MULTIPLE KERNEL AND MULTI-LABEL LEARNING FOR IMAGE CATEGORIZATION

By

Serhat Selçuk Bucak

A DISSERTATION

Submitted to
Michigan State University
in partial fulfillment of the requirements
for the degree of

Computer Science – Doctor of Philosophy

2014

ABSTRACT

MULTIPLE KERNEL AND MULTI-LABEL LEARNING FOR IMAGE CATEGORIZATION

By

Serhat Selçuk Bucak

One crucial step in recovering useful information from large image collections is image categorization. The goal of image categorization is to find the relevant labels for a given image from a closed set of labels. Despite the huge interest and significant contributions by the research community, there remains much room for improvement in the image categorization task. In this dissertation, we develop efficient multiple kernel learning and multi-label learning algorithms with high prediction performance for image categorization.

There are many image representation methods available in the literature. However, it is not possible to pick one as the best method for image categorization, since different representations work better in different scenarios. Multiple kernel learning (MKL), a natural extension of kernel methods for information fusion, is often used by researchers to improve image representation by integrating it to the learning step for selecting and combining different image features. MKL is mostly considered as a binary classification tool, and it is difficult to scale up MKL when the number of labels is large. We address this computational challenge by developing a stochastic approximation based framework for MKL that aims to learn a single kernel combination that benefits all classes.

Another contribution of this dissertation is to develop efficient multi-label learning algorithms. Multi-label learning is arguably the most suitable formulation for the image categorization task. Many researchers have employed decomposition methods, particularly one-vs-all framework, with SVM (support vector machines) as a base classifier for addressing the image categorization problem. However, the decomposition methods have several shortcomings, such as their inability to

exploit label correlations. Further, they suffer from imbalanced data distributions when the number of labels is large. Our contribution is to address multi-label learning via a ranking approach, termed multi-label ranking. Given a test image, multi-label ranking algorithms aim to order all the image classes such that the relevant classes are ranked higher than the irrelevant ones. The advantage of the proposed multi-label ranking approach, termed MLR-L_1 (multi-label ranking with L_1 norm), over other multi-label learning methods is its computational efficiency and high prediction performance.

Image categorization is a supervised learning task, thus requiring a large set of training images annotated by humans. Unfortunately, labeling is an expensive process, and it is often the case that the annotators provide a limited set of labels, meaning that they only give a small subset of relevant tags for an image. One of the contributions of this dissertation is defining the problem of multi-label learning with incomplete class assignments and presenting a robust multi-label ranking algorithm, termed MLR-GL (multi-label ranking with group lasso norm), that addresses the challenge of learning from incompletely labeled data.

Finally, we present a multiple kernel multi-label ranking algorithm to simultaneously address two essential factors for improving the performance of image categorization: Heterogeneous information fusion, and exploiting label correlations in multi-label data. We propose a multiple kernel multi-label ranking method that learns a shared sparse kernel combination that benefits all image classes. This way, we not only improve the training and prediction efficiency, but also improve the accuracy, particularly for classes with a small number of samples, by enabling information sharing between classes. We integrate the proposed MLR-L_1 algorithm with an efficient semi-infinite linear programming (SILP) based MKL solver and develop a computationally efficient wrapper algorithm, termed MK-MLR (multiple kernel multi-label ranking).

To Dani

ACKNOWLEDGMENTS

I would like to express the deepest appreciation to my thesis advisor, Professor Anil K. Jain, for his continuous support, generosity, patience, enthusiasm, and wisdom. Being his student and a part of the PRIP Lab is something that will always make me feel proud and privileged. It has been a great opportunity for me to work with such an intelligent, hard-working and renowned researcher like Professor Jain, and I have tried to gain as much as possible from his immense knowledge of pattern recognition and life.

I am thankful to Professor Rong Jin for working closely with me during my PhD. I was very fortunate to work with a such a smart, disciplined, and knowledgeable researcher, and collaborating with him taught me the importance of passion and hard work in research. I am grateful to have Professor Selin Aviyente and Professor Pang-Ning Tan on my thesis committee. Their valuable comments and suggestions helped me to improve my thesis. I would also like to thank Professor Todd Fenton and Professor Roger Haut for supporting me in my last year of PhD under the National Institute of Justice grant and giving me the opportunity to work with them in the pediatric fracture printing project. I would also like to thank Professor George Stockman for the valuable advice he gave to me throughout my PhD.

I thankfully acknowledge the funding sources that made my Ph.D. work possible. My research was supported by grants from the Office of Naval Research, ONR N00014-09-1-0663. I was funded by the National Institute of Justice grant, NIJ Award No. 2011-DN-BX-K540, in my last year.

Professor Bilge Gunsel is a very important person for me. I started working with her in my senior year and continued to study under her supervision for my MSc degree at Istanbul Technical University. Her generosity, support, and passion for research helped me to have very rewarding and pleasant time at ITU. Working with her was one of the main factors that encouraged me to

pursue a PhD.

I was fortunate to have great collaborations outside MSU. It was a very valuable learning experience for me to work at IBM with Vikas Sindhwani and Jianying Hu. I also had a very fruitful internship experience at Samsung working with Ankur Saxena, Abhishek Nagar, Felix Fernandes, and Kong-Posh Bhat. I also had the pleasure of working on a research paper with Professor Akgul from ITU.

I would like to thank the fellow PRIP students and friends: Soweon, Brendan, Pavan, Abhishek, Radha, Jung-Eun, Kien, Alessandra, Tim, Sunpreet, Scott, Lacey, Charles, Unsang, and Mayur. They made my life at MSU easier and more fun. I also consider myself fortunate and honored to work on research papers with Pavan, Brendan, and Abhishek. Ali Mutlu, Mehrdad Mahdavi and Jen Vollner are other fellow PhD students that I want to thank.

Sezai Turkes is another person I need to thank, not only for the school he created that provided an excellent education and seven fun years for me, but also for his generosity and vision, which were always a source of motivation.

Last but not least, I want to thank my families in US and in Turkey. My parents-in-law Shari and Tom made my life in Michigan much easier with their kindness and generosity. I am grateful to have three great siblings, Efkan, Serhan, and Tuba, who gave me support and encouragement whenever I needed. My mother and father have been providing me a constant support with endless patience during my long years of study, and it is not possible to thank them enough. Finally, I would like to thank my dear wife Danielle for making my life much more beautiful.

TABLE OF CONTENTS

LIST OF TABLES . . . . . **x**

LIST OF FIGURES . . . . . **xiv**

Chapter 1 Introduction **1**
1.1 Multiple Kernel Learning for Image Categorization 2
1.2 Multi-label Learning for Image Categorization 4
1.3 Challenges 5
1.3.1 Challenges in MKL for Image Categorization 6
1.3.2 Challenges in Multi-label Learning for Image Categorization 7
1.4 Contributions 8
1.5 Notation 15

Chapter 2 Multiple Kernel Learning for Image Categorization: A Review **17**
2.1 Introduction 17
2.2 Overview 19
2.2.1 Overview of Multiple Kernel Learning (MKL) 20
2.2.2 Relationship to the Other Approaches 21
2.3 Multiple Kernel Learning (MKL): Formulations 23
2.3.1 Multiple Kernel Learning and Group Lasso 24
2.3.2 Regularization in MKL 26
2.4 Multiple Kernel Learning: Optimization Techniques 28
2.4.1 Direct Approaches for MKL 29
2.4.1.1 A Sequential Minimum Optimization (SMO) based Approach for MKL 29
2.4.2 Wrapper Approaches for MKL 30
2.4.2.1 A Semi-infinite Programming Approach for MKL (MKL-SIP) 30
2.4.2.2 Subgradient Descent Approaches for MKL (MKL-SD & MKL-MD) 31
2.4.2.3 An Extended Level Method for MKL (MKL-Level) 32
2.4.2.4 An Alternating Optimization Method for MKL (MKL-GL) 33
2.4.3 Online Learning Algorithms for MKL 33
2.4.4 Computational Efficiency 34
2.5 Experiments 35
2.5.1 Data sets, Features and Kernels 35
2.5.2 MKL Methods Used in Comparison 37
2.5.3 Implementation 38
2.5.4 Classification Performance of MKL 39
2.5.4.1 Experiment 1: Classification Performance 39

2.5.4.2 Experiment 2: Number of Kernels vs. Classification Accuracy . . . 42
2.5.5 Computational Efficiency . . . 43
2.5.5.1 Experiment 4: Evaluation of Training Time . . . 43
2.5.5.2 Experiment 5: Evaluation of Sparseness . . . 45
2.5.6 Large-scale MKL on ImageNet . . . 46
2.6 Summary and Conclusions . . . 47

Chapter 3 Multi-label Multiple Kernel Learning by Stochastic Approximation . . . 59
3.1 Introduction . . . 59
3.2 Previous Work . . . 60
3.3 Multi-label Multiple Kernel Learning (ML-MKL) . . . 62
3.3.1 A Minimax Framework for Multi-label MKL . . . 64
3.3.2 Convergence Analysis . . . 67
3.4 Experimental Results . . . 68
3.4.1 Data Sets . . . 68
3.4.2 Baseline Methods . . . 69
3.4.3 Implementation . . . 70
3.4.4 Classification Performance . . . 70
3.4.5 Training Time . . . 74
3.4.6 Sensitivity to Parameters . . . 80
3.4.7 Large-scale MKL on ImageNet . . . 81
3.5 Conclusions and Future Work . . . 84

Chapter 4 Image Categorization by Multi-label Ranking . . . 87
4.1 Introduction . . . 87
4.2 Previous Work . . . 88
4.2.1 Label Set Transformation Methods . . . 89
4.2.1.1 Problem Transformation Methods . . . 89
4.2.1.2 Label Set Projection Methods . . . 90
4.2.2 Supervised Algorithm Adaptation Methods . . . 92
4.2.2.1 Transfer learning for multi-label classification . . . 93
4.2.3 Multi-label Ranking Methods . . . 93
4.2.4 Exploiting Label Correlation in Multi-label Learning . . . 94
4.2.5 Related Problems . . . 95
4.3 Maximum Margin Framework for Multi-label Ranking . . . 96
4.4 Approximate Formulation . . . 97
4.4.1 Relation to the One-vs-all Approach . . . 98
4.4.2 Proposed Approximation . . . 99
4.5 Efficient Algorithm . . . 102
4.6 Experimental Results . . . 103
4.6.1 Data Sets . . . 103
4.6.2 Baseline Methods . . . 104
4.6.3 Multi-label Ranking Performance . . . 105

4.6.4 Training Time . . . 110
4.7 Conclusions and Future Work . . . 113

Chapter 5 Multi-label Ranking for Image Categorization with Incomplete Class Assignments . . . 116
5.1 Introduction . . . 116
5.2 A Framework for Multi-label Learning from Incompletely Labeled Data . . . 120
5.3 Optimization Algorithm . . . 122
5.4 Experimental Results . . . 127
5.4.1 Data Sets . . . 127
5.4.2 Baseline Methods . . . 128
5.4.3 Multi-label Ranking Performance on Incompletely Labeled Data . . . 130
5.4.4 Training Time . . . 132
5.5 Conclusions and Future Work . . . 134

Chapter 6 Multiple Kernel Multi-label Ranking . . . 138
6.1 Introduction . . . 138
6.2 Previous Work . . . 140
6.3 Multiple Kernel Multi-Label Ranking (MK-MLR) . . . 141
6.3.1 A Minimax Framework for Multiple kernel Multi-label Ranking . . . 141
6.3.2 Proposed Approximation . . . 144
6.3.3 Optimization via Semi-infinite Linear Programming . . . 145
6.4 Experimental Results . . . 146
6.4.1 Data Sets . . . 146
6.4.2 Baseline Methods . . . 147
6.4.3 Implementation . . . 148
6.4.4 Evaluation Measures . . . 148
6.4.5 Multi-label Learning Performance . . . 149
6.4.6 Training Efficiency . . . 155
6.4.7 Prediction Efficiency . . . 161
6.5 Conclusions and Future Work . . . 163

Chapter 7 Contributions and Future Work . . . 165
7.1 Contributions . . . 165
7.2 Future Work . . . 168

APPENDIX . . . 169

BIBLIOGRAPHY . . . 196

LIST OF TABLES

Table 1.1 Multi-label ranking performance (AUC-ROC) for the ESP Game and MIR Flickr25000 data sets 12

Table 1.2 AUC-ROC (%) scores for the ESP Game and MIR Flickr25000 data sets for the missing label scenario. 13

Table 1.3 The list of symbols used in this dissertation 16

Table 2.1 Comparison of MKL baselines and simple baselines ("Single" for single best performing kernel and "AVG" for the average of all the base kernels) in terms of classification accuracy. The last three columns give the references in which either "method1" or "method2" performs better, or both methods give comparable results, respectively. 18

Table 2.2 Comparison of computational efficiency of MKL methods. The last three columns give the references, where "method1" is better, "method2" is better, or both give similar results. 19

Table 2.3 Description of the 48 kernels built for the Caltech 101 data set. 36

Table 2.4 Classification results (MAP) for the Caltech 101 data set. We report the average values over five random splits and the associated standard deviation. 40

Table 2.5 Classification results (MAP) for the VOC 2007 data set. We report the average values over five random splits and the associated standard deviation. 41

Table 2.6 Comparison with the state-of-the-art performance for object classification on the Caltech 101 (measured by classification accuracy) and VOC 2007 data sets (measured by MAP). 42

Table 2.7 Comparison of training time between MKL-SMO and MKL-SIP 45

Table 2.8 Total training time (seconds), number of iterations, and total time spent on combining the base kernels (seconds) for different MKL algorithms vs. number of training examples for Caltech 101. 49

Table 2.9 Total training time (seconds), number of iterations, and total time spent on combining the base kernels (seconds) for different MKL algorithms vs. number of training examples for the VOC 2007 data set. 56

Table 2.10 Total training time (seconds), number of iterations, and total time spent on combining the base kernels (seconds) for different MKL algorithms vs. number of base kernels for the Caltech 101 data set. 57

Table 2.11 Total training time (seconds), number of iterations, and total time spent on combining the base kernels (seconds) for different MKL algorithms vs. number of base kernels for the VOC 2007 data set. 58

Table 3.1 Classification results (MAP) for the Caltech 101 data set. We report the average values over five random splits and the associated standard deviation. 71

Table 3.2 Classification results (MAP) for the VOC 2007 data set. We report the average values over five random splits and the associated standard deviation. 72

Table 3.3 Training time (seconds) for the Caltech 101 data set. We report the average values over five random splits and the associated standard deviation. 74

