear to see.

Obviously, several genes which should be closely related with gene 202708-s-at receive much higher values compared with the others. Similarly, we sort converged $x$ in descending order and list the largest 15 values with their gene IDs respectively in Table 5.5. We may see a gap appears between value 0.0312 of gene 209911-x-at and
Table 5.5: The 15 largest values in $x$ with their gene IDs for example 2 in the three gene co-expression networks of cancers.

<table>
<thead>
<tr>
<th>Gene IDS</th>
<th>Value of $x$</th>
</tr>
</thead>
<tbody>
<tr>
<td>202708-s-at</td>
<td>0.5364</td>
</tr>
<tr>
<td>210387-at-at</td>
<td>0.0511</td>
</tr>
<tr>
<td>209398-at</td>
<td>0.0457</td>
</tr>
<tr>
<td>214290-s-at</td>
<td>0.0457</td>
</tr>
<tr>
<td>215071-s-at</td>
<td>0.0457</td>
</tr>
<tr>
<td>214455-at</td>
<td>0.0456</td>
</tr>
<tr>
<td>218280-x-at</td>
<td>0.0446</td>
</tr>
<tr>
<td>206110-at</td>
<td>0.0446</td>
</tr>
<tr>
<td>208579-x-at</td>
<td>0.0368</td>
</tr>
<tr>
<td>208180-s-at</td>
<td>0.0367</td>
</tr>
<tr>
<td>214469-at</td>
<td>0.0360</td>
</tr>
<tr>
<td>209911-x-at</td>
<td>0.0312</td>
</tr>
<tr>
<td>206640-x-at</td>
<td>$1.3683 \times 10^{-17}$</td>
</tr>
<tr>
<td>207086-x-at</td>
<td>$1.3683 \times 10^{-17}$</td>
</tr>
<tr>
<td>207663-x-at</td>
<td>$1.3683 \times 10^{-17}$</td>
</tr>
</tbody>
</table>

Table 5.6: Contribution scores in $y$ of three cancers for example 2.

<table>
<thead>
<tr>
<th>Cancers</th>
<th>CBM</th>
<th>LUSC</th>
<th>OV</th>
</tr>
</thead>
<tbody>
<tr>
<td>Value of $y$</td>
<td>0.6008</td>
<td>0.3672</td>
<td>0.0320</td>
</tr>
</tbody>
</table>

Consider the converged results of $y$ in Table 5.6. We may see different from the previous example, the contribution scores of three cancers are different.

In order to figure out the reason, we check the subgraphs formulated by identified nodes in all the three networks see Figure 5.5.

We may see that vertices are closely related with each other in GBM and LUSC cancer networks while the subgraph in OV cancer does not contain as many edges as which in GBM and LUSC networks. It turns out that the proposed algorithm assigns 0.6008 as contribution score to the GBM cancer network and 0.3672 to LUSC.
Figure 5.5: Subgraphs constructed by 12 genes selected for example 2 in the three gene co-expression networks of cancers, left: subgraph in GBM cancer network; middle: subgraph in LUCS cancer network; right: subgraph in OV cancer network.

cancer network while OV cancer network only receives 0.0320. We do the enrichment analysis for Gene Ontology (GO, biological process) and KEGG pathways for this identified module. All the genes in this module belong to histone cluster 1 or histone cluster 2. 10 of the twelve genes in the module enrich 12 GO terms and they cover all the genes belonging to these GO terms among all the genes we consider. In Table 5.6, the related functions of these 12 GO terms can be found with P-value less than $10^{-11}$. In addition, 8 in the 12 genes in this module enrich the pathway: hsa05322: Systemic lupus erythematosus. According to [3, 57], this pathway is related to several cancers such as liver cancers, lung cancers and kidney cancers. It is interesting that genes in
### Table 5.7: Gene ontology enrichment of the module identified for example 2 in the three gene co-expression networks of cancers.

<table>
<thead>
<tr>
<th>Enriched GO terms</th>
<th>%</th>
<th>P-value</th>
</tr>
</thead>
<tbody>
<tr>
<td>GO:006334 nucleosome assembly</td>
<td>100</td>
<td>8.83×10⁻²¹</td>
</tr>
<tr>
<td>GO:0031497 chromatin assembly</td>
<td>100</td>
<td>1.21×10⁻²⁰</td>
</tr>
<tr>
<td>GO:0065004 protein-DNA complex assembly</td>
<td>100</td>
<td>1.88×10⁻²⁰</td>
</tr>
<tr>
<td>GO:0034728 nucleosome organization</td>
<td>100</td>
<td>2.31×10⁻²⁰</td>
</tr>
<tr>
<td>GO:0006323 DNA packaging</td>
<td>100</td>
<td>1.98×10⁻¹⁹</td>
</tr>
<tr>
<td>GO:0006333 chromatin assembly or disassembly</td>
<td>100</td>
<td>4.25×10⁻¹⁹</td>
</tr>
<tr>
<td>GO:0034622 cellular macromolecular complex assembly</td>
<td>100</td>
<td>1.96×10⁻¹⁵</td>
</tr>
<tr>
<td>GO:0034621 cellular macromolecular complex subunit organization</td>
<td>100</td>
<td>5.62×10⁻¹⁵</td>
</tr>
<tr>
<td>GO:0006325 chromatin organization</td>
<td>100</td>
<td>9.46×10⁻¹⁵</td>
</tr>
<tr>
<td>GO:0051276 chromosome organization</td>
<td>100</td>
<td>9.11×10⁻¹⁴</td>
</tr>
<tr>
<td>GO:0065003 macromolecular complex assembly</td>
<td>100</td>
<td>1.59×10⁻¹²</td>
</tr>
<tr>
<td>GO:0043933 macromolecular complex subunit organization</td>
<td>100</td>
<td>2.88×10⁻¹²</td>
</tr>
</tbody>
</table>

This module identified in considered cancers networks also enrich this pathway. Based on the previous discussions, we may draw the conclusion that the module discovered by the proposed algorithm is meaningful in biological study. Therefore, we may also claim that the edges are preserved well in GBM and LUSC cancer networks while corrupted in the OV cancer network.