Table 3.4 Training time (seconds) for the VOC 2007 data set. We report the average values over five random splits and the associated standard deviation. 75

Table 4.1 AUC-ROC and MAP results for the VOC 2007 data set 106

Table 4.2 AUC-ROC (%) for the ESP Game data set with 10,000 training images 107

Table 4.3 MAP (%) for the ESP Game data set with 10,000 training images 108

Table 4.4 AUC-ROC and MAP results for the MIR Flickr25000 data set 110

Table 4.5 The label predictions by the baselines for four images from the ESP Game data set. The first row under the images gives the true image class labels. For each baseline, we provide the top six returned labels (three in the top row, and three in the lower row) ranked from left to right. The hits are written with bold characters. . 111

Table 5.1 Some concepts that can be confused with the incomplete label assignment problem 119

Table 5.2 AUC-ROC (%) for the ESP Game data set with 10,000 training images and 200 classes. 128

Table 5.3 MAP (%) for the ESP Game data set with 10,000 training images and 200 classes. 128

Table 5.4 The label predictions by the baselines for four images from the ESP Game data set, when 40% of the training labels are missing. The first row under the images gives the true image class labels. For each baseline, we provide the top nine returned labels (three in the top row, and three it the lower row) ranked from left to right. The hits are written with bold characters. 129

Table 5.5 AUC-ROC results for the MIR Flickr data set 129

Table 5.6 Examples of training images from the ESP Game data set with true labels and annotations generated by different multi-label learning methods. Only the underlined true labels are provided to the methods for training. For each method, the correct (returned) keywords are highlighted by bold font whereas the incorrect ones are highlighted by italic font. 136

Table 5.7 Examples of test images from the ESP Game data set with annotations generated by different multi-label learning methods. The correct keywords are highlighted by bold font whereas the incorrect ones are highlighted by italic font. 137

Table 6.1 The change of category based AUC score (%) withe respect to the number of selected classes for a subset of the ESP Game data set with 2,500 training images. . 149

Table 6.2 The change of image based AUC score (%) withe respect to the number of selected classes for a subset of the ESP Game data set with 2,500 training images. . 150

Table 6.3 The change of category based AUC score (%) withe respect to the number of selected classes for a subset of the MIR Flickr data set with 6,250 training images. . 150

Table 6.4 The change of image based AUC score (%) withe respect to the number of selected classes for a subset of the MIR Flickr data set with 6,250 training images. . 151

Table 6.5 The change of category based AUC score (%) with respect to the number of training samples for a subset of the ESP Game data set. The AUC score is calculated using the top 200 classes. 153

Table 6.6 The change of image based AUC score (%) with respect to the number of training samples for a subset of the ESP Game data set. The AUC score is calculated using the top 200 classes. 153

Table 6.7 The change of category based AUC score (%) with respect to the number of training samples for a subset of the MIR Flickr data set. The AUC score is calculated using the top 200 classes. 154

Table 6.8 The change of image based AUC score (%) with respect to the number of training samples for a subset of the MIR Flickr data set. The AUC score is calculated using the top 200 classes. 155

Table 6.9 Sparsity (%) of kernel weights and dual variables for the multiple kernel baselines and the resulting prediction times. These results are obtained from a subset of the ESP Game data set with $5,000$ training images and 200 classes. 163

Table A.1 A list of techniques that can be used in each module of the Bag-of-Words (BOW) model 172

Table A.2 Data set statistics 173

LIST OF FIGURES

Figure 1.1 The first column shows the surface graphs that demonstrate the influence of different kernel combination weights on the mean average precision score for three different classes. Four examples from each class are given in the second column. For interpretation of the references to color in this and all other figures, the reader is referred to the electronic version of this thesis. 3

Figure 1.2 Illustration of some image categorization challenges: (a) Blue Mosque under two different illumination conditions, (b) two miniatures with background clutter and object deformation, (c) two different views of the Topkapi Palace, (d) two ferry images, one being partially occluded. 6

Figure 1.3 In Chapter 2, we discuss binary MKL methods for the one-vs-all framework, where an individual MKL algorithm is trained for each class. 9

Figure 1.4 In Chapter 3, we present our multi-label MKL algorithm, which solves one MKL problem for all classes. 10

Figure 1.5 The difference between the two proposed multi-label ranking approaches MLR-L_1 (Chapter 4) and MLR-GL (Chapter 3) is that MKL-L_1 strictly addresses the complete class assignment problem whereas MLR-GL can handle missing class assignments. For example, the complete and full annotations are provided with all four labels (*soccer, referee, field, goalkeeper*) for the given image. 13

Figure 1.6 The difference between the two proposed multi-label ranking approaches (a) MLR-L_1 (Chapter 4) and (b) MLR-GL (Chapter 3) is that MKL-L_1 strictly addresses the complete class assignment problem whereas MLR-GL can handle missing class assignments. For example, only two labels (*soccer* and ***field***, written with bold characters) are given for the above image whereas two labels (*goalkeeper* and *referee*, underlined text) are missing. 14

Figure 2.1 A summary of representative MKL optimization schemes 50

Figure 2.2 Mean average precision (MAP) scores of different L_1-MKL methods vs. number of iterations for the *anchor* class of the Caltech101 data set. 51

Figure 2.3 Mean average precision (MAP) scores of different L_1-MKL methods vs. number of iterations for the *bonsai* class of the Caltech101 data set. 51

Figure 2.4 Mean average precision (MAP) scores of different L_1-MKL methods vs. number of iterations for the *camera* class of the Caltech101 data set. 52

Figure 2.5 The change in MAP score with respect to the number of base kernels for the Caltech 101 data set. 52

Figure 2.6 The change in MAP score with respect to the number of base kernels for the VOC 2007 data set. 53

Figure 2.7 Number of active kernels learned by the MKL-SIP algorithm vs. number of iterations for the Caltech 101 data set. Note that it is difficult to distinguish the results of L_2-MKL and L_4-MKL from each other as they are identical. 53

Figure 2.8 Number of active kernels learned by the MKL-SIP algorithm vs. number of iterations for the VOC 2007 data set. Note that it is difficult to distinguish the results of L_2-MKL and L_4-MKL from each other as they are identical. 54

Figure 2.9 Classification performance for different training set sizes for the ImageNet data set. 54

Figure 2.10 Training times for L_1-MKL and L_2-MKL on different training set sizes for the ImageNet data set. 55

Figure 3.1 For the 4 classes (*ant, butterfly, ceiling fan, chair*) taken from the Caltech 101 data set, the first row gives images which produced false negatives for the single kernel baseline and true positives for ML-MKL-SA baseline. The second row gives images which produced false positives for the single kernel baseline and true negatives for the ML-MKL-SA baseline for the corresponding classes. 69

Figure 3.2 For the 4 classes (*bird, potted plant, dining table, train*) taken from the VOC 2007 data set, the first row gives images which produced false negatives for the single kernel baseline and true positives by the GMKL baseline. The second row gives images which produced false positives for the single kernel baseline and true negatives for the ML-MKL-SA method for the corresponding classes. 71

Figure 3.3 The evolution of kernel weights computed by the MKL-Level method over time for the Caltech 101 data set with 30 training instances per class. 76

Figure 3.4 The evolution of kernel weights computed by the MKL-SIP-L_1 method over time for the Caltech 101 data set with 30 training instances per class. 77

Figure 3.5 The evolution of kernel weights computed by the ML-MKL-Sum method over time for the Caltech 101 data set with 30 training instances per class. 78

Figure 3.6 The evolution of kernel weights computed by the ML-MKL-SA method over time for the Caltech 101 data set with 30 training instances per class. 79

Figure 3.7 Classification performance (MAP) of the proposed algorithm ML-MKL-SA on Caltech 101 with 30 training instances per class using different values of δ (for $\eta_\beta = \eta_\gamma = 0.01$). 80

Figure 3.8 Classification performance (MAP) of the proposed algorithm ML-MKL-SA on Caltech 101 with 30 training instances per class using different values of η_β (for $\eta_\gamma = 0.0001$ and $\delta = 0.2$). 81

Figure 3.9 Classification performance (MAP) of the proposed algorithm ML-MKL-SA on Caltech 101 with 30 training instances per class using different values of η_γ ($\eta_\beta = 0.0001$ and $\delta = 0.2$). 82

Figure 3.10 Comparison of the mean average precision scores for different training set sizes for the ImageNet data set. 83

Figure 3.11 Comparison training times for different training set sizes for the ImageNet data set. 84

Figure 4.1 A diagram summarizing the label set projection schemes for multi-label learning. 91

Figure 4.2 For four images from the VOC 2007 data set, the original labels are given in addition to the outputs of baseline methods. 106

Figure 4.3 Change of the AUC-ROC score with respect to the number of training images. . 109

Figure 4.4 Training time of the three baselines for a fixed number of categories (100) with respect to the number of training samples for the ESP Game data set. 111

Figure 4.5 Training time of the three baselines for a fixed number of training samples (10,000) with respect to the number of categories for the ESP Game data set. 112

Figure 5.1 Some example images from the VOC 2007 (top row) and ESP Game (bottom row) data sets with their annotations. The labels written in italic are provided with the images, whereas the ones written in bold fonts are the missing labels. These images, with their missing annotations, are examples of incomplete labeled data. . 117

Figure 5.2 Example images from the ESP Game data set and their annotations. The annotations highlighted by bold font, which are used to annotate the same concept/object in the corresponding images, are examples of the label ambiguity problem. 119

Figure 5.3 The change in the baseline training times (seconds) with respect to the number of training images from the ESP Game data set. 133

Figure 5.4 The change in the training time (seconds) for the proposed multi-label ranking algorithms and one-vs-all SVM with respect to the number of image labels (m). . . 134

Figure 6.1 The plot of recall vs. number of retrieved labels per image. The number of training images is $2,500$. 152

Figure 6.2 Comparing MK-MLR to ML-MKL methods that learn optimal kernel combination separately for each class in terms of training time. We use $5,000$ training images and create four different settings by changing the number of classes $\{50, 100, 200, 500\}$ 156

Figure 6.3 Comparing MK-MLR to ML-MKL methods that learn one optimal kernel combination for all classes in terms of training time. We use $5,000$ training images and create four different settings by changing the number of classes $\{50, 100, 200, 500\}$ 157

Figure 6.4 Comparing MK-MLR to ML-MKL methods that learn one optimal kernel combination separately for each class in terms of training time. We use images from 200 classes and create three settings by changing the data set size $\{1,000, 2,500, 5,000\}$ 159

Figure 6.5 Comparing MK-MLR to ML-MKL methods that learn one optimal kernel combination for all classes in terms of training time. We use images from 200 classes and create three settings by changing the data set size $\{1,000, 2,500, 5,000\}$ 160

Figure A.1 Four example images from the Caltech 101 data set with their labels. 174

Figure A.2 Four example images from the ImageNet data set. A *cat* and a *car* image are shown in the top row. The second row has two dog images, one from the *dalmatian* synset, and one from the *Mexican hairless* synset 175

Figure A.3 Two example images from the MIR Flickr data set. Left image (reflection effect) is by Szymczak [1] and the right image (fish eye effect) is by Wild. [2] . . . 176

Figure A.4 Four example images from the ESP Game data set. 177

Chapter 1

Introduction

In this dissertation, we develop multiple kernel and multi-label learning algorithms for the image categorization problem. The goal of image categorization is labeling an image with the relevant categories from a predefined tag set. In other words, image categorization requires desinging classifiers to ask the following type of question: "Does the query image have a *cat* in it?" Answering questions such as this (*cat* is one of the possible image labels) is also the goal of visual object recognition and automatic image annotation tasks, which we consider as two very closely related subsets of image categorization. Visual object recognition is defined as the task of determining if any of the predefined objects (visible or tangible things) are present in an image or not. On the other hand, automatic image annotation task differs from visual object recognition in that the goal is not only to look for the existence of tangible objects, but also concepts like color (*green, white*), place (*Paris, Ireland*), and scene (*sunset, fight*). The methods we present in this dissertation are designed to be used in both of these tasks.

Image categorization is a very good fit as a benchmark to test multiple kernel and multi-label learning algorithms for several reasons. Firstly, we see that many state-of-the-art methods for image categorization use information fusion to combine different image representations. Therefore, multiple kernel learning (MKL), which is an information fusion technique, is expected to perform

well in image categorization. Secondly, different classes in image categorization data sets require using similar features (i.e., the scale-invariant feature transform, SIFT, works well for the majority of image classes). Therefore, the assumption we use for our multiple kernel learning algorithms holds, which is a kernel combination that benefits all classes can be learned. Thirdly, only a small number of image representations are needed to obtain the optimal classification performance. This means that sparseness, one of the goals of the multiple kernel learning algorithms we develop, is a useful feature in image categorization. Fourthly, since image classes are often correlated with each other, multi-label learning is expected to work well with image categorization. Finally, incompletely labeled data, which is one of the problems we address in this dissertation, frequently occur in image categorization applications.

1.1 Multiple Kernel Learning for Image Categorization

Given the variety of alternatives and the large number of ways for constructing image representations, one critical issue in developing statistical models for image categorization is how to effectively combine different image features. MKL presents a principled framework for combining multiple image representations: It creates a set of base kernels for each representation and finds the optimal kernel combination via a linear combination of kernels.