#### 5.2.2.3 Example 3

In this example, we consider the case that gene 200606-at is employed as prior. Similar to the previous examples, we set $\alpha = 0.5$ and run the algorithm on multiple networks of three cancers. The time cost of the algorithm is around 20.4 seconds and the converged $x$ can be found in Figure 5.6. We explore that there are in total 8 genes receiving much larger values than the others. When considering $y$, which indicates the contribution of each network in Table 5.8, we may find the contribution scores of both GBM and LUSC cancer are very small. Then we check the networks of all three cancers and discover that the subgraphs in both GMB and LUSC contain no edge, see Figure 5.7. Thus the module structure is not consistent in both GMB and LUSC.
Figure 5.6: Value of $x$ with gene 200606-at as prior information for example 3 in the three gene co-expression networks of cancers.

<table>
<thead>
<tr>
<th>Cancers</th>
<th>CBM</th>
<th>LUSC</th>
<th>OV</th>
</tr>
</thead>
<tbody>
<tr>
<td>Value of $y$</td>
<td>0.0009</td>
<td>0.0030</td>
<td>0.9961</td>
</tr>
</tbody>
</table>

Table 5.8: Contribution scores in $y$ of three cancers for example 3.

cancers. Then edges appearing in subgraph of OV cancer networks are considered to be miss-linked due to the noise or parameter effect.

5.2.3 Module identification from co-expression network of different tissues of morbidly obese patients

In this part we present conducted experiments and results on co-expression network constructed by gene expression profile of liver (LIV), ometal (OME) and subcutaneous (SUBC) tissues for morbidly obese patients (GEO Accession number: GSE24294). There are in total 459 subjects with data across all three tissues and all the data are measured on the number of 40638 probes. We select the genes covered by more than one probe and employ the mean value as expression of that gene. Moreover, the genes with greater than 10% missing observations are excluded, and we use mean of available data to express other missing values. Among the rest 17282 common genes, we select 1800 with largest variance across samples for each tissue and thus, we construct three gene networks which evolve the number of 2637 genes. Then we perform hard thresholding at 0.5 on multiple networks of all three tissues. Similar to
Figure 5.7: Subgraphs constructed by 12 genes selected for example 3 in the three gene co-expression networks of cancers, left: subgraph in GBM cancer network; middle: subgraph in LUCS cancer network; right: subgraph in OV cancer network.

In the previous experiments, we remove the nodes which do not connect with any other nodes in the network and finally, each network consists of 1873 common genes.

By formulating the multiple networks into tensor data, we are able to employ the proposed method to handle the module identification problem. For the same purpose as previous experiments, we select a satisfied $\alpha$ which brings clear gap between values received by genes in $x$. To validate the proposed method, we present some examples to clarify the effect and efficiency of the proposed tensor based method.
Figure 5.8: Value of $x$ with gene SAA1 as prior information for example 1 in the three gene co-expression networks of different tissues of morbidly obese patients.

Table 5.9: The 5 largest values in $x$ with their gene IDs for example 1 in the three gene co-expression networks of different tissues of morbidly obese patients.

<table>
<thead>
<tr>
<th>Gene ID</th>
<th>Value of $x$</th>
</tr>
</thead>
<tbody>
<tr>
<td>SAA1</td>
<td>0.5706</td>
</tr>
<tr>
<td>SAA4P</td>
<td>0.1162</td>
</tr>
<tr>
<td>SAA3</td>
<td>0.1149</td>
</tr>
<tr>
<td>SAA2</td>
<td>0.1035</td>
</tr>
<tr>
<td>CPR</td>
<td>0.0192</td>
</tr>
</tbody>
</table>

5.2.3.1 Example 1

In this example, we select gene SAA1 as prior information to run the proposed algorithm. The parameter $\alpha$ is set to be 0.5 and it takes around 13.7 seconds for our method to converge. The resulting $x$ is plotted in Figure 5.8. Obviously, some of the genes receive much higher values than the others.

It can be seen more clearly in Table 5.9 that the first 4 genes listed in the table receive much larger scores (at least 0.1035 received by SAA2) than the others (at most only 0.0192 for CPR). Moreover, we also report evaluation on contribution of multiple networks in Table 5.10.

According to the converged $y$, the differences on contribution of multiple networks
Table 5.10: Contribution scores of three tissues in $y$ for example 1

<table>
<thead>
<tr>
<th>Tissues</th>
<th>LIV</th>
<th>OME</th>
<th>SUBC</th>
</tr>
</thead>
<tbody>
<tr>
<td>Value of $y$</td>
<td>0.4014</td>
<td>0.3091</td>
<td>0.2895</td>
</tr>
</tbody>
</table>

Figure 5.9: Value of $x$ with gene ALAS2 and HBA1 as prior information for example 2 in the three gene co-expression networks of different tissues of morbidly obese patients.

are not very significant which indicate that the module structure is quite consistent across three networks of different tissues. This module of 4 genes enriches 2 GO terms: GO:000695 acute-phase response and GO:0002526 acute inflammatory response with P-value $2.39 \times 10^{-8}$ and $3.69 \times 10^{-7}$ respectively. SAA is a well-known protein in inflammation-associated reactive amyloidosis (AA-type). These facts indicate this module identified by the proposed method is reasonable biologically.

5.2.3.2 Example 2

In this example, we would like to use multiple nodes as prior information to test the effect of the proposed algorithm. More specifically, we select ALAS2 and HBA1 as prior information in $p$, and perform the proposed tensor-based Markov chain algorithm with $\alpha = 0.5$. Computational time for the algorithm is 12.5 seconds and the converged $x$ can be found in Figure 5.9.