We demonstrate MKL on a simple image categorization problem. We create two kernels: one based on color histogram and one based on texture distribution in the image. We choose three object classes (*crocodile, snoopy, strawberry*) from the Caltech 101 data set [3], each with 15 instances, and train one-vs-all support vector machines (SVM) for each of the three classes by using different combinations of these two kernels. To combine the kernels, we vary the combination coefficients in the set $\{0, 0.2, 0.4, 0.6, 0.8, 1\}$. In Figure 1.1 we generate a heat map to represent classification performance of different linear combinations of the two kernels. We observe that the optimal combination varies from one class to another. For example, while the texture based kernel

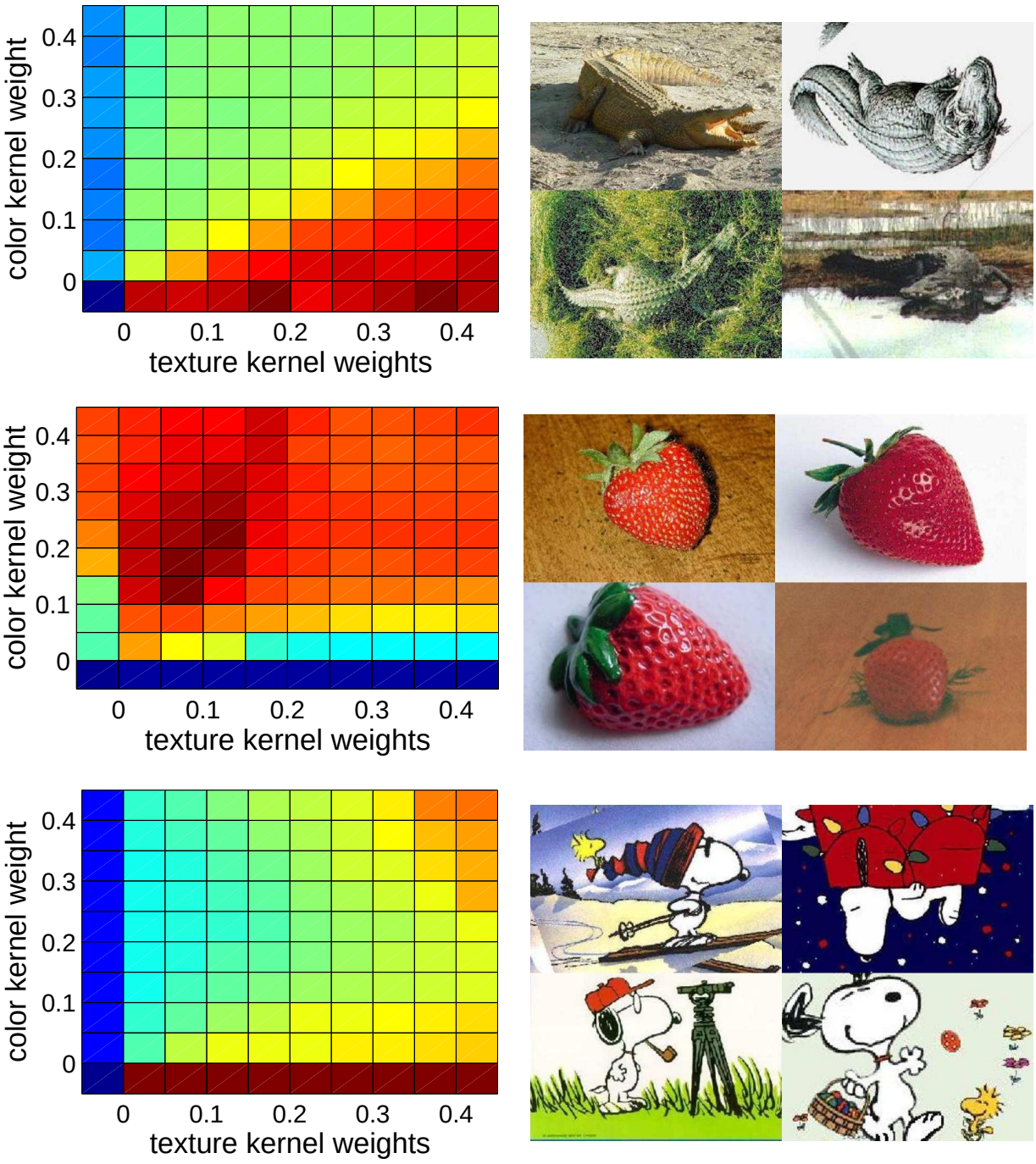

Figure 1.1: The first column shows the surface graphs that demonstrate the influence of different kernel combination weights on the mean average precision score for three different classes. Four examples from each class are given in the second column. For interpretation of the references to color in this and all other figures, the reader is referred to the electronic version of this thesis.

is assigned a higher coefficient for crocodile classification task, the color kernel should be used with a higher weight for the strawberry class. This simple example illustrates the significance of identifying the appropriate combination of kernels for recognizing a specific class of visual objects. It also motivates the need for developing automatic approaches for finding the optimal combination of kernels from training examples, as there is no universal solution for kernel combination that works well for all classes.

MKL has been successfully applied to a number of tasks in computer vision, particularly to image categorization. For instance, the winning group in the Pascal VOC 2010 object categorization challenge [4] used MKL to combine multiple sets of visual features. The best performance reported on the Caltech 101 data set was achieved by learning the optimal combination of multiple kernels [5]. Recent studies have also shown promising performance of MKL for object detection [6].

1.2 Multi-label Learning for Image Categorization

In multi-label learning, more than one class can be assigned to an instance. With the increase in the number of data sets where each image has multiple labels, there have been a vast amount of studies that focus on developing strong classification methods for image categorization [7–9]. Many researchers employ decomposition methods, particularly one-vs-all framework, with SVM as a base classifier. In this setting, a separate classifier is trained for each image label, leading to an independent prediction for each label on a query image. Although decomposition based methods are frequently used to solve multi-label classification, they do have some limitations (see Chapter 4). To overcome the limitations of decomposition techniques, there have been many direct multi-label learning methods proposed in the literature that do not decompose or transform the multi-label learning problem into a set of binary classification tasks [10–14]. In this dissertation, we are particularly interested in multi-label ranking, in which the learning task is formulated as a bipartite

ranking problem. Multi-label ranking is an example of a direct multi-label learning approach that can exploit label correlations. Also, by avoiding a binary decision, multi-label ranking is usually more robust than the classification approaches, particularly when the number of classes is very large [10, 15].

Ranking has been successfully used in other application domains such as document classification and recommender systems. For example, it makes more sense in recommender systems to provide the user an ordered list of items that she/he might be interested in. Also, since the preference ratings given by the users are not universal (i.e., the rating "7" is not same for every user) ranking results would be easier to obtain compared to predicting the exact ratings. Similarly, ranking labels might be useful for image categorization systems. Consider an image search system where the search is based on image labels. Being able to rank image labels can be useful for refining the search. For example, if a user is interested in finding "cafe shop" images from the internet to decide where to go, then a system that only focuses on the label "cafe shop" would not help in refining the search. If the user is looking for images of pet-friendly cafe shops where more people read books than use computers, then ranking labels would be useful. Such a system would aim to retrieve images where the labels *cafe shops, books, cats, dogs,* have higher scores than the label *computer*. This does not mean that the image should not contain any computers, but the emphasis on the other labels is set to be higher.

1.3 Challenges

There are thousands of possible image classes and as such, there is no optimal image representation technique that would work best for all of these classes. In fact, it is very difficult to find a salient representation for even a single image class due to large variations in the visual appearance of samples within a class, a phenomenon known as the intra-class variation problem [16, 17]. In addition to intra-class variation, challenges include translation [18], scale [19], rotation [20],

illumination problem

background clutter

viewpoint variation

viewpoint occlusion

Figure 1.2: Illustration of some image categorization challenges: (a) Blue Mosque under two different illumination conditions, (b) two miniatures with background clutter and object deformation, (c) two different views of the Topkapi Palace, (d) two ferry images, one being partially occluded.

affine transformation [21], viewpoint variation [22], occlusion [23], background clutter [24], and illumination [25]. Figure 1.2 shows example images that demonstrate some of these challenges.

The problems we have stated above often force recognition algorithms to utilize complex models. More specifically, kernel machines, which use non-linear functions of the features, generally work better than linear classification models. For instance, we see from the image categorization literature that using SVM with RBF (radial basis function) or χ^2 kernel gives superior performance compared to linear SVM [26]. However, there are some challenges of using kernel machines for image categorization. We examine these under the following two topics: (i) challenges of multiple kernel learning and (ii) challenges of multi-label learning for image categorization.

1.3.1 Challenges in MKL for Image Categorization

- The application of MKL to multi-labeled data, such as in image categorization, is primarily limited to one-vs-all framework, which fails to exploit label correlations. As MKL solvers for each class operate independently, no interaction or information transfer between image

classes takes place, leading to suboptimal performance [15, 27, 28].

- The training complexities of MKL algorithms are quadratic in terms of the number of training samples and linear in terms of the number of classes. More importantly, the prediction is computationally expensive. Once the distance between a query sample and the support vectors is calculated, a different kernel combination needs to be calculated for each class prior to prediction, which is a costly process.

1.3.2 Challenges in Multi-label Learning for Image Categorization

- Exploiting correlations or dependencies between different classes is an important research problem, and a number of approaches have been developed for multi-label learning that aim to capture dependencies among classes [10, 12, 13, 29, 30]. The majority of such methods make strong assumptions regarding the type of relationships that exist between class labels. Although these methods give promising results when the underlying assumptions hold, there is no guarantee that the assumptions would hold for all types of data.

- Formulating a multi-label learning problem as multi-label ranking methods is an effective approach that takes advantage of the label correlations without making a strong assumption about the data structure. However, the bipartite ranking constraints make the computational complexity quadratic in the number of classes, making these algorithms computationally inefficient when the number of classes is large.

- It is unclear if strong multi-label learning algorithms would work well in practice. One of the main concerns for real world systems is that the labeling process is very expensive and often inaccurate. In image categorization systems, the image annotations for the training data set are provided primarily by online users through services like Amazon Mechanical Turk [31]. As a result, the retrieved annotations are often incomplete; only a subset of the true image

labels is given by the annotators. Therefore, it is important to build robust classifiers that would work well even when the full label information is not provided.

1.4 Contributions

We can divide our contributions in this dissertation into two parts: (i) multiple kernel learning and (ii) multi-label learning for image categorization. Chapters 2 and 3 show how multiple kernel learning can be used to simultaneously improve the representation and learning stages. Chapters 4 and 5 discuss the multi-label learning problem, which is arguably the most appropriate formulation of the image categorization problem. We present our (single) kernel based multi-label learning algorithms in Chapters 4 and 5. Finally, we merge these two directions by developing a multiple kernel multi-label ranking approach in Chapter 6 and address our main goal, which is to develop efficient algorithms that outperform published classification methods when state-of-the-art image representations are used. We can list our contributions as follows:

- Our contribution in Chapter 3 is to improve the computational efficiency of MKL with respect to the number of classes for both the training and prediction steps. The majority of MKL methods require executing a binary MKL algorithm individually for each image class, see Figure 1.3, making the training and prediction complexities linear in terms of the number of classes. This is the reason that the existing MKL solvers do not scale well when the number of classes is large. We address this computational challenge by developing a framework for MKL that learns a single kernel combination benefiting all classes by combining a worst-case analysis with stochastic approximation (see Figure 1.4). Our analysis shows that the training complexity of our algorithm is $O(m^{1/3} \log m)$ in terms of the number of classes, m. Moreover, since our algorithm learns a single sparse kernel combination for all classes, the time consumed for the kernel construction step of the prediction phase is also reduced significantly.

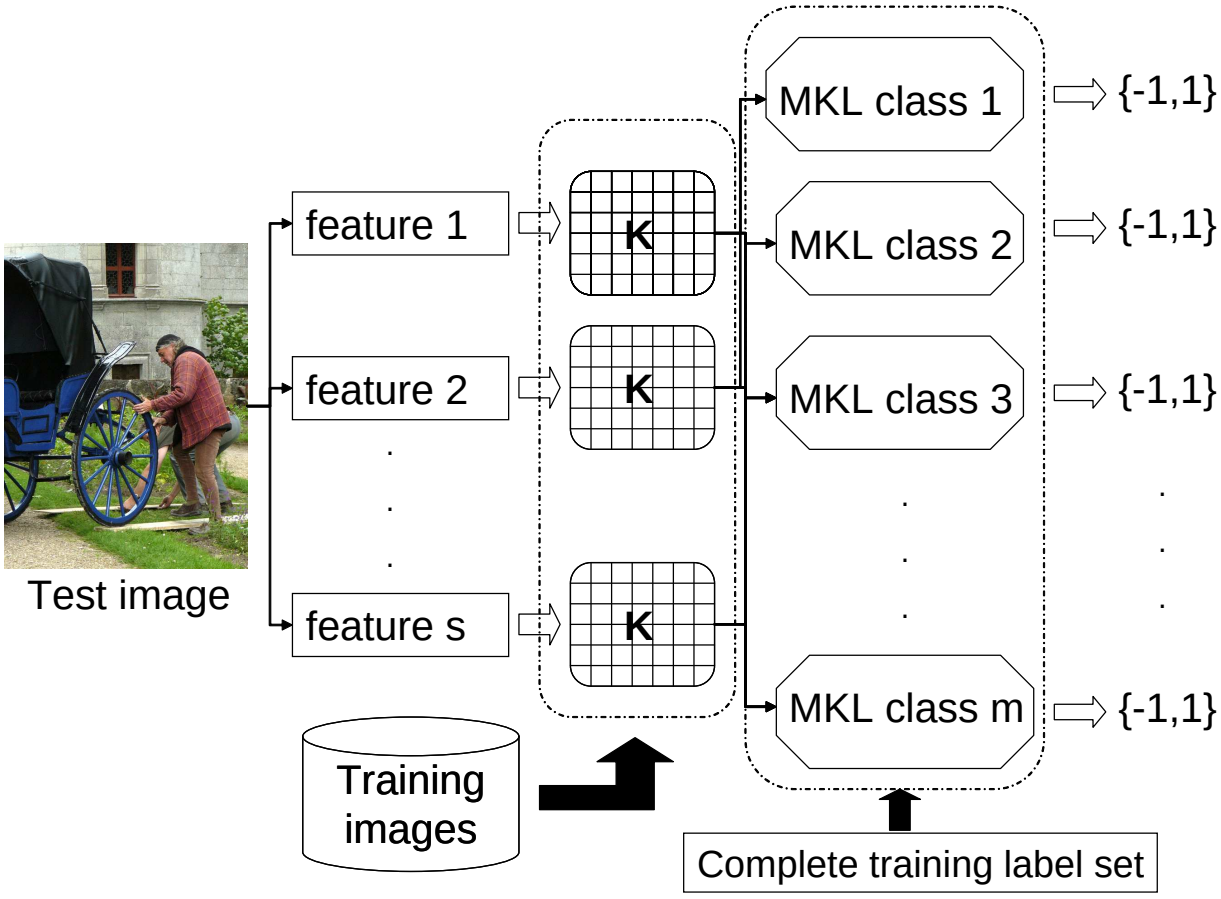

Figure 1.3: In Chapter 2, we discuss binary MKL methods for the one-vs-all framework, where an individual MKL algorithm is trained for each class.