Moreover, it is listed in Table 5.11 that first 10 genes with largest value in $x$ receive at least 0.0228 when the value of others is at most $7.084 \times 10^{-3}$. We check
<table>
<thead>
<tr>
<th>Gene IDS</th>
<th>Value of $x$</th>
</tr>
</thead>
<tbody>
<tr>
<td>HBA1</td>
<td>0.2817</td>
</tr>
<tr>
<td>ALAS2</td>
<td>0.2723</td>
</tr>
<tr>
<td>TRIM58</td>
<td>0.0755</td>
</tr>
<tr>
<td>HBA2</td>
<td>0.0703</td>
</tr>
<tr>
<td>GPR144</td>
<td>0.0690</td>
</tr>
<tr>
<td>HBB</td>
<td>0.0592</td>
</tr>
<tr>
<td>HBG2</td>
<td>0.0454</td>
</tr>
<tr>
<td>CA1</td>
<td>0.0366</td>
</tr>
<tr>
<td>BHD</td>
<td>0.0361</td>
</tr>
<tr>
<td>HBG1</td>
<td>0.0228</td>
</tr>
<tr>
<td>HEMGN</td>
<td>$7.084 \times 10^{-3}$</td>
</tr>
<tr>
<td>RHAG</td>
<td>$7.084 \times 10^{-3}$</td>
</tr>
</tbody>
</table>

Table 5.11: The 12 largest values in $x$ with their gene IDs for example 2 in the three gene co-expression networks of different tissues of morbidly obese patients.

<table>
<thead>
<tr>
<th>Tissues</th>
<th>LIV</th>
<th>OME</th>
<th>SUBC</th>
</tr>
</thead>
<tbody>
<tr>
<td>Value of $y$</td>
<td>0.4120</td>
<td>0.3622</td>
<td>0.2258</td>
</tr>
</tbody>
</table>

Table 5.12: Contribution scores of three tissues in $y$ for example 2.

the contribution scores of networks in Table 5.12. The result indicates that the contribution scores received by LIV and OME networks are a little higher than SUBC. Still, it could tell that the module constructed by the chosen nodes is quite consistent across multiple networks.

This module enriches the GO:0015671 oxygen transport with P-value $3.58 \times 10^{-11}$ and GO:0015669 gas transport with P-value $1.53 \times 10^{-10}$ in the GO enrichment analysis. Among the considered genes in three tissues, half number of genes which carry out the function of oxygen transport and gas transport are located in this module.

In conclusion, our proposed TMI method is capable of identifying the module structure both efficiently and effectively. Moreover, it also allows the users to check the contributions of different networks in $y$. In the above experiments, both data sets contain three different networks while under our formulation, we can handle
any number of multiple networks. Remark that the noise level in different networks can be also evaluated by examining the values in $y$. If contribution scores of only a few networks are apparently distinct with the others, we may infer them as being noised severely or taking improper thresholding values. These results can be more convincible when more networks are involved in the tensor data.

5.3  Concluding Remarks on TMI Method

We propose a novel tensor based method for identification of module structures\footnote{For more details, please refer to [66].}. The main idea of the proposed algorithm is to construct two Markov chains and iteratively update both label indicator $x$ and contribution indicator $y$ until global steady state is reached. Then by considering converged $x$ and $y$ we may tell the consistent module embedded in multiple networks. In order to illustrate the efficiency and effectiveness of proposed algorithm, we conduct experiments on two gene data sets. The results support that proposed algorithm is capable of identifying modules in multiple networks.
Chapter 6

Random Under-sampling in Medical Imaging

Random under-sampling strategy is developed to tackle the application involving large scale data which brings serious computational and storage concern. Compared to under-sampling with regular pattern, the random under-sampling may preserve more information that benefits the data analysis. In particular, in CT imaging random under-sampling is capable of reducing the total radiation dose delivered to patient which has arisen increasingly clinical concern. The main contribution of this chapter is illustrating the effectiveness of random under-sampling in CT image reconstruction compared to the regular ones. In addition, we also propose a novel scatter removal algorithm that enhances the performance of random under-sampling strategy.

6.1 The Proposed Method

6.1.1 Least Square Problem

Mathematically, CT projection can be modeled simply by:

\[ \mathcal{P} f(\alpha, \beta) = g. \]

Here \( f(\alpha, \beta) \) indicates the demanded reconstruction results. \( \alpha \) and \( \beta \) are two angles used to parametrized the geometry of CT projection system. Note that \( \alpha \in [0, 2\pi] \) and
if we assume $d$ to be the radius of x-ray source, then $\beta \in [-\arcsin(1/d), \arcsin(1/d)]$. $\mathcal{P}$ is the projection operator which is usually called divergent beam transform. $g$ represents the log-transformed projection data measured on image detectors at various projection angles. According this model, CT reconstruction can be intuitively formulated into a quadratic optimization problem,

$$\arg\min_f \|\mathcal{P} f - g\|^2. \quad (6.1)$$

Let $\mathcal{P} = U \Sigma V^T$ denote the SVD of $\mathcal{P}$ where $\Sigma$ is diagonal and both $U$ and $V$ are orthogonal. Then the solution $f$ can be characterized directly as $f = V \Sigma^{-} U^T g$. $\Sigma^-$ is pseudo-inverse of $\Sigma$ which is also diagonal and $\sigma^{-}_{i,i} = 1/\sigma_{i,i}$ if $\sigma_{i,i} \neq 0$ and $\sigma^{-}_{i,i} = 0$ otherwise. This naturally implies a reconstruction strategy: based on the projection operator, decompose to basis in projection domain ($U$ in SVD), and superposition in image domain ($V$ in SVD), then reverse the projection process for construction of image data from projection data.

One particular problem of important importance is the complete data case. It has been well known that this operator $\mathcal{P}$ has a SVD form of $\{v_{m,l}, u_{m,l}; \sigma_{m,l}\}$ with $m \geq l$, $-m \leq l \leq m$ and $m + l$ should be even number. $m$ and $l$ introduce the block structure: $l$ denotes the block index which gives the sector information, while $m$ indicates index with each block. $v_{m,l}$, $u_{m,l}$ and $\sigma_{m,l}$ can be calculated by following formulas respectively:

$$v_{m,l}(x) = \pi^{-1/2}(m + 1)^{1/2}|x|^{l} P_{(m-l)/2}^{(0,l)}(2|x|^2 - 1)e^{il\arg x},$$

$$u_{m,l}(s, w) = \pi^{-1}w(s)U(s)e^{i\phi},$$

$$\sigma_{m,l} = 2\pi^{1/2}(m + 1)^{-1/2}.$$  

$w(s) = (1-s^2)^{1/2}, P_{n}^{(a,b)}$ here are Jacobi-polynomials and $U_m$ are Chebyshev-polynomials of second kind. More details please refer to [45, 46].

When it comes to under-sampling, let us consider a general under-sampled projection operator $P_s = S \mathcal{P}$, where $S = S(x, \alpha)$ is the under-sampling operator, with value to be 1 or 0 which imply corresponding projection data should be preserved or not. $x$ suggests the under-sampled position in each view, while $\alpha$ provides the angular information. For view under-sampling case, it is obvious that the under-sampling operator is only related to angles. Thus the under-sampling operator $S(x, \alpha)$ degraded
to $S(\alpha)$. On the other hand, when consider regular ray under-sampling, projection data is blocked according to specific pattern in each view which means under sampling operator $S(x)$ which has nothing related to $\alpha$ should be applied in this case. The random ray under-sampling case is more complicated compared to these two cases since projection data are randomly blocked instead of fixed pattern in each view. Thus projection operator should correspond to both angular variable $\alpha$ and position variable $x$.

For under-sampling case, it usually requires to reconstruct image under very ill-conditioned systems. Compared to $\mathcal{P}$, the stability of $P_s$ may be significantly lower especially when encountering large percent of down-sampling (which is the probability of $S(x, \alpha) = 0$ in $Q = \Omega \times B$ where $x \in \Omega$ and $\alpha \in B$). From the matrix point of view, $S$ brings information lost in row space of $P_s$ which may reduce the matrix rank compared to $\mathcal{P}$. More precisely, some singular values of $\mathcal{P}$ are removed to be 0 in $P_s$ such that information stored in corresponding basis in projection domain as well as superposition in image domain will be lost. In this case, under-sampling may cause serious quality decay in reconstructed image. Therefore, it is important to seek for specific down-sampling strategy in which information loss is the lowest when similar under-sampling ratios are applied. As far as our concern, random ray under-sampling in each view usually preserves more information compared to either regular ray under-sampling or regular view under-sampling. In this paper, one of significant goals is to demonstrate the superiority of random ray under-sampling by comparing SVD results with the other two under-sampling strategies.

To calculate SVD for $P_s$, the key step is to compute the matrix element for $A_s = P_s^TP_s$. As long as $A$ symmetric and positive semi-definite, it is possible to calculate all the eigenvalues $\Lambda$ and eigenvectors $X$ of $A$, such that $A_sX = X\Lambda$. Let $P_s = U^s\Sigma^sV_s^T$ be the SVD, $\sigma_{s,i,i} = \lambda_{s,i,i}^{1/2}$ and $V^s = X$. 
6.1.2 Full Projection

First we derive the full projection case as an example. Assume \( A = \mathcal{P}^T \mathcal{P} \), consider the inner product \( \langle Av_{m,l}, v_{n,k} \rangle = \langle \mathcal{P} v_{m,l}, \mathcal{P} v_{n,k} \rangle \).

\[
\langle \mathcal{P} v_{m,l}, \mathcal{P} v_{n,k} \rangle = \frac{4}{\pi \sqrt{(m+1)(n+1)}} \int_{-1}^{1} w(s)U_m(s)U_n(s)ds \int_{0}^{2\pi} e^{i(l-k)\phi} d\phi
\]

\[
= \delta_{l,k} \frac{8}{\sqrt{(m+1)(n+1)}} \int_{-1}^{1} w(s)U_m(s)U_n(s)ds
\]

\( \delta_{l,k} \) is the indicator function: \( \delta_{l,k} = 1 \) only for \( l = k \) or \( \delta_{l,k} = 0 \) otherwise. Recover that \( l \) and \( k \) refer to index of block (sector) in operator \( \mathcal{P} \), we may conclude that only diagonal blocks of \( V^T AV \) contain non-zero elements. Further consider the case that \( l = k \), we have:

\[
\langle \mathcal{P} v_{m,k}, \mathcal{P} v_{n,k} \rangle = \frac{8}{\sqrt{(m+1)(n+1)}} \int_{-1}^{1} w(s)U_m(s)U_n(s)ds
\]

Based on the property of Chebyshev-polynomials, \( \int_{-1}^{1} w(s)U_m(s)U_n(s)ds = \pi/2\delta_{m,n} \), it turns out that only diagonal elements of diagonal blocks can be non-zero. The fact of \( A \) is diagonalized by \( V \) exactly match the results that right singular vector of \( \mathcal{P} \) should be eigenvectors of \( A \). In addition, the diagonal elements \( 4\pi/m + 1 = \sigma^2_{m,l} \) gives the same results stated in [45,46].

Similarly, we will try to compute the inner product \( \langle A_s v_{m,l}, v_{n,k} \rangle \). As \( V \) is capable of diagonalizing \( A_s \), we expect it can in some sense, simplify the matrix structure for \( A_s \) too such that reduce the computational costs required for computing eigenvalues as well as eigenvectors of \( A_s \).

6.1.3 Regular View Under-sampling

In view under-sampling case, the down-sampling operator \( S(\alpha) \) implies the strategy that remove projection information of specific angle. Let \( \bar{B} = \{ \alpha | S(\alpha) = 1, \alpha \in B \} \), the ratio of remaining projection information \( r = |\bar{B}|/|B| \) and down-sampling ratio \( \rho = 1 - r \). Consider the following inner product:

\[
\langle P_s v_{m,l}, P_s v_{n,k} \rangle = \frac{4}{\pi \sqrt{(m+1)(n+1)}} \int_{-1}^{1} w(s)U_m(s)U_n(s)ds \int_{0}^{2\pi} e^{i(l-k)\phi} S(\phi)d\phi.
\]

(6.2)
In practice, the number of total projection view is always limited. Here we consider that at most we may have \( M \) projections with angle \( 2j\pi/M \) for \( j = 1, 2, \ldots, M \) and it is possible that \( M \to \infty \). Then the number of projection views which are preserved in view under-sampling case should be \( N = rM \). To achieve the regular down-sampling, projections are selected uniformly among the total \( M \) available ones, such that projections at angle \( \phi_j = 2j\pi/N \) for \( j = 1, 2, \ldots, N \) the integral \( \int_0^{2\pi} e^{i(l-k)\phi} S(\phi) d\phi \) can be approximated by \( \sum_{j=1}^{N} e^{i(l-k)\phi_j} 2\pi/M \). Therefore, only when \( l - k = tN, t \in \mathbb{Z} \), \( \sum_{j=1}^{N} e^{i(l-k)\phi_j} 2\pi/M = \sum_{j=1}^{N} 2\pi/M = r2\pi \). Otherwise, \( \sum_{j=1}^{N} e^{i(l-k)\phi_j} 2\pi/M = 0 \). Particularly, the diagonal blocks may contain non-zero elements since when \( l = k, t = 0 \in \mathbb{Z} \). However, the elements in off diagonal blocks which satisfy \( l - k = tN \) for view under-sampling may also appear to be non-zero.