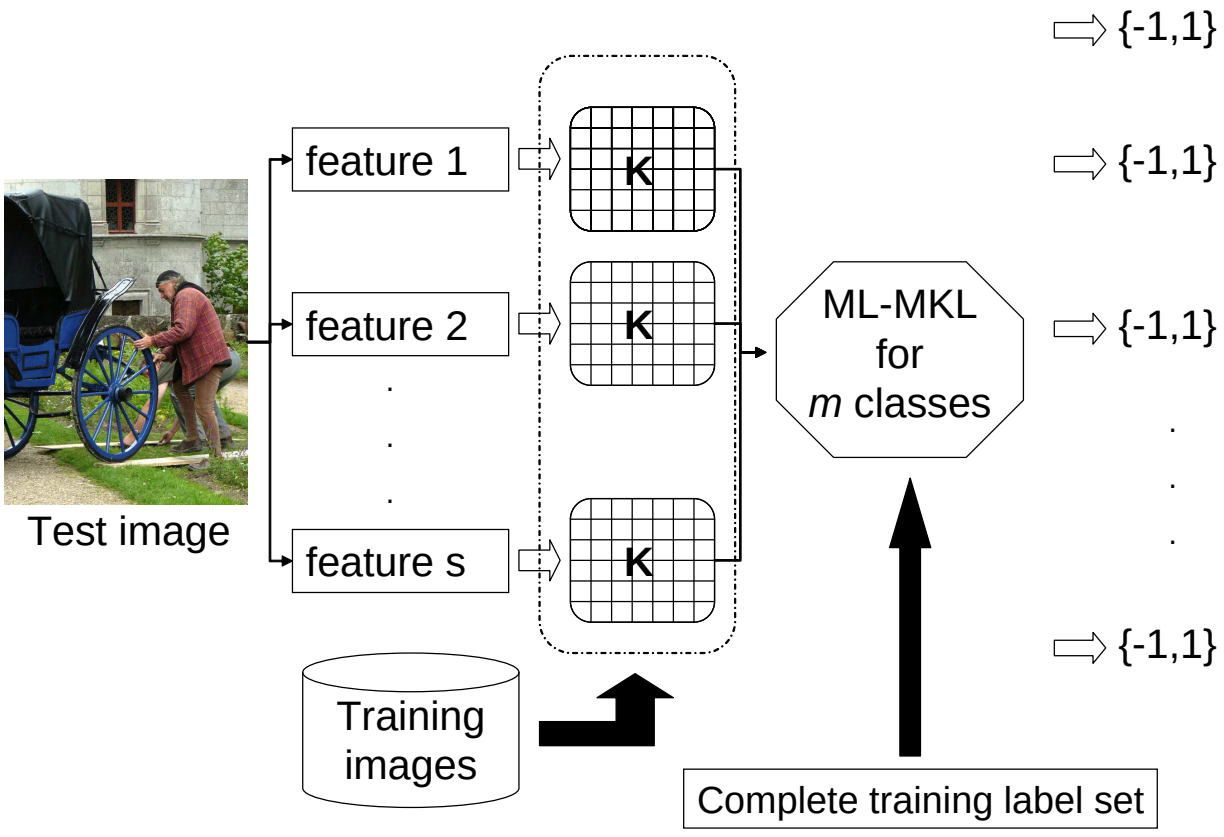

Figure 1.4: In Chapter 3, we present our multi-label MKL algorithm, which solves one MKL problem for all classes.

- Our contributions in Chapters 4 and 5 are efficient multi-label ranking algorithms. Given a test image, a multi-label ranking method aims to order all the object classes such that the relevant classes are ranked higher than the irrelevant classes (Figure 1.5). We present two efficient algorithms for multi-label ranking based on the idea of block coordinate descent. The proposed methods are computationally efficient; their computational complexity is linear in the number of classes, while the majority of the multi-label ranking schemes suffer from quadratic dependence on the number of classes. Our experimental results show that the proposed methods outperform state-of-the-art classification methods. Table 1.1 gives a comparison between the proposed multi-label ranking methods (MLR-L_1 and MLR-GL), and two state-of-the-art approaches on two benchmark data sets, ESP Game and MIR Flickr25000, in terms of AUC-ROC score. We use dense-SIFT features to generate the results in Table 1.1; however, the proposed methods consistently outperform the baselines even when different features are used.

- In Chapter 5 we present a robust multi-label learning method that performs well under the setting of limited annotations. Specifically, we consider a situation where the training example class assignments are incomplete, see Figure 1.6. Consider a training image whose true class assignment is (c_1, c_2, c_3, c_4), but is only assigned to classes c_1 and c_4. We refer to this problem as multi-label learning with incomplete class assignments, which has not been addressed in the multi-label learning literature. Incompletely labeled data is frequently encountered when the number of classes is very large (hundreds as in the MIR Flickr data set) or when there is a large ambiguity between classes (e.g., labels *jet* and *plane*). In both cases, it is difficult for users to provide complete class assignments for objects.

- We propose a ranking based multi-label learning framework that explicitly addresses the challenge of learning from incompletely labeled data by exploiting the group lasso technique to combine the ranking errors. Table 1.2 reports the results on two benchmarks data sets,

Table 1.1: Multi-label ranking performance (AUC-ROC) for the ESP Game and MIR Flickr25000 data sets

	ESP Game	MIRFlickr25000
SVM	79.5	70.2
MLLS	79.4	75.9
MLR-L_1	**81.5**	75.4
MLR-GL	80.5	**76.2**

ESP Game and MIR Flickr25000, in terms of AUC-ROC score, in two scenarios: (i) the complete label information is provided, (ii) 60% of the training labels are randomly removed. With performance in Table 1.2 and the experimental results in Chapter 5, we claim that the proposed method, MLR-GL outperforms the state-of-the-art multi-label classification methods on incompletely labeled data, including our other multi-label ranking approach MLR-L_1.

- Finally, we propose a multiple kernel multi-label ranking method (MK-MLR) by combining the strengths of the algorithms in Chapters 2, 3, and 4. We extend the proposed MLR-L_1 method to multiple kernel setting by integrating it into the SILP (semi-infinite linear programming) based wrapper MKL solver, which is the most efficient MKL-L_1 optimization method according to our detailed analysis in Chapter 2. We also use the idea of learning a shared kernel combination for all image classes to improve the computational efficiency. The MK-MLR method addresses the two essential factors for improving the performance of image categorization: (i) heterogeneous information fusion, and (ii) exploiting label correlation of multi-label data.

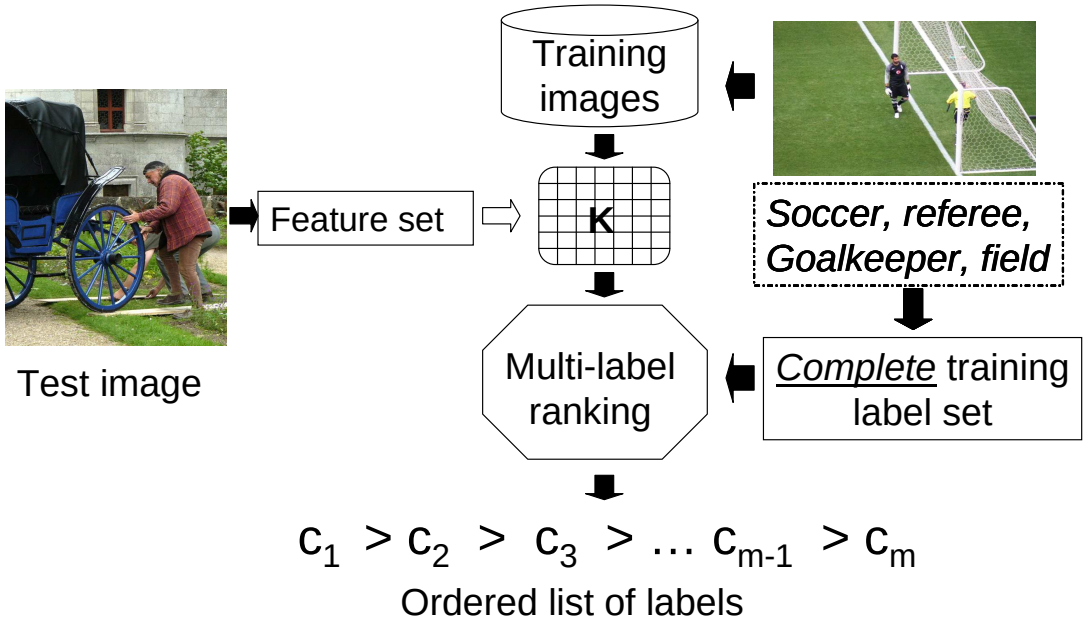

Figure 1.5: The difference between the two proposed multi-label ranking approaches MLR-L_1 (Chapter 4) and MLR-GL (Chapter 3) is that MKL-L_1 strictly addresses the complete class assignment problem whereas MLR-GL can handle missing class assignments. For example, the complete and full annotations are provided with all four labels (*soccer, referee, field*, *goalkeeper*) for the given image.

Table 1.2: AUC-ROC (%) scores for the ESP Game and MIR Flickr25000 data sets for the missing label scenario.

	ESP Game		MIR Flickr25000	
	complete	**60% missing**	**complete**	**60% missing**
SVM	80.2	75.2	70.2	65.7
MLLS	79.8	75.0	75.9	71.5
MLR-L_1	82.9	79.4	75.4	69.1
MLR-GL	**83.8**	**82.1**	**76.2**	**74.1**

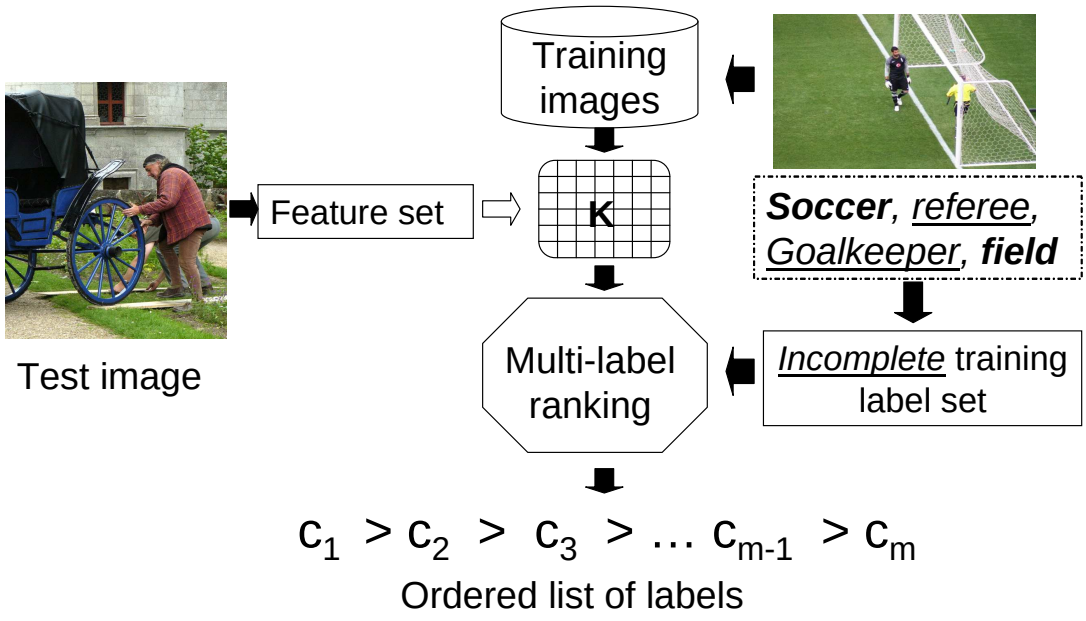

Figure 1.6: The difference between the two proposed multi-label ranking approaches (a) MLR-L_1 (Chapter 4) and (b) MLR-GL (Chapter 3) is that MKL-L_1 strictly addresses the complete class assignment problem whereas MLR-GL can handle missing class assignments. For example, only two labels (*soccer* and *field*, written with bold characters) are given for the above image whereas two labels (*goalkeeper* and *referee*, underlined text) are missing.

1.5 Notation

Let $\mathcal{D} = \{\mathbf{x^1}, \ldots, \mathbf{x^n}\}$ be a collection of n training instances, where $\mathcal{X} \subseteq \mathbb{R}^d$ is a compact domain. Each training example $\mathbf{x^i}$ is annotated by a set of class labels from $\mathcal{L}$, denoted by a binary vector $\mathbf{y^i} = (y_1^i, \ldots, y_m^i) \in \{-1, 1\}^m$, where m is the total number of classes, and $y_k^i = 1$ when $\mathbf{x^i}$ is assigned to class c_k and -1, otherwise. In multi-label ranking, we aim to learn m classification functions $f_k(\mathbf{x}) : \mathbb{R}^d \mapsto \mathbb{R}, k = 1, \ldots, m$, one for each class.

We denote by $\{\kappa_j(\mathbf{x}, \mathbf{x}') : \mathcal{X} \times \mathcal{X} \mapsto \mathbb{R},\ j = 1, \ldots, s\}$ a set of s base kernels to be combined in multiple kernel learning (MKL). For each kernel function $\kappa_j(\cdot, \cdot)$, we construct a kernel matrix $\mathbf{K}_j = [\kappa_j(\mathbf{x}, \mathbf{x}')]_{n \times n}$ by applying $\kappa_j(\cdot, \cdot)$ to the training instances in $\mathcal{D}$. We denote by $\boldsymbol{\beta} = (\beta_1, \ldots, \beta_s)^\top \in \mathbb{R}_+^s$ the set of coefficients used to combine the base kernels, and denote by $\kappa(\mathbf{x}, \mathbf{x}'; \boldsymbol{\beta}) = \sum_{j=1}^s \beta_j \kappa_j(\mathbf{x}, \mathbf{x}')$ and $\mathbf{K}(\boldsymbol{\beta}) = \sum_{j=1}^s \beta_j \mathbf{K}_j$ the combined kernel function and kernel matrix, respectively. We further denote by $\mathcal{H}_\beta$ the Reproducing Kernel Hilbert Space (RKHS) endowed with the combined kernel $\kappa(\mathbf{x}, \mathbf{x}'; \boldsymbol{\beta})$. The list of symbols and descriptions are given in Table 1.3.

The vectors and matrices are denoted by bold lowercase and uppercase characters, respectively. We use superscript to indicate the training instance index and subscript to show the class index for the feature and label vectors. For example, $\mathbf{y^i} \in \mathbb{R}^m$, with m being the number of labels, denotes the label vector for multi-labeled the training instance $\mathbf{x^i}$. On the other hand, $\mathbf{y_k} \in \mathbb{R}^n$, where n is the number of training instances, is the label assignment vector on all training instances for class c_k. We use a scalar $y_k^i \in \{-1, +1\}$ to indicate the label assignment of instance i for class c_k. For binary classification tasks, for example Chapter 2, we drop the subscript, i.e., $y^i \in \{-1, +1\}$, for simplicity. For a matrix $\mathbf{K}$, $\mathbf{K}_{:,i}$ and $\mathbf{K}_{j,:}$ denote the ith column and and jth row vectors, respectively. For the multiple kernel learning section, $\mathbf{K}_j$ indicates the jth base kernel.