Base on the previous results, 6.2 can be simplified as:

\[
\langle P_s v_{m,l}, P_s v_{n,k} \rangle = \delta(l - k) \frac{4}{\pi \sqrt{(m+1)(n+1)}} \int_{-1}^{1} w(s) U_m(s) U_n(s) ds.
\]

In order to satisfy the constraint that \( m \geq l, -m \leq l \leq m \) and \( m+l \) should be even, without loss of generality, may further employ \( 2m+l \) and \( 2n+k \) instead of \( m \) and \( n \) in original formula. Then we may focus on following integral:

\[
\delta(l - k) \frac{4}{\pi \sqrt{(2m+l+1)(2n+k+1)}} \int_{-1}^{1} w(s) U_{2m+l}(s) U_{2n+k}(s) ds. \quad (6.3)
\]

It’s easy to see that 6.3 is non-zero if and only if \( 2m+l = 2n+k \). Thus, in non-zero blocks, only the elements at position \((m, n)\) satisfying \( m - n = -tN/2 \) can be non-zero elements:

\[
\frac{4}{\pi \sqrt{(2m+l+1)(2n+k+1)}} \int_{-1}^{1} w(s) U_{2m+l}(s) U_{2n+k}(s) ds \int_{0}^{2\pi} e^{i(l-k)\phi} S(\phi) d\phi = \frac{4\pi}{2m+k+1}
\]

For the view under-sampling case, \( V \) as transform basis fails to diagonalize \( A_s \). However, it still significantly reduces the computational effort. Only very few blocks in \( A_s \) contain non-zero elements while in each block (assume the size of each block to be \( n_1 \)-by-\( n_1 \)), at most \( n_1 \) elements can be non-zero. In other words, \( V^T A_s V \) should be a very sparse matrix and we only need to compute its no-zero elements in order to decompose \( A_s \). The most computational concerns lies in the step of solving eigenvalue problem for \( V^T A_s V \). The size of \( V^T A_s V \) can be really huge if large number of sectors are considered. We remark here that once the eigenvalue problem is figured
out $V^T A_s VW = W\tilde{\Lambda}$, once can compute the singular values of $P_s$ directly from $\tilde{\Lambda}$ since $\tilde{\Lambda} = \Lambda^*$. In addition, the left singular vectors can be also computed easily by $V^s = VW$.

Compared to $A$ of full projection case, $A_s$ in view under-sampling case is perturbed by non-zeros elements in $(l, k)$ off-diagonal blocks satisfying $l - k = tN$. It’s easy to see that the smaller $N$ is, the more perturbation occurs in $A_s$. On the other hand, when $N \to \infty$, the effect of off-diagonal elements can be ignored and $A_s = A$ if we only consider finite dimensioned case. Remember that $N$ physically denotes the number of projection views preserved in projection system. Small $N$ indicates a large under-sampling ratio which naturally brings sever decay for reconstruction, while large $N$ means low under-sampling ratio which preserves most information of projection system. This illustrate that the results generated for view under-sampling exact matches physical performance of under-sampling projection and reconstruction system.

6.1.4 Regular Ray Under-sampling

Different from view under-sampling that reduces the projection views, however, in regular ray under-sampling case, all the projection views are preserved while x-ray is blocked by specific common regular pattern in each view. Therefore, the down-sampling operator denoted as $S(x)$ should be only related to the position $x$ where x-ray is blocked. Here we also define projection information preserved ratio $r = |\tilde{\Omega}|/|\Omega|$ and down-sampling ratio $\rho = 1 - r$ where $\tilde{\Omega} = \{x|S(x) = 1, x \in \Omega\}$. In this case, we turn to consider the following inner product:

$$\langle P_s v_{m,l}, P_s v_{n,k} \rangle = \frac{4}{\pi \sqrt{(m+1)(n+1)}} \int_{-1}^{1} w(s)U_m(s)U_n(s)S(s)ds \int_{0}^{2\pi} e^{i(l-k)\phi} d\phi. \tag{6.4}$$

Similar to the full projection case, it is easy to get that $\int_{0}^{2\pi} e^{i(l-k)\phi} d\phi = 2\pi \delta_{l,k}$, based on which we may claim that only diagonal blocks of $V^T A_s V$ for regular ray under-sampling can contain non-zero elements. Then, it is possible to simplify and rewrite 6.4 into:

$$\langle P_s v_{m,l}, P_s v_{n,k} \rangle = \frac{8}{\sqrt{(m+1)(n+1)}} \int_{-1}^{1} w(s)U_m(s)U_n(s)S(s)ds.$$
For consistency, $2m + l$ and $2n + k$ should be utilized instead of $m$ and $n$ as it is in view under-sampling case. Therefore, we may further generate the following formula.

$$\langle P_{sv_{m,l}}, P_{sv_{n,k}} \rangle = \frac{8}{\sqrt{(2m + l + 1)(2n + k + 1)}} \int_{-1}^{1} w(s) U_{2m+l}(s) U_{2n+k}(s) S(s) ds.$$  

(6.5)

Similar to full projection case, this regular ray under-sampling strategy also block diagonalized by $V$. However, each diagonal block should not be diagonal any more. Compared to view under-sampling which is perturbed by non-zero elements in off diagonal blocks, this strategy is corrupted by the non-zero element in off-diagonal positions of diagonal blocks. In particular, if data preserving ratio $r \rightarrow 1$, all these off-diagonal elements should be small while diagonal ones become dominant. This fact is again physically meaningful since when only very few x-ray signal is blocked, the reconstruction results are close to ones generated in full projection case.

From the computational prospective, it can be also very efficient to compute the matrix elements for $V^T A_s V$. The main reason is also that $V^T A_s V$ is sparse that only those elements in diagonal blocks can be non-zero. Moreover, compared to view under-sampling strategy, the computational cost required for solving eigenvalue problem of $V^T A_s V$ for regular ray under-sampling should be significantly smaller. It is mainly because that the diagonal block structure makes it possible to solve the eigenvalues and eigenvectors for $V^T A_s V$ block by block individually. Even though the scale of whole system can be very large, it is still efficient to compute the eigenvalue problem for small blocks one by one.

### 6.1.5 Random Ray Under-sampling

Random ray under-sampling indicates the strategy that block x-ray in patterns constructed under random fashion for different projection views. Thus, mathematically, the random down-sampling operator should corresponds to both position variable $x$ and angular variable $\alpha$. Similarly, we may also define the data preserving ratio as well as down-sampling ratio. First let $\bar{q} = \{(x, \alpha)|S(x, \alpha) = 1, x \in \Omega \text{ and } \alpha \in B\}$, then the data preserving ratio $r = |\bar{Q}|/|Q|$ while down-sampling ratio $\rho = 1 - r$. Then we
may consider the inner product for random ray under-sampling:

\[
\langle P_s v_{m,l}, P_s v_{n,k} \rangle = \frac{4}{\pi \sqrt{(m+1)(n+1)}} \int_{-1}^{1} w(s) U_m(s) U_n(s) ds \int_{0}^{2\pi} e^{i(l-k)\phi} S(s, \phi) d\phi.
\]

(6.6)

Again, we first focus on the integral \( \int_{0}^{2\pi} e^{i(l-k)\phi} S(s, \phi) d\phi \) in 6.6. When \( l = k \),

\[
\int_{0}^{2\pi} e^{i(l-k)\phi} S(s, \phi) d\phi = \int_{0}^{2\pi} S(s, \phi) d\phi = 2\pi f(s).
\]

(6.7)

As we know that \( S \) in 6.7 is random binary operator with probability \( r \) to be 1, then \( f \) should be should a function following normal distribution with mean value \( ra \) and variance \( r(1 - r) \): \( f \sim \mathcal{N}(p, p(1 - p)) \).

On the other hand, when \( l \neq k \), according the the random fashion of binary operator \( S \), It has a large probability that the integral \( \int_{0}^{2\pi} e^{i(l-k)\phi} S(s, \phi) d\phi \) equals or closes to 0. Therefore, we set that \( \int_{0}^{2\pi} e^{i(l-k)\phi} S(s, \phi) d\phi = 0 \) for the case that \( l \neq k \).

Now, 6.6 can be further written as:

\[
\delta_{l,k} \frac{8}{\sqrt{(m+1)(n+1)}} \int_{-1}^{1} w(s) U_m(s) U_n(s) f(s) ds.
\]

Then to make it consistent to pervious under-sampling strategy, use \( 2m+l \) and \( 2n+k \) instead, we have:

\[
\delta_{l,k} \frac{8}{\sqrt{(2m+l+1)(2n+k+1)}} \int_{-1}^{1} w(s) U_{2m+l}(s) U_{2n+k}(s) f(s) ds.
\]

(6.8)

Consider when data preserving ratio \( r \to 1 \), function \( f(s) \to 1 \). 6.8 can be approximately written in \( \delta_{l,k} \frac{8}{\sqrt{(2m+l+1)(2n+k+1)}} \int_{-1}^{1} w(s) U_{2m+l}(s) U_{2n+k}(s) ds \) which is exactly same to the case of full projection data.

Similar to the regular ray under-sampling strategy, \( V^T A_s V \) for random under-sampling is also sparse based on our derivation since non-zero elements only appear in diagonal blocks. In addition, it is also possible to solve the eigenvalue problem block-wisely instead of computing for the whole system.

Both strategies of regular under-sampling and random under-sampling are applicable of incorporating the proposed scatter removal method. Scattering effect which is difficult to get rid of usually degrades the image quality for CT reconstruction. Theoretically, the signal received at blocked positions on detector should be scatter signal. According to these signals, it is possible to interpolate the scatter signals
for whole detector plane based on the smooth assumption of scatter. Therefore, the
data by removing the interpolated scatter signal at positions where projection data
is available, improvement in image quality in reconstruction can be expected. In
particular, we focus on performing scatter effect removal by employing random ray
under-sampling since we believe that random under-sampling may outperform the
regular one significantly. Another issue we should point out here is that this scatter
removal method is not applicable in view under-sampling strategy.

6.2 Experimental Results

In previous sections, a possible approach to characterize singular values and right
singular vectors is introduced. However, it is still difficult to tell if random ray
under-sampling possesses any advantage compared to the regular ones. Therefore,
we propose to make comparison of the three under-sampling strategies numerically.
Firstly, the projection operators of all three under-sampling approaches will be evalu-
ated by matrix rank, singular values and singular vectors respectively. Then one step
further, under-sampling CT image reconstruction based on Monte Carlo simulations
is also performed. As traditional FD algorithm fails to handle under-sampling case,
tight frame based model is employed to iteratively reconstruct CT image. Evaluation
of different under-sampling strategies is made by comparing quality of reconstructed
images. In addition, the effect of scatter removal of random ray under-sampling is
also tested in simulation study.

6.2.1 Numerical Study of Under-sampling Operators

As information loss in under-sampling projection is unavoidable, the key issue to
evaluate different under-sampling projection operator becomes that which strategy
preserves most information with the same down-sampling ratio is applied. Motivated
by such consideration, we propose to evaluate under-sampling projection operators
by make comparison of rank, singular values and singular vectors with full projection
operator.

In Figure 6.1, we show ’rank(under-sampling)/rank(full)’ along different under-
Figure 6.1: Comparison of rank sampling ratios for all three under-sampling approaches. Rank reduction in under-sampling indicates serious information loss. From the matrix point of view, rank decreasing is caused by loss of basis in column space (or row space). It brings uncertainty in linear system which is unfavorable in CT reconstruction. From Figure 6.1, we can see clearly that rank of projection operators for both regular ray and view under-sampling decrease significantly with the decreasing of projection data preserved which indicates that projection data loss corrupts and brings serious singularity for linear systems of regular ray and view under-sampling while, the degrading caused in random ray projection operator is much less.