Table 1.3: The list of symbols used in this dissertation

Definition	**Symbol**
Instance space	$\mathcal{X} \in \mathbb{R}^d$
Label set	$\mathcal{L}$
Number of dimensions	d
Number of instances	n
Number of class labels	m
Number of base kernels for MKL	s
Kernel function	$\kappa(.,.)$
$\mathbf{1_k}$	k dimensional vector of all ones
$\mathbf{0_k}$	k dimensional vector of all zeros
$\mathbf{M}_{:,i}$	ith column vector of the matrix $\mathbf{M}$
Classification function for class k	$f_k(x) : \mathbb{R}^d \mapsto \mathbb{R}$
Reproducing Kernel Hilbert Space (RKHS) endowed with the combined kernel	$\mathcal{H}_\beta$
Kernel coefficients for MKL	$\boldsymbol{\beta} = (\beta_1, \ldots, \beta_s)^\top \in \mathbb{R}^s_+$
Training instance	$\mathbf{x^i} = (x^i_1, x^i_2, \ldots, x^i_d) \in \mathbb{X}$
Label vector	$\mathbf{y^i} = (y^i_1, \ldots, y^i_m) \in \{-1, 1\}^m$

Chapter 2

Multiple Kernel Learning for Image Categorization: A Review

2.1 Introduction

Kernel methods [32] have become popular in computer vision, particularly for image categorization. The key idea of kernel methods is to introduce nonlinearity into the decision function by mapping the original features to a higher dimensional space. Many studies [4, 33, 34] have shown that nonlinear kernels, such as radial basis functions (RBF) or chi-squared kernels, yield significantly higher accuracy for image categorization than a linear classification model.

One difficulty in developing kernel classifiers is to design an appropriate kernel function for a given task. We often have multiple kernel candidates for image categorization. These kernels arise either because multiple feature representations are derived for images, or because different kernel functions (e.g., polynomial, RBF, and chi-squared) are used to measure the visual similarity between two images for a given feature representation. One of the key challenges in image categorization is to find the optimal combination of these kernels for a given object class. This is the central question addressed by Multiple Kernel Learning (MKL).

Table 2.1: Comparison of MKL baselines and simple baselines ("Single" for single best performing kernel and "AVG" for the average of all the base kernels) in terms of classification accuracy. The last three columns give the references in which either "method1" or "method2" performs better, or both methods give comparable results, respectively.

meth1	**meth2**	**dataset**	**# samples**	**# kernels**	**mtd1**	**mtd2**	**comp.**
MKL	Single	UCI	[1-6K]	[1-10]	[35]		[36]
MKL	Single	UCI	[1-2K]	[10-200]	[37]		
L_1-MKL	AVG	Cal-101	[510-3K]	[10-1K]	[38], [9]	[39], [40]	[41]
L_1-MKL	AVG	VOC07	5011	[10-22]	[9],	[41]	[42]
L_1-MKL	AVG	Oxford Flowers	680	[5-65]			[43]
L_p-MKL	AVG	VOC07	5011	10	[42]		
L_p-MKL	AVG	Cal-101	[1K-3K]	[24-1K]	[41]		[40]
L_p-MKL	AVG	Oxford Flowers	680	[5,65]			[41]
L_1-MKL	L_p-MKL	UCI	[1-2K]	[1-50]	[44]	[45], [46]	[47]
L_1-MKL	L_p-MKL	VOC07	5011	[10-22]		[42], [41]	
L_1-MKL	L_p-MKL	Cal-101	[510-3K]	[10-1K]		[40]	[41]

A lack of comprehensive studies has resulted in different, sometimes conflicting, statements regarding the effectiveness of various MKL methods on real-world problems, particularly for image categorization. For instance, some of the studies [5, 9, 41, 46] reported that MKL outperforms the average kernel baseline while other studies made the opposite conclusion [40,48,49], see Table 2.1. Moreover, as Table 2.2 shows, there are also some confusing results and statements about the efficiency of different MKL methods. Besides summarizing the latest developments in MKL and its application to image categorization, an important contribution of this chapter is to resolve the conflicting statements by conducting a comprehensive evaluation of state-of-the-art MKL algorithms under various experimental conditions.

The main contributions of the survey we give in this chapter are:

- A review of a wide range of MKL formulations that use different regularization mechanisms, and the related optimization algorithms.

- A comprehensive study that evaluates and compares a representative set of MKL algorithms

Table 2.2: Comparison of computational efficiency of MKL methods. The last three columns give the references, where "method1" is better, "method2" is better, or both give similar results.

meth1	**meth2**	**datasets**	**# samples**	**# kernels**	**mtd1**	**mtd2**	**cmp.**
training time							
L_1-MKL	L_p-MKL	MedMill	30,993	3			[50]
MKL-L1	L_p-MKL	UCI	[1-2K]	[90-800]	[48]		
MKL-SD	MKL-SIP	UCI	[1-2K]	[50-200]	[51], [52]		
MKL-SD	MKL-SIP	UCI	[1-2K]	[50-200]	[53], [46]		
MKL-SD	MKL-SIP	Oxford Flowers	680	[5-65]		[43]	
MKL-SD	MKL-MD	Oxford Flowers	680	[5-65]		[39]	
MKL-SD	MKL-MD	Cal-101	3,060	9	[9]		
MKL-SD	MKL-MD	VOC07	5,011	22		[9]	
MKL-SD	MKL-Lev	UCI	[1-2K]	[50-200]	[52]		
MKL-SIP	MKL-Lev	UCI	[1-2K]	[50-200]		[52]	
# active kernels							
MKL-SD	MKL-SIP	UCI	[1-2K]	[50-200]		[51]	
MKL-SD	MKL-SIP	UCI	[1-2K]	[50-200]		[53]	
MKL-SD	MKL-Lev	UCI	[1-2K]	[50-200]		[52]	
MKL-SIP	MKL-Lev	UCI	[1-2K]	[50-200]	[52]		

for image categorization under different experimental settings.

- An exposition of the conflicting statements regarding the performance of different MKL methods, particularly for image categorization. We attempt to understand these statements and determine to what degree and under what conditions these statements are correct.

2.2 Overview

In this section we give an overview of multiple kernel learning.

2.2.1 Overview of Multiple Kernel Learning (MKL)

MKL was first proposed in [54], where it was cast into a Semi-Definite Programming (SDP) problem. Most studies on MKL are centered around two issues, (i) how to improve the classification accuracy of MKL by exploring different formulations, and (ii) how to improve the learning efficiency of MKL by exploiting different optimization techniques (see Figure 2.1).

In order to learn an appropriate kernel combination, various regularizers have been introduced for MKL, including L_1 norm [55], L_p norm ($p > 1$) [56], entropy based [48], and mixed norms [57]. Among them, L_1 norm is probably the most popular choice because it results in sparse solutions and could potentially eliminate irrelevant and noisy kernels. In addition, theoretical studies [58, 59] have shown that L_1 norm will result in a small generalization error even when the number of kernels is very large.

A number of empirical studies have compared the effect of different regularizers used for MKL [41,46,60]. Unfortunately, different studies arrive at contradictory conclusions. For instance, while many studies claim that L_1 regularization yields good performance for object recognition [40, 61], others show that L_1 regularization results in information loss by imposing sparseness over MKL solutions, thus leading to suboptimal performance [41, 46, 48, 60, 62].

In addition to a linear combination of base kernels, several algorithms have been proposed to find a nonlinear combination of base kernels [39, 45, 63–65]. Some of these algorithms try to find a polynomial combination of the base kernels [45, 63], while others aim to learn an instance-dependent linear combination of kernels [5, 66, 67]. The main shortcoming of these approaches is that they have to deal with non-convex optimization problems, leading to poor computational efficiency and suboptimal performance. Given these shortcomings, we will not review them in detail.

Despite significant efforts in improving the effectiveness of MKL, one of the critical questions remaining is whether MKL is more effective than the popular simple baselines, e.g., taking the average of the base kernels. While many studies show that MKL algorithms bring significant

improvement over the average kernel approach [46, 62, 68], opposite conclusions have been drawn by some other studies [40, 41, 48, 49]. Our empirical studies show that these conflicting statements are largely due to the variations in the experimental conditions, or in other words, the consequence of a lack of comprehensive studies on MKL.

The second line of research in MKL is to improve the learning efficiency. Many efficient MKL algorithms [46, 48, 53, 55, 64, 69, 70] have been proposed, mostly for L_1 regularized MKL, based on the first order optimization methods. We again observe conflicting statements in the MKL literature when comparing different optimization algorithms. For instance, while some studies [46, 51, 52] report that the subgradient descent (SD) algorithms [53] are more efficient in training MKL than the semi-infinite linear programming (SILP) based algorithm [71], an opposing statement was given in [61]. It is important to note that besides the training time, the sparseness of the solution also plays an important role in computational efficiency: both the number of active kernels and the number of support vectors affect the number of kernel evaluations and, consequentially, computational times for both training and testing. Unfortunately, most studies focus on only one aspect of computational efficiency: some only report the total training time [48, 61] while others focus on the number of support vectors (support set size) [46, 67]. Another limitation of the previous studies is that they are mostly constrained to small data sets (around 1,000 samples) and limited number of base kernels (10 to 50), making it difficult to draw meaningful conclusions on the computational efficiency.

2.2.2 Relationship to the Other Approaches

Multiple kernel learning is closely related to feature selection [72], where the goal is to identify a subset of features that are optimal for a given prediction task. This is evidenced by the equivalence between MKL and group lasso [73], a feature selection method where features are organized into groups, and the selection is conducted at the group level instead of at the level of individual features.

Feature selection and feature combination can be given among the main motivations of multiple kernel learning, particularly for the image categorization task. There is a vast amount of choices of image representations. Feature selection is related to choosing the correct image representation for the given classification task. In this manner, MKL is closely related to feature selection. However, selecting one type of representation might not be adequate, since image categorization often involves many classification tasks, one for each image class, and one representation that would work for some of the classes might not work for others. One way to tackle this problem is combining several features. The early approaches for feature combination includes unweighted combination of features [34] or employing brute force learning of feature combination parameters [74]. However, the goal of MKL is to find a more principled way of performing feature combination. It is important to note that equivalence between MKL and group lasso has been proven in [73] building a formal connection between MKL and feature selection.

MKL is also related to metric learning [75], where the goal is to find a distance metric, or more generally a distance function, consistent with the class assignment. MKL generalizes metric learning by searching for a combination of kernel functions that gives a larger similarity to any instance pair from the same class than instance pairs from different classes.

Finally, it is important to note that multiple kernel learning is a special case of kernel learning. In addition to MKL, another popular approach for learning a linear combination of multiple kernels is kernel alignment [76], which finds the optimal combination of kernels by maximizing the alignment between the combined kernel and the class assignments matrix. More generally, kernel learning methods can be classified into two groups: parametric and non-parametric kernel learning. In parametric kernel learning, a parametric form is assumed for the combined kernel function [77, 78]. In contrast, nonparametric kernel learning does not make any parametric assumption about the target kernel function [76, 79, 80]. Multiple kernel learning belongs to the category of parametric kernel learning. Despite its generality, the high computational cost of nonparametric kernel learning limits its applications to real-world problems. Aside from supervised

kernel learning, both semi-supervised and unsupervised kernel learning have also been investigated [76, 78, 81]. We do not review them in detail here because of their limited success in practice and because of their high computational cost.

2.3 Multiple Kernel Learning (MKL): Formulations

In this section, we first review the theory of multiple kernel learning for binary classification. We leave the discussion of the MKL methods for multi-class and multi-label learning to Chapter 3.

Let $\mathcal{D} = \{\mathbf{x^1}, \ldots, \mathbf{x^n}\}$ be a collection of n training instances, where $\mathcal{X} \subseteq \mathbb{R}^d$ is a compact domain. Let $\mathbf{y} = (y^1, \ldots, y^n)^\top \in \{-1, +1\}^n$ be the vector of class assignments for the instances in $\mathcal{D}$. We denote by $\{\kappa_j(\mathbf{x}, \mathbf{x}') : \mathcal{X} \times \mathcal{X} \mapsto \mathbb{R},\ j = 1, \ldots, s\}$ the set of s base kernels to be combined. For each kernel function $\kappa_j(\cdot, \cdot)$, we construct a kernel matrix $\mathbf{K}_j = [\kappa_j(\mathbf{x}, \mathbf{x}')]_{n \times n}$ by applying $\kappa_j(\cdot, \cdot)$ to the training instances in $\mathcal{D}$. We denote by $\boldsymbol{\beta} = (\beta_1, \ldots, \beta_s)^\top \in \mathbb{R}^s_+$ the set of coefficients used to combine the base kernels, and denote by $\kappa(\mathbf{x}, \mathbf{x}'; \boldsymbol{\beta}) = \sum_{j=1}^s \beta_j \kappa_j(\mathbf{x}, \mathbf{x}')$ and $\mathbf{K}(\boldsymbol{\beta}) = \sum_{j=1}^s \beta_j \mathbf{K}_j$ the combined kernel function and kernel matrix, respectively. We further denote by $\mathcal{H}_\beta$ the Reproducing Kernel Hilbert Space (RKHS) endowed with the combined kernel $\kappa(\mathbf{x}, \mathbf{x}'; \boldsymbol{\beta})$. In order to learn the optimal combination of kernels, we first define the regularized classification error $\mathcal{L}(\boldsymbol{\beta})$ for a combined kernel $\kappa(\cdot, \cdot; \boldsymbol{\beta})$, i.e.,

$$\mathcal{L}(\boldsymbol{\beta}) = \min_{f \in \mathcal{H}_\beta} \frac{1}{2} ||f||^2_{\mathcal{H}_\beta} + C \sum_{i=1}^n \ell(y^i f(\mathbf{x}_i)), \tag{2.1}$$

where $\ell(z) = \max(0, 1 - z)$ is the hinge loss and $C > 0$ is a regularization parameter. Given the regularized classification error, the optimal combination vector $\boldsymbol{\beta}$ is found by minimizing $\mathcal{L}(\boldsymbol{\beta})$, i.e.,

$$\min_{\boldsymbol{\beta} \in \Delta, f \in \mathcal{H}_\beta} \frac{1}{2} ||f||^2_{\mathcal{H}_\beta} + C \sum_{i=1}^n \ell(y^i f(\mathbf{x}_i)) \tag{2.2}$$

where Δ is a convex domain for combination weights $\boldsymbol{\beta}$ that will be discussed later. As in [54],

the problem in Eq. (2.2) can be written into its dual form, leading to the following convex-concave optimization problem

$$\min_{\boldsymbol{\beta}\in\Delta}\max_{\alpha\in\mathcal{Q}}\widehat{\mathcal{L}}(\boldsymbol{\alpha},\boldsymbol{\beta})=\mathbf{1}^{\top}\boldsymbol{\alpha}-\frac{1}{2}(\boldsymbol{\alpha}\circ\mathbf{y})^{\top}\mathbf{K}(\boldsymbol{\beta})(\boldsymbol{\alpha}\circ\mathbf{y}), \tag{2.3}$$

where $\circ$ denotes the Hadamard (element-wise) product, $\mathbf{1}$ is a vector of all ones, and $\mathcal{Q}=\{\boldsymbol{\alpha}\in[0,C]^n\}$ is the domain for dual variables $\boldsymbol{\alpha}$.