Another important criterion to evaluate under-sampling operator is singular values. Let $\|A\|_* = \sum_i \sigma_i$ where $\sigma_i$ indicates the $i$-th singular value of $A$ and $\|A\|_\sigma = \sigma_{max}$ denotes the general spectral norm with $\sigma_{max}$ as the largest singular value of $A$. It’s easy to observe that:

$$\|P_s\|_* = \|SP\|_* \leq \|S\|_\sigma \|P\|_* = \|P\|_*.$$  

Thus, we try to compare $\|P_s\|_*$ with $\|P\|_*$ in order to show the amount of lost information of singular values. Figure 6.2 provides the $\|P_s\|_*/\|P\|_*$ for the three under-sampling strategies when different under-sampling ratios is applied. If the ratio of preserved data is larger than 25%, regular view under-sampling seems to be the best choice from singular value point of view. However, the performance of random under-sampling is still competitive and it works much better compared with
Figure 6.2: Comparison of singular values

regular ray under-sampling. With the decreasing of ratio of preserved projection data, the advantage of random ray under-sampling becomes more and more obvious. This fact strongly suggests the random ray under-sampling to be the best under-sampling strategy especially for extremely low dose CT imaging.

Moreover, we also compare the right singular vectors. Actually, these singular vectors form a basis in the image domain thus corruption in these singular vectors may cause severe decay in image quality for CT reconstruction task. Here we remark that singular vector respect to 0 singular value does not contribute to reconstruction. Therefore, it is not reasonable to make any comparison of such singular vectors. Based on the results listed in Figure 6.1, both regular ray and view under-sampling may bring in 0 singular values. To make fair comparisons, we try to compare the singular vectors respect to singular values in similar scale from full projection system as well as under-sampling ones. In particular, we are more interested in eigenvectors corresponding to smaller singular values. Recover $f = V \Sigma^{-1} U^T g$ indicates the manner of how to achieve CT reconstruction, for small non-zero singular value $\sigma^{-} = \sigma^{-1}$ can be a relatively large number, such that its right singular vector may play an important role for reconstruction. Figure 6.3 gives some examples of comparison in right singular vectors respect to singular values in similar scale. All the singular values listed in the figure are selected among smallest non-zero singular values for regular view under sampling strategy. Although smaller singular values may exist for projection operator of random ray under-sampling, it is not possible to find non-zero
singular values of similar scale from the regular under-sampling projection systems since all information of these smaller singular values is lost when such under-sampling strategies are applied. From Figure 6.3, it is easy to find out that singular vectors of random ray under-sampling are the most similar ones to singular vectors of full projection operator.

All the numerical results above demonstrate the superiorities of random ray under-sampling compared with regular under-sampling strategies. With the same down-sampling ratio, random under-sampling is always capable of conserving more information. Besides, the linear system of random under-sampling remains to be well-defined even when most of the projection data is blocked out. In addition, random under-sampling also preserves right singular vectors well which outperforms both of the regular under-sampling approaches.

6.2.2 Simulation Study on CT Reconstruction

We conduct Monte Carlo based simulation study on under-sampling CT reconstruction. In order to testing the consistent performance of different under-sampling schemes, we propose to reconstruct CT image with three data preserving ratios: 25% (1/4), 16.7% (1/6) and 10% (1/10). Moreover, we also perform scatter removal for both regular ray under-sampling and random ray under-sampling (recover that scatter removal is not applicable for regular view under-sampling scheme).

6.2.3 Scatter Removal Method

For both ray under-sampling schemes, scattering signal is available at the positions where x-ray is blocked. Thus, it is possible to interpolate scattering signals for whole detector plane based on the smoothness assumption. The main idea of interpolation is assigning missing position by its neighborhood while also taking in account of smoothness. More specifically, let $Sca_{i,j}$ indicates the missing scattering information at position $(i,j)$, we employ information from its neighbor of size $k$ first as initial value:

$$Sca_{i,j} = \frac{\sum_{l\in R, m\in C} Sca_{l,m} \omega(l, m)}{\sum_{l\in R, m\in C, Sca_{l,m} \neq 0} \omega(l, m)},$$
Figure 6.3: Comparison of right singular vectors
Figure 6.4: Effect of interpolation for scattering signals: original scatter (left), blocked scatter (middle) and interpolated scatter (right)

where \( R = \{ l | |l - i| \leq k \} \), \( C = \{ m | |m - j| \leq k \} \) and \( \omega(l, m) = 1/\sqrt{(l - i)^2 + (m - j)^2} \).

Note that if all the elements contained in the neighborhood are missing signals to be interpolated, we just skip it first. Therefore, such interpolation algorithm might be required to repeat for several times until all the missing signals are interpolated. Once all the missing signals in \( Sca \) are interpolated, we may further force smoothness to scatter by optimizing following minimization problem:

\[
\min_Z \frac{1}{2} \| Z - Sca \|^2_F + \beta \| \nabla Z \|^2_F. \tag{6.9}
\]

Here \( \| \cdot \|^2_F \) indicates the famous Frobenius norm in matrix. The first term in (6.9) guarantees the optimal solution \( Z \) to be close to \( Sca \) while the second term requires \( Z \) to be smooth. In figure 6.4, we provide an example of scattering interpolation. It is easy to see that the blocked image is interpolated well by the proposed method. Then remove the interpolated signals from projection image should give result closed to clean projection image. Both regular and random ray under-sampling are applicable to the proposed method which is capable of reducing scattering effect. We report experimental results in CT reconstruction as further illustration in the next part.
6.2.4 CT Reconstruction Results

We propose to employ tight-frame based method [36] to iteratively reconstruct the image from under-sampling projection data. In the experiments, different situations are considered and tested when 25% (1/4), 16.7% (1/6) and 10% (1/10) projection data are preserved for three under-sampling schemes. In each case, parameters are tuned in order to achieving the best quality of image. In addition, for easy comparison, we also reconstruct CT images from full projection data (see Figure 6.5, (a)) as ground truth.

In Figure 6.5, 6.6 and 6.7, reconstruction image of regular ray under-sampling clearly contains more artifacts and image quality is not as good as the other two. The resulting image of regular view under-sampling has relatively low contrast may be due to the scattering effect. The reconstruction image of random under-sampling, on the other hand, gives the best result.

In addition, we also compute the mean squared error (MSE) of all reconstruction images compared with ground truth. In Table 6.1, we report the relative MSE.
Figure 6.6: Reconstruction results where 16.7% of projection data preserved: regular ray under-sampling (a), regular view under-sampling (b) and random ray under-sampling (c).
Figure 6.7: Reconstruction results where 10% of projection data preserved: regular ray under-sampling (a), regular view under-sampling (b) and random ray under-sampling (c).
<table>
<thead>
<tr>
<th>Ratio</th>
<th>Regular in ray</th>
<th>Regular in view</th>
<th>Random in ray</th>
</tr>
</thead>
<tbody>
<tr>
<td>25%</td>
<td>0.1868</td>
<td>0.1686</td>
<td><strong>0.0620</strong></td>
</tr>
<tr>
<td>16.7%</td>
<td>0.1962</td>
<td>0.1752</td>
<td><strong>0.0764</strong></td>
</tr>
<tr>
<td>10%</td>
<td>0.2567</td>
<td>0.1880</td>
<td><strong>0.0957</strong></td>
</tr>
</tbody>
</table>

Table 6.1: MSE compared with grand truth

\(MSE/\|G\|_2\), where \(G\) is used to denote the ground truth). The random ray under-sampling scheme with scatter removed clearly gives the lowest error along different under-sampling ratios.

All the experimental results we have demonstrated the effectiveness of random ray under-sampling scheme as well as the performance of the proposed scatter removal method. With the same under-sampling ratio (or radiation dose), the random ray under-sampling scheme usually gives the best reconstruction results among all the tested under-sampling schemes.

### 6.3 Concluding Remarks on Ray Under-sampling with Scatter Removed

In this work, we provide comparisons of three under-sampling scheme: regular ray, regular view and random ray. Based on the numerical results of rank, singular values and singular vectors, it is easy to draw the conclusion that random under-sampling often outperforms the other two under-sampling schemes. In addition, we propose a novel scatter removal method for ray under-sampling schemes. In the simulation study, we conduct experiments to test all the three under-sampling schemes. The advantages of random ray under-sampling are obvious in both image quality and relative MSE. In addition, we may also conclude that the proposed scattering removal algorithm also performs well for ray under-sampling schemes.
Chapter 7

Conclusion

In this thesis, we mainly introduce five proposed methods for four specific research tasks: multi-label learning (ML), multi-instance multi-label learning (MIML), module learning from large networks and random under-sampling strategy for CT reconstruction. For ML problem, we design a low-rank based method in order to exploring the correlation between different objects which might be one of the most important issues should be considered in ML learning. In addition, a convex sparsity based Markov chain model (Sparse-MIML) is proposed to improve the learning accuracy for MIML tasks while another fast MIML method (OF-MIML) is proposed to target the computational efficiency problem which is bottle neck of most existing MIML algorithm. Later, we have introduced module identification which has become more and more popular in large network learning. Consider network construction is sensitive to both parameter selection and noise, which is an obvious drawback of single network learning, we employ tensor Markov model (TMI) to achieve multiple networks learning. Finally, we consider the ray random under-sampling techniques with application to CT reconstruction. We have shown the numerical advantages of random under-sampling compared with regular ones. In addition, we have proposed a novel scatter removal algorithm based on the ray under-sampling for CT imaging. Experimental results reported in previous parts of this thesis illustrate the superiority of the proposed methods compared with other state-of-art methods.

Among all the proposed methods reported in this thesis, two important regularization terms are employed: sparsity term and low rank term. Both of them have been
widely applied to solve real world problems in many computational aspects such as machine learning, computer vision, pattern recognition, and data mining and so on. In this thesis, they are employed to tackle the label prediction problem for different learning tasks. In fact, we may also treat low rank as sparsity constraint however, in different domain. Normal sparsity terms force the sparsity for matrix elements which indicates the sparsity in matrix domain while low rank actually requires the singular value of matrix to be sparse which force sparsity in singular value domain. It is common to see method that combine different sparsity terms such as element-wise sparsity as we just mentioned, group-wise sparsity (column-wise, row-wise and so on) in matrix domain together to varies applications. What if we combine sparsity both in matrix domain and singular value domain? This might be one thing we may continue exploring in the future.

Moreover, we have not built any regularization in either OF-MIML or TMI. Remember for OF-MIML, we provide an equivalent optimization problem which is convex. Therefore it is easy to modify the model by adding different regularization terms such as low rank and sparsity. In the future, we may also test if the learning accuracy of OF-MIML can be improved with proper regularization as we have mentioned above in concluding remarks on OF-MIML. In addition, the more interesting thing is trying to designing model and algorithms for regularized TMI which involves tensor structure in the model. It is well known that tensor structure problem is usually highly non-linear and non-convex, such that often very difficult to be solved. In the next step, we will try to follow similar scheme that try to reduce tensor structure into matrix to see if we can build and solve the regularized TMI.

In addition, ADMM is the main solver we utilize in this thesis. The convergence of ADMM algorithm is guaranteed for $\beta > 0$ (which is the parameter to control the convergence rate of the algorithm). Practically, the convergence speed is sensitive to different selections of $\beta$. Therefore, one of our future work is to explore how to adaptively tune $\beta$ for faster convergence and better results.

Finally, although CT reconstruction results based on Monte Carlo simulation demonstrate the advantages of random under-sampling techniques, experiments on real data from patient have not been conducted yet. Therefore, we may test the
performance of the under-sampling reconstruction method to real data from patients
to see if the proposed method maintains the potential for clinical application. Also,
we may try to apply all the other proposed learning methods to larger real world
data to test the ability of these methods in practical situations.
Bibliography


CURRICULUM VITAE

Academic qualifications of the thesis author, Mr. SHEN Chenyang:

- Received the degree of Bachelor of Science in Mathematics from School of Mathematical Science, Yangzhou University, September 2010.

- Received the degree of Master of Philosophy in Department of Mathematics, Hong Kong Baptist University, November 2012.

August 2015