The choice of domain Δ for kernel coefficients can have a significant impact on both classification accuracy and efficiency of MKL. One common practice is to restrict $\boldsymbol{\beta}$ to a probability distribution, leading to the following definition of domain Δ [54, 55],

$$\Delta_1=\left\{\boldsymbol{\beta}\in\mathbb{R}_+^s:\|\boldsymbol{\beta}\|_1=\sum_{j=1}^{s}|\beta_j|\leq 1\right\}. \tag{2.4}$$

Since Δ_1 bounds $\|\boldsymbol{\beta}\|_1$, we also refer to MKL using Δ_1 as the L_1 regularized MKL, or L_1-MKL. The key advantage of using Δ_1 is that it results in a sparse solution for $\boldsymbol{\beta}$, leading to the elimination of irrelevant kernels and consequentially an improvement in computational efficiency as well as robustness in classification.

2.3.1 Multiple Kernel Learning and Group Lasso

Lasso (least absolute shrinkage and selection operator), regression with L_1 regularization, is a popular technique that performs feature selection and shrinkage [82]. Shrinkage in this context means producing sparse solutions, since the L_1-norm regularization forces some of the covariates to shrink to zero. An extension of the lasso technique, in which the L_1-norm is replaced by a block L_1-norm, is called the group lasso. In group lasso the covariates are assumed to be clustered and the absolute values of each group's Euclidean norm are added when constructing the regularizer term. Therefore, the shrinkage is forced at the group level, meaning that all covariates within a

group are forced to be zero altogether.

Let each training instance $\mathbf{x^i} \in \mathbb{R}^d$ have a block structure with m blocks, such that $\mathbf{x^i} = (\mathbf{x^{i1}}, \mathbf{x^{i2}}, \ldots, \mathbf{x^{im}})$, where $\mathbf{x^{ik}} \in \mathbb{R}^{d_k}$, $k = 1, 2, \ldots, m$ and $\sum_{k=1}^{m} d_k = d$. The group lasso can be formulated as the optimization problem in Eq. (2.5),

$$\min_{\mathbf{w}\in\mathbb{R}^d, b\in\mathbb{R}} C\sum_{i=1}^{n} \ell((\mathbf{x^i}, y^i); \mathbf{w}) + \sum_{k=1}^{m} \lambda_k ||\mathbf{w^k}||, \tag{2.5}$$

where $\mathbf{w}$ is a linear classifier, b is a bias term, C is a constant, and λ_k, $k = 1, \ldots, m$ are positive weights. Square of the block L_1-norm, $(\sum_{k=1}^{m} \lambda_k ||\mathbf{w^k}||)^2$, can also be used as an alternative group lasso regularizer and would give the same path of solutions [35, 73].

The group lasso formulation with the squared block L_1-norm, can be extended to nonlinear case by using functions and reproducing kernel Hilbert norms instead of linear predictors and Euclidean norms as expressed in Eq.(2.6),

$$\min_{\{f_k\}_{k=1}^{m}\in} \frac{1}{2}(\sum_{k=1}^{m} ||f_k||_{\mathcal{H}_k})^2 + C\sum_{i=1}^{n} \ell(y^i \sum_{k=1}^{m} f_k(\mathbf{x^i})), \tag{2.6}$$

where $\mathcal{H}_k$ is the k-th Reproducing Kernel Hilbert Space (RKHS). Note that this formulation, which learns a sparse combination of functions, enables using an infinite dimensional space for each group. By following [46, 73], it is possible to show that this formulation is equivalent to learning a convex combination of kernel functions, each corresponding to one group and endows the corresponding RKHS. To prove this connection, we will use an alternative MKL formulation that is given by Eq. (2.7).

$$\min_{\lambda\in\mathbb{R}_+^m, \sum_k \lambda_k = 1} \min_{\{f_k\in\mathcal{H}_k\}_{k=1}^{m}} \frac{1}{2}\sum_{k=1}^{m} \lambda_k ||f_k||_{\mathcal{H}_k}^2 + C\sum_{i=1}^{n} \ell(\sum_{k=1}^{m} y^i \lambda_k f_k(\mathbf{x^i})). \tag{2.7}$$

We provide the proof of equivalence between Eqs. (2.2) and (2.7) in the Appendix.

Replacing $\lambda_k f_k$ with $\tilde{f}_k$, we rewrite Eq. (2.7) as Eq. (2.8).

$$\min_{\lambda \in \mathbb{R}_+^m, \sum_k \lambda_k = 1} \min_{\{\tilde{f}_k \in \mathcal{H}_k\}_{k=1}^m} \frac{1}{2} \sum_{k=1}^{m} \frac{1}{\lambda_k} ||\tilde{f}_k||_{\mathcal{H}_k}^2 + C \sum_{i=1}^{n} \ell(\sum_{k=1}^{m} y^i \tilde{f}_k(\mathbf{x^i})). \tag{2.8}$$

It is straightforward to show that the expression in Eq. (2.9) is the minimizer of Eq. (2.8),

$$\lambda_k = \frac{||\tilde{f}_k||_{\mathcal{H}_k}}{\sum_{k=1}^{m} ||\tilde{f}_k||_{\mathcal{H}_k}}. \tag{2.9}$$

Substituting the expression in Eq. (2.9) into Eq. (2.8) leads to the following optimization problem,

$$\min_{\{f_k\}_{k=1}^m \in} \frac{1}{2} (\sum_{k=1}^{m} ||f_k||_{\mathcal{H}_k})^2 + C \sum_{i=1}^{n} \ell(y^i \sum_{j=1}^{m} f_k(\mathbf{x^i})), \tag{2.10}$$

which is the same as Eq. (2.10), proving the equivalance between MKL and group lasso.

2.3.2 Regularization in MKL

The robustness of L_1-MKL is verified by the analysis in [58], which states that the additional generalization error caused by combining multiple kernels is $O(\sqrt{\log s/n})$ when using Δ_1 as the domain for $\boldsymbol{\beta}$, implying that L_1-MKL is robust to the number of kernels as long as the number of training examples is not too small. The advantage of L_1-MKL is further supported by the equivalence between L_1-MKL and feature selection using group Lasso [73]. Since group Lasso is proved to be effective in identifying the groups of irrelevant features, L_1-MKL is expected to be resilient to weak kernels.

Despite the advantages of L_1-MKL, it was reported in [50] that sparse solutions generated by L_1-MKL might result in information loss and consequentially suboptimal performance. As a result, L_p regularized MKL (L_p-MKL), with $p > 1$, was proposed in [56, 61] in order to obtain a

smooth kernel combination, with the following definition for domain Δ

$$\Delta_p = \left\{ \boldsymbol{\beta} \in \mathbb{R}_+^s : ||\boldsymbol{\beta}||_p \leq 1 \right\}. \tag{2.11}$$

Among various choices of L_p-MKL ($p > 1$), L_2-MKL is probably the most popular one [49,50,56]. Other smooth regularizers proposed for MKL include negative entropy (i.e., $\sum_{j=1}^{s} \beta_j \log \beta_j$) [48] and Bregman divergence [70]. In addition, hybrid approaches have been proposed to combine different regularizers for MKL [49, 83, 84].

Although many studies compared L_1 regularization to smooth regularizers for MKL, the results are inconsistent. While some studies claimed that L_1 regularization yields better performance for image categorization [40, 61], others show that L_1 regularization may result in suboptimal performance due to the sparseness of the solutions [41, 46, 48, 60, 62]. In addition, some studies reported that training an L_1-MKL is significantly more efficient than training a L_2-MKL [48], while others claimed that the training times for both MKL techniques are comparable [50].

A resolution to these contradictions, as revealed by our empirical study, depends on the number of training examples and the number of kernels. In terms of classification accuracy, smooth regularizers are more effective for MKL when the number of training examples is small. Given a sufficiently large number of training examples, particularly when the number of base kernels is large, L_1 regularization is likely to outperform the smooth regularizers.

In terms of computation time, we found that L_p-MKL methods are generally more efficient than L_1-MKL. This is because the objective function of L_p-MKL is smooth while the objective function of L_1-MKL is not [1]. As a result, L_p-MKL enjoys a significantly faster convergence rate $(O(1/T^2))$ than L_1-MKL $(O(1/T))$ according to [85], where T is the number of iterations. However, when the number of kernels is sufficiently large and kernel combination becomes the dominant computational cost at each iteration, L_1-MKL can be as efficient as L_p-MKL because

[1] A function is smooth if its gradient is Lipschitz continuous

L_1-MKL produces sparse solutions.

One critical question that remains to be answered is whether MKL is more effective than simple approaches for kernel combination, e.g., using the best single kernel (selected by cross validation) or the average kernel method. Most studies show that L_1-MKL outperforms the best performing kernel, although there are scenarios where kernel combination might not perform as well as the single best performing kernel [50]. Regarding the comparison of MKL to the average kernel baseline, the answer is far from conclusive (see Table 2.2). While some studies show that L_1-MKL brings significant improvement over the average kernel approach [46, 62, 68, 86], other studies claim the opposite [40, 41, 48, 49]. As revealed by the empirical study presented in Section 2.5, the answer to this question depends on the experimental setup. When the number of training examples is not sufficient to identify the strong kernels, MKL may not perform better than the average kernel approach. But, with a large number of base kernels and a sufficiently large number of training examples, MKL is very likely to outperform, or at least yield similar performance as, the average kernel technique.

2.4 Multiple Kernel Learning: Optimization Techniques

A large number of algorithms have been proposed to solve the optimization problems posed in Eqs. (2.2) and (2.3). We can broadly classify them into two categories. The first group of approaches directly solve the primal problem in Eq. (2.2) or the dual problem in Eq. (2.3). We refer to them as the *direct approaches*. The methods of the second group solve the convex-concave optimization problem in Eq. (2.3) by alternating between two steps, i.e., the step for updating the kernel combination weights and the step for solving the SVM classifier for the given combination weights. We refer to them as the *wrapper approaches*. Figure 2.1 summarizes different optimization methods developed for MKL. We note that due to the scalability issue, almost all MKL algorithms are based on first order methods (i.e., iteratively updating the solutions which use the gradient of the

objective function or the most violated constraint). We refer the readers to [52, 60, 87] for more discussion about the equivalence or similarities among different MKL algorithms.

2.4.1 Direct Approaches for MKL

Lanckriet et al. [54] showed that the problem in Eq. (2.2) can be cast into Semi-Definite Programming (SDP) problem, i.e.,

$$\begin{aligned} \min_{\mathbf{z}\in\mathbb{R}^n, \boldsymbol{\beta}\in\Delta, t\geq 0} \quad & t/2 + C\sum_{i=1}^{n}\max(0, 1-y^i z^i) \\ \text{s. t.} \quad & \begin{pmatrix} \mathbf{K}(\boldsymbol{\beta}) & \mathbf{z} \\ \mathbf{z}^\top & t \end{pmatrix} \succeq 0. \end{aligned} \tag{2.12}$$

Although general-purpose optimization tools such as SeDuMi [88] and Mosek [89] can be used to directly solve the optimization problem in Eq. (2.12), they are computationally expensive and are unable to handle more than a few hundred training examples.

Besides directly solving the primal problem, several algorithms have been developed to directly solve the dual problem in Eq. (2.3). Bach et al. [35] proposed to solve the dual problem using sequential minimal optimization (SMO) [90]. In [48], the authors applied the Nesterov's method to solve the optimization problem in Eq. (2.3). Although both approaches are significantly more efficient than the direct approaches that solve the primal problem of MKL, they are generally less efficient than the wrapper approaches [55].

2.4.1.1 A Sequential Minimum Optimization (SMO) based Approach for MKL

This approach is designed for L_p-MKL. Instead of constraining $\|\beta\|_p \leq 1$, Vishwanathan et al. proposed to solve a regularized version of MKL in [70], and converted it into the following optimization problem,

$$\max_{\boldsymbol{\alpha}\in Q} \mathbf{1}^\top \boldsymbol{\alpha} - \frac{1}{8\lambda}\left(\sum_{j=1}^{s}\left[(\boldsymbol{\alpha}\circ\mathbf{y})^\top \mathbf{K}_j(\boldsymbol{\alpha}\circ\mathbf{y})\right]^q\right)^{\frac{2}{q}}. \tag{2.13}$$

It can be shown that given $\boldsymbol{\alpha}$, the optimal solution for $\boldsymbol{\beta}$ is given by

$$\beta_j = \frac{\gamma_j}{2\lambda}\left(\sum_{k=1}^{s}\left((\boldsymbol{\alpha}\circ\mathbf{y})^\top \mathbf{K}_k(\boldsymbol{\alpha}\circ\mathbf{y})\right)^q\right)^{\frac{1}{q}-\frac{1}{p}} \tag{2.14}$$

where $\gamma_j = \left((\boldsymbol{\alpha}\circ\mathbf{y})^\top K_j(\boldsymbol{\alpha}\circ\mathbf{y})\right)^{\frac{q}{p}}$ and $q^{-1}+p^{-1}=1$. Since the objective given in Eq. (2.13) is differentiable, a Sequential Minimum Optimization (SMO) approach [70] can be used.

2.4.2 Wrapper Approaches for MKL

The main advantage of the wrapper approaches is that they are able to effectively exploit the off-the-shelf SVM solvers, making them, in general, significantly more efficient than the direct approaches. Below, we describe several representative wrapper approaches for MKL, including a semi-infinite programming (SIP) approach, a subgradient descent approach, an extended level method, an alternating optimization approach, and a sequential minimum optimization (SMO) based approach.

2.4.2.1 A Semi-infinite Programming Approach for MKL (MKL-SIP)

It was shown in [71] that the dual problem in Eq. (2.3) can be cast into the following SIP problem:

$$\begin{aligned} \min_{\theta\in\mathbb{R},\boldsymbol{\beta}\in\Delta} \quad & \theta \\ \text{s. t.} \quad & \sum_{j=1}^{s}\beta_j\{\alpha^\top\mathbf{1} - \frac{1}{2}(\boldsymbol{\alpha}\circ\mathbf{y})^\top\mathbf{K}_j(\boldsymbol{\alpha}\circ\mathbf{y})\} \geq \theta, \\ \forall\alpha\in\mathcal{Q} & \end{aligned} \tag{2.15}$$

When the domain Δ_1 is used for $\boldsymbol{\beta}$, the problem in Eq. (2.15) is reduced to a Semi-Infinite Linear Programming (SILP) problem. To solve Eq. (2.15), we first initialize the problem with a small number of linear constraints. Then the SIP problem in Eq. (2.15) is solved by alternating between two steps, i.e., (i) finding the optimal $\boldsymbol{\beta}$ and θ with fixed constraints, and (ii) finding the unsatisfied constraints with the largest violation under the fixed $\boldsymbol{\beta}$ and θ and adding them to the system. Note that in the second step, to find the most violated constraints, the following optimization problem, which is an SVM problem for the combined kernel $\kappa(\cdot,\cdot;\boldsymbol{\beta})$, needs to be solved:

$$\max_{\alpha\in\mathcal{Q}} \quad \sum_{j=1}^{s}\beta_j S_j(\alpha) = \alpha^\top \mathbf{1} - \frac{1}{2}(\alpha\circ\mathbf{y})^\top K(\boldsymbol{\beta})(\alpha\circ\mathbf{y}).$$

2.4.2.2 Subgradient Descent Approaches for MKL (MKL-SD & MKL-MD)

A popular wrapper approach for MKL is SimpleMKL [53], which solves the dual problem in Eq. (2.3) by a subgradient descent approach. The authors turn the convex concave optimization problem in Eq. (2.3) into a minimization problem $\min_{\boldsymbol{\beta}\in\Delta} J(\boldsymbol{\beta})$, where the objective $J(\boldsymbol{\beta})$ is defined as

$$J(\boldsymbol{\beta}) = \max_{\boldsymbol{\alpha}\in\mathcal{Q}} -\frac{1}{2}(\boldsymbol{\alpha}\circ\mathbf{y})^\top \mathbf{K}(\boldsymbol{\beta})(\boldsymbol{\alpha}\circ\mathbf{y}) + \mathbf{1}^\top\boldsymbol{\alpha}. \tag{2.16}$$

Since the partial gradient of $J(\boldsymbol{\beta})$ is given by $\partial_{\beta_j} J(\boldsymbol{\beta}) = 1 - \frac{1}{2}(\boldsymbol{\alpha}^*\circ\mathbf{y})^\top \mathbf{K}_j(\boldsymbol{\alpha}^*\circ\mathbf{y}), j = 1,\ldots,s$, where $\boldsymbol{\alpha}^*$ is an optimal solution to Eq. (2.16), following the subgradient descent algorithm, we update the solution $\boldsymbol{\beta}$ by

$$\boldsymbol{\beta} \leftarrow \pi_\Delta\left(\boldsymbol{\beta} - \eta\partial J(\boldsymbol{\beta})\right)$$

where $\eta > 0$ is the step size determined by a line search [53] and $\pi_\Delta(\boldsymbol{\beta})$ projects $\boldsymbol{\beta}$ into the domain Δ. Similar approaches were proposed in [62, 63].

A generalization of the subgradient descent method for MKL is a mirror descent method (MKL-MD) [39]. Given a proximity function $w(\boldsymbol{\beta}', \boldsymbol{\beta})$, the current solution $\boldsymbol{\beta}^t$ and the subgra-

dient $\partial J(\boldsymbol{\beta}^t)$, the new solution $\boldsymbol{\beta}^{t+1}$ is obtained by solving the following optimization problem

$$\boldsymbol{\beta}^{t+1} = \arg\min_{\boldsymbol{\beta}\in\Delta} \eta(\boldsymbol{\beta} - \boldsymbol{\beta}^t)^\top \partial J(\boldsymbol{\beta}^t) + w(\boldsymbol{\beta}^t, \boldsymbol{\beta}), \tag{2.17}$$

where $\eta > 0$ is the step size.

The main shortcoming of SimpleMKL arises from the high computational cost of line search. It was indicated in [46] that many iterations may be needed by the line search to determine the optimal step size. Since each iteration of the line search requires solving a kernel SVM, it becomes computationally expensive when the number of training examples is large. Another subtle issue of SimpleMKL, as pointed out in [53], is that it may not converge to the global optimum if the kernel SVMs in the intermediate steps are not solved with high precision.

2.4.2.3 An Extended Level Method for MKL (MKL-Level)

An extended level method is proposed for L_1-MKL in [52]. To solve the optimization problem in Eq. (2.3), at each iteration, the level method first constructs a cutting plane model $g^t(\boldsymbol{\beta})$ that provides a lower bound for the objective function $J(\boldsymbol{\beta})$. Given $\{\boldsymbol{\beta}^a\}_{a=1}^t$, the solutions obtained for the first t iterations, a cutting plane model is constructed as $g^t(\beta) = \max_{1\le a\le t} L(\boldsymbol{\beta}, \boldsymbol{\alpha}^a)$, where $\boldsymbol{\alpha}^a = \arg\max_{\boldsymbol{\alpha}\in Q} L(\boldsymbol{\beta}^a, \boldsymbol{\alpha})$. Given the cutting plane model, the level method then constructs a level set $\mathcal{S}_t$ as

$$\mathcal{S}_t = \{\boldsymbol{\beta} \in \Delta_1 : g^t(\boldsymbol{\beta}) \le l^t = \lambda \bar{L}^t + (1-\lambda)\underline{L}^t\}, \tag{2.18}$$

and obtain the new solution $\boldsymbol{\beta}^{t+1}$ by projecting $\boldsymbol{\beta}^t$ into $\mathcal{S}_t$, where $\bar{L}^t$ and $\underline{L}^t$, the upper and lower bounds for the optimal value $L(\boldsymbol{\beta}^*, \boldsymbol{\alpha}^*)$, are given by $\underline{L}^t = \min_{\boldsymbol{\beta}\in\Delta} g^t(\boldsymbol{\beta})$ and $\bar{L}^t = \min_{1\le a\le t} L(\boldsymbol{\beta}^a, \boldsymbol{\alpha}^a)$.

Compared to the subgradient-based approaches, the main advantage of the extended level method is that it is able to exploit all the gradients computed in the past for generating new solutions, leading to a faster convergence to the optimal solution.

2.4.2.4 An Alternating Optimization Method for MKL (MKL-GL)

This approach was proposed in [53, 56] for L_1-MKL. It is based on the equivalence between group Lasso and MKL, and solves the following optimization problem for MKL

$$\begin{array}{ll} \min & \frac{1}{2}\sum_{j=1}^{s}\frac{\|f_j\|_{\mathcal{H}_j}^2}{\beta_j} + C\sum_{i=1}^{n}\ell\left(y^i\sum_{j=1}^{s}f_j(\mathbf{x^i})\right) \\ \boldsymbol{\beta}\in\Delta_1 & \\ f_j\in\mathcal{H}_j & \end{array} \tag{2.19}$$

The solution requires alternating between two steps, i.e., the step of optimizing f_j under fixed $\boldsymbol{\beta}$ and the step of optimizing $\boldsymbol{\beta}$ given fixed f_j. The first step is equivalent to solving a kernel SVM with a combined kernel $\kappa(\cdot,\cdot;\boldsymbol{\beta})$, and the optimal solution in the second step is given by

$$\beta_j = \frac{||f_j||_{\mathcal{H}_j}}{\sum_{k=1}^{s}||f_k||_{\mathcal{H}_k}}, j = 1,\ldots,s. \tag{2.20}$$

It was shown in [46] that the above approach can be extended to L_p-MKL.

2.4.3 Online Learning Algorithms for MKL

Online learning is computationally efficient as it only needs to process one training example at each iteration. In [91], the authors proposed several online learning algorithms for MKL that combine the Perceptron algorithm [92] with the Hedge algorithm [93]. More specifically, the authors applied the Perceptron algorithm to update the classifiers for the base kernels and the Hedge algorithm to learn the combination weights. In [38], Jie et al. presented an online learning algorithm for MKL, based on the follow-the-regularized-leader (FTRL) framework. One disadvantage of online learning for MKL is that it usually yields suboptimal recognition performance compared to the batch learning algorithms. As a result, we did not include online MKL in our empirical study.

2.4.4 Computational Efficiency

In this section, we review the conflicting statements in MKL literature about the computational efficiency of different optimization algorithms for MKL. First, there is no consensus on the efficiency of the SIP based approach for MKL (MKL-SIP). While several studies show a slow convergence of MKL-SIP [52, 53, 68, 70], it was stated in [87] that only a few iterations would suffice when the number of relevant kernels is small. According to our empirical study, the SIP based approach can converge in a few iterations for L_p-MKL. On the other hand, MKL-SIP takes many more iterations to converge for L_1-MKL.

Second, several studies evaluated the training time of SimpleMKL in comparison to the other approaches for MKL, but with different conclusions. In [46] MKL-SIP was found to be significantly slower than SimpleMKL while the studies in [51, 52] reported the opposite.

The main reason behind the conflicting conclusions is that the size of test bed (i.e. the number of training examples and the number of base kernels) varies significantly from one study to another (Table 2.2). When the number of kernels and the number of training examples are large, calculation and combination of the base kernels take a significant amount of the computational load, while for small data sets, the computational efficiency is mostly decided by the iteration complexity of algorithms. In addition, implementation details, including the choice of stopping criteria and programming tricks for calculating the combined kernel matrix, can also affect the running time.

Our empirical study for image categorization showed that SimpleMKL is less efficient than MKL-SIP. Although SimpleMKL requires a smaller number of iterations, it takes significantly longer time to finish one iteration compared to the other approaches for MKL, due to the high computational cost of the line search. Overall, we observed that MKL-SIP is more efficient than the other wrapper optimization techniques for MKL whereas MKL-SMO is the fastest method for solving L_p-MKL.

2.5 Experiments

Our goal is to evaluate the classification performance of different MKL formulations and the efficiency of different optimization techniques for MKL. We focus on MKL algorithms for binary classification, and apply the one-vs-all strategy to convert a multi-label learning problem into a set of binary classification problems. Among various formulations for MKL, we only evaluate algorithms for L_1 and L_p regularized MKL. As stated earlier, we do not consider (i) online MKL algorithms due to their suboptimal performance and (ii) nonlinear MKL algorithms due to their high computational costs.

The first objective of this empirical study is to compare L_1-MKL algorithms to the two simple baselines of kernel combination mentioned in Section 2.2.1, i.e., the single best performing kernel and the average kernel approach. As already mentioned in Section 2.2.1, there are contradictory statements from different studies regarding the comparison of MKL algorithms to these two baselines. The goal of our empirical study is to examine and identify the factors that may contribute to the conflicting statements. The factors we consider here include (i) the number of training examples and (ii) the number of base kernels. The second objective of this study is to evaluate the classification performance of different MKL formulations for image categorization. In particular, we will compare L_1-MKL to L_p-MKL with $p = 2$ and $p = 4$. The final objective of this study is to evaluate the computational efficiency of different optimization algorithms for MKL. To this end, we choose seven representative MKL algorithms in our study (See Section 2.5.2).

2.5.1 Data sets, Features and Kernels

Three benchmark data sets for image categorization are used in our study: Caltech 101 [3], Pascal VOC 2007 [94], and a subset of ImageNet (see Appendix A). All the experiments conducted in this study are repeated five times, each with an independent random partition of training and testing data. Average classification accuracies along with the associated standard deviation are reported.

The Caltech 101: To obtain the full spectrum of classification performance for MKL, we vary the number of training examples per class (10, 20, 30). We construct 48 base kernels (Table 2.3) for the Caltech 101 data set: 39 of them are built by following the procedure in [43] and the remaining 9 are constructed by following [69]. For all the feature sets except the one that is based on geometric blur, RBF kernel with χ^2 distance is used as the kernel function [33]. For the geometric blur feature, RBF kernel with the average distance of the nearest descriptor pairs between two images is used [69].

Table 2.3: Description of the 48 kernels built for the Caltech 101 data set.

Kernel indices	Description	Color Space	# levels for SPK
1-3	LBP [95]	Gray	3
4	LBP (combined histogram)	Gray	3
5-8	BoW with dense-SIFT (300 bins)	HSV	4
9-12	BoW with dense-SIFT (1000 bins)	Gray	4
13-16	BoW with dense-SIFT (1000 bins)	HSV	4
17-18	SIFT on 100 sub-windows [40]	Gray-HSV	1
19-22	BoW with dense-SIFT (300 bins)	Gray	4
23-26	Canny edge detector + histogram of unoriented gradient feature (40 bins)	Gray	4
27-30	Canny edge detector + histogram of oriented gradient feature (40 bins) [96]	Gray	4
31,34, 33,34	Product of kernels: $\{20$ to $23\}$, $\{24$ to $27\}$, $\{16$ to $19\}$, and $\{4$ to $7\}$		1
35	V1S+ feature [97]	Gray	1
36-38	Region covariance [98]	Gray	3
39	Product of kernels 4 to 7		1
40	Geometric blur [99]	Gray	1
41-43	BoW with dense-SIFT (300 bins)	Gray	4
44-46	BoW with dense-SIFT (300 bins)	HSV	4
47-48	BoW (300 visual words) [100] with self-similarity features	Gray	2

The Pascal VOC 2007: Similar to the Caltech 101 data set, we vary the number of training examples, by randomly selecting 1%, 25%, 50%, and 75% of images to form the training set. Due

to the different characteristics of the two data sets, we choose a different set of image features for VOC 2007, suggested by the participants of the VOC Challenges. In particular, for the MKL experiments, we follow [101] and create 15 sets of features: (i) GIST features [102]; (ii) six sets of color features generated by two different spatial pooling layouts [103] (1×1 and 3×1), and three types of color histograms (i.e. RGB, LAB, and HSV). (iii) eight sets of local features generated by two key-point detection methods (i.e., dense sampling and Harris-Laplacian [104]), two spatial layouts (1×1 and 3×1), and two local descriptors (SIFT and robust hue descriptor [105]). An RBF kernel function with χ^2 distance is applied to each of the 15 feature sets.

A Subset of ImageNet: Following the protocol in [106], we use $81,738$ images from ImageNet that belong to the 18 (out of 20) categories specified in VOC 2007. This data set is significantly larger than Caltech 101 and VOC 2007, making it possible to examine the scalability of MKL methods for image categorization. Both dense sampling and Harris-Laplacian [104] are used for key-point detection, and SIFT is used as the local descriptor. We create four BoW models by setting the vocabulary size to be $1,000$ and applying two descriptor pooling techniques (i.e. max-pooling and mean-pooling) for two types of spatial partitioning (i.e. 1×1 and 2×2). We also create six color histograms by applying two pooling techniques (i.e. max-pooling and mean-pooling) to three different color spaces, namely RGB, LAB and HSV. In total, ten kernels are created for the ImageNet data set. We note that the number of base kernels we construct for the ImageNet data set is significantly smaller than the other two data sets because of the significantly larger number of images in the ImageNet data set. The common practice for large scale data sets has been to use a small number of features/kernels for scalability concerns [106].

2.5.2 MKL Methods Used in Comparison

We divide the MKL baselines into two groups. The first group consists of the two simple baselines for kernel combination, i.e., the average kernel method (AVG) and the best performing kernel selected by the cross validation method (Single). The second group includes seven MKL meth-

ods designed for binary classification. These are: GMKL [63], SimpleMKL [53], VSKL [64], MKL-GL [46], MKL-Level [52], MKL-SIP [56], MKL-SMO [70]. The difference between the two subgradient descent based methods, SimpleMKL and GMKL, is that SimpleMKL performs a golden section search to find the optimal step size while GMKL applies a simple backtracking method.

In addition to different optimization algorithms, we use L_1-MKL and L_p-MKL with $p = 2$ and $p = 4$. For L_p-MKL, we apply MKL-GL, MKL-SIP, and MKL-SMO to solve the related optimization problems.

2.5.3 Implementation

To make a fair comparison, we followed [46] and implemented all wrapper MKL methods within the framework of SimpleMKL using MATLAB, where we used LIBSVM [107] as the SVM solver. For MKL-SIP and MKL-Level, CVX [108] and MOSEK [89] were used to solve the related optimization problems, as suggested in [52].

The same stopping criteria were applied to all baselines. The algorithms were stopped when one of the following criteria is satisfied: (i) the maximum number of iterations (specified as 40 for wrapper methods) is reached, (ii) the difference in the kernel coefficients $\boldsymbol{\beta}$ between two consecutive iterations is small (i.e., $||\boldsymbol{\beta}^t - \boldsymbol{\beta}^{t-1}||_\infty < 10^{-4}$), (iii) the duality gap drops below a threshold value (10^{-3}).

The regularization parameter C was chosen with a grid search over $\{10^{-2}, 10^{-1}, \ldots, 10^4\}$. The bandwidth of RBF kernels was set to the average pair-wise χ^2 distance of image features.

In our empirical study, all the feature vectors were normalized to have the unit L_2 norm before they are used to construct the base kernels. According to [109] and [56], kernel normalization can have a significant impact on the performance of MKL. Various normalization methods have been proposed, including unit trace normalization [109], normalization with respect to the variance of kernel features [56], and spherical normalization [56]. However, we did not observed significant

differences in the classification accuracy when applied the above normalization techniques.

The experiments with varied numbers of kernels on the ImageNet data set were performed on a cluster of Sun Fire X4600 M2 nodes, each with 256 GB of RAM and 32 AMD Opteron cores. All other experiments were run on a different cluster, where each node has two four-core Intel Xeon E5620s at 2.4 GHz with 24 GB of RAM. We pre-computed all the kernel matrices and loaded them into the memory. This allowed us to avoid re-computing and loading kernel matrices at each iteration of optimization.

2.5.4 Classification Performance of MKL

We evaluate the classification performance by the category based mean average precision (MAP) score. For convenience, we report normalized MAP scores (percentage).

2.5.4.1 Experiment 1: Classification Performance

Table 2.4 summarizes the classification results for the Caltech 101 data set with 10, 20, and 30 training examples per class. First, we observe that both the MKL algorithms and the average kernel approach (AVG) outperform the best base kernel (Single). This is consistent with most of the previous studies [5, 69]. Compared to the average kernel approach, we observe that the L_1-MKL algorithms have the worst performance when the number of training examples per class is small ($n = 10, 20$), but significantly outperform the average kernel approach when $n = 30$. This result explains the seemingly contradictory conclusions reported in the literature. When the number of training examples is insufficient to determine the appropriate kernel combination, it is better to assign all the base kernels equal weights. MKL becomes effective only when the number of training examples is large enough to determine the optimal kernel combination.

Next, we compare the performance of L_1-MKL to that of L_p-MKLs. We observe that L_1-MKL performs worse than L_p-MKLs ($p = 2, 4$) when the number of training examples is small (i.e., $n = 10, 20$), but outperforms L_p-MKLs when $n = 30$. This result again explains why conflicting

Table 2.4: Classification results (MAP) for the Caltech 101 data set. We report the average values over five random splits and the associated standard deviation.

		Number of training instances per class		
Baseline	Norm	10	20	30
Single		45.3 ± 0.9	55.2 ± 0.9	70.6 ± 0.9
Average		**59.0** $\pm$ **0.7**	**69.7** $\pm$ **0.6**	77.2 ± 0.5
GMKL	$p = 1$	54.2 ± 1.1	64.1 ± 0.7	**84.8** $\pm$ **0.7**
SimpleMKL	$p = 1$	53.6 ± 0.9	63.4 ± 0.6	**84.6** $\pm$ **0.5**
VSKL	$p = 1$	53.9 ± 0.9	64.0 ± 0.6	**85.3** $\pm$ **0.5**
level-MKL	$p = 1$	54.7 ± 1.0	63.4 ± 0.6	**84.4** ± 0.4
MKL-GL	$p = 1$	54.3 ± 1.0	64.7 ± 0.7	**85.4** $\pm$ **0.4**
MKL-GL	$p = 2$	**60.3** $\pm$ **0.6**	**70.7** $\pm$ **1.0**	80.0 ± 0.6
MKL-GL	$p = 4$	**60.1** $\pm$ **0.7**	**70.7** $\pm$ **1.0**	80.0 ± 0.6
MKL-SIP	$p = 1$	53.8 ± 0.6	63.8 ± 0.9	**83.9** $\pm$ **0.7**
MKL-SIP	$p = 2$	**60.1** $\pm$ **0.6**	**70.7** $\pm$ **1.0**	79.1 ± 0.6
MKL-SIP	$p = 4$	**59.4** $\pm$ **0.6**	**70.0** $\pm$ **1.0**	77.5 ± 0.5
MKL-SMO	$p = 2$	**59.8** $\pm$ **0.5**	**69.7.0** $\pm$ **0.9**	79.3 ± 0.9
MKL-SMO	$p = 4$	**59.6** $\pm$ **0.4**	**69.6** $\pm$ **0.7**	79.0 ± 0.5

results were observed in different MKL studies in the literature. Compared to L_p-MKL, L_1-MKL gives a sparser solution for the kernel combination weights, leading to the elimination of irrelevant kernels. When the number of training examples is small, it is difficult to determine the subset of kernels that are irrelevant to a given task. As a result, the sparse solution obtained by L_1-MKL may be inaccurate, leading to a relatively lower classification accuracy than L_p-MKL. L_1-MKL becomes advantageous when the number of training examples is large enough to determine the subset of relevant kernels.

We observe that there is no significant difference in the classification performance between different MKL optimization techniques. This is not surprising since they solve the same optimization problem. It is interesting to note that although different optimization algorithms converge to the same solution, they could behave very differently over iterations. In Figures 2.2, 2.3, and 2.4, we show how the classification performances of the L_1-MKL algorithms change over the iterations for three classes from Caltech101 data set. We observe that,

- SimpleMKL converges in a smaller number of iterations compared to the other L_1-MKL

Table 2.5: Classification results (MAP) for the VOC 2007 data set. We report the average values over five random splits and the associated standard deviation.

	Percentage of the samples used for training			
baseline	1%	25%	50%	75%
Single	**23.4** $\pm$ **0.1**	44.7 $\pm$ 0.8	48.6 $\pm$ 0.8	50.0 $\pm$ 0.8
Average	21.9 $\pm$ 0.5	48.2 $\pm$ 0.8	54.5 $\pm$ 0.8	57.5 $\pm$ 0.8
L_1-MKL	**23.5** $\pm$ **0.7**	**51.9** $\pm$ **0.4**	**57.4** $\pm$ **0.4**	**59.9** $\pm$ **0.9**
L_2-MKL	22.7 $\pm$ 0.4	49.8 $\pm$ 0.2	**57.3** $\pm$ **0.2**	**60.6** $\pm$ **0.5**

algorithms. Note that convergence in a smaller number of iterations does not necessarily mean a shorter training time, as SimpleMKL takes significantly longer time to finish one iteration.

- The classification performance of MKL-SIP fluctuates significantly over iterations. This is due to the greedy nature of MKL-SIP as it selects the most violated constraints at each iteration of optimization.

For simplicity, from now on, unless specified, we will only report the results of one representative method for both L_1-MKL (Level-MKL) and L_p-MKL (MKL-SIP, $p = 2$).

Table 2.5 shows the classification results for the VOC 2007 data set with 1%, 25%, 50%, and 75% of images used for training. These results confirm the conclusions drawn from the Caltech 101 data set: MKL methods do not outperform the simple baseline (i.e., the best single kernel) when the number of training examples is small (e.g., 1%); the advantage of MKL is clear only when the number of training examples is sufficiently large.

Finally, we compare in Table 2.6 the performance of MKL to that of the state-of-the-art methods for image categorization on the Caltech 101 and VOC 2007 data sets. For Caltech 101, we use the standard splitting formed by randomly selecting 30 training examples for each class, and for VOC 2007, we use the default partitioning. We observe that the L_1-MKL achieves similar classification performance as the state-of-the-art approaches for the Caltech 101 data set. However, for the VOC 2007 data set, the performance of MKL is significantly worse than the best ones [112, 113].

Table 2.6: Comparison with the state-of-the-art performance for object classification on the Caltech 101 (measured by classification accuracy) and VOC 2007 data sets (measured by MAP).

Caltech 101 (30 per class)			
This paper		state-of-the-art	
AVG :	77.09	**[5]**:	**84.3**
L_1-MKL :	79.93	[110]:	81.9
L_2-MKL :	77.94	[111]:	80.0
VOC 2007			
This paper		state-of-the-art	
AVG:	55.4	**[112]**:	**73.0**
L_1-MKL:	57.2	[113]:	63.5
L_2-MKL:	57.4	[114]:	61.7

The gap in the classification performance is because object detection (localization) methods are utilized in [112, 113] to boost the recognition accuracy for the VOC 2007 data set but not in this dissertation. We also note that the authors of [114] get a better result by using only one strong and well-designed (Fisher vector) representation compared to the MKL results we report. Interested readers are referred to [114], which provides an empirical study that shows how the different steps of the BoW model can affect the classification results. Note that the performance of MKL techniques can be improved further by using the different and stronger options discussed in [114].

2.5.4.2 Experiment 2: Number of Kernels vs. Classification Accuracy

In this experiment, we examine the performance of MKL methods with increasing numbers of base kernels. To this end, we rank the kernels in the descending order of their weights computed by L_1-MKL, and measure the performance of MKL and baseline methods by adding kernels sequentially. The number of kernels is varied from 2 to 48 for the Caltech 101 data set and from 2 to 15 for the VOC 2007 data set. Figures 2.5 and 2.6 summarizes the classification performance of MKL and baseline methods as the number of kernels is increased. We observe that when the number of kernels is small, all the methods are able to improve their classification performance with increasing number of kernels. But, the performance of average kernel and L_2-MKL starts to

drop as more and more weak kernels (i.e., kernels with small weights computed by L_1-MKL) are added. In contrast, we observe a performance saturation for L_1-MKL after five to ten kernels have been added. We thus conclude that L_1-MKL is more resilient to the introduction of weak kernels than the other kernel combination methods.

2.5.5 Computational Efficiency

To evaluate the learning efficiency of MKL algorithms, we report training time for the experiments with different numbers of training examples and base kernels. Many studies on the computational efficiency of MKL algorithms focused on the convergence rate (i.e., number of iterations) [52], which is not necessarily the deciding factor in determining the training time. For instance, according to Figure 2.2, although SimpleMKL requires a smaller number of iterations to obtain the optimal solution than the other L_1-MKL approaches, it is significantly slower in terms of running time than the other algorithms because of its high computational cost per iteration. Thus, besides the training time, we also examine the sparseness of the kernel coefficients, which can significantly affect the efficiency of both training and testing.

2.5.5.1 Experiment 4: Evaluation of Training Time

We first examine how the number of training examples affects the training time of the wrapper methods. Tables 2.8 and 2.9 summarize the training time of different MKL algorithms for the Caltech 101 and VOC 2007 data sets, respectively. We also include in the table the number of iterations and the time for computing the combined kernel matrices. We did not include the time for computing kernel matrices because it is shared by all the methods. We draw the following observations from Tables 2.8 and 2.9:

- The L_p-MKL methods require a considerably smaller number of iterations than the L_1-MKL methods, indicating they are computationally more efficient. This is not surprising because

L_p-MKL employs a smooth objective function that leads to more efficient optimization [85].

- Since a majority of the training times is spent on computing combined kernel matrices, the time difference between different L_1-MKL methods is mainly due to the sparseness of their intermediate solutions. Since MKL-SIP yields sparse solutions throughout its optimization process, it is the most efficient wrapper algorithm for MKL. Although SimpleMKL converges in a smaller number of iterations than the other L_1-MKL methods, it is not as efficient as the MKL-SIP method because it does not generate sparse intermediate solutions.

In the second set of experiments, we evaluate the training time as a function of the number of base kernels. For both the Caltech 101 and VOC 2007 data sets, we choose 15 kernels with the best classification accuracy, and create 15, 30, and 60 kernels by simply varying the kernel bandwidth (i.e., from 1 times, to 1.5 and 2 times the average χ^2 distance). The number of training examples is set to be 30 per class for Caltech 101 and 50% of images are used for training for VOC 2007. Tables 2.10 and 2.11 summarize for different MKL algorithms, the training time, the number of iterations, and the time for computing the combined kernel matrices. Overall, we observe that L_p-MKL is still more efficient than L_1-MKL, even when the number of base kernels is large. But the gap in the training time between L_1-MKL and L_p-MKL becomes significantly smaller for the MKL-SIP method when the number of combined kernels is large. In fact, for the Caltech 101 data set with 108 base kernels, MKL-SIP for L_1-MKL is significantly more efficient than MKL-SIP for L_p-MKL ($p > 1$). This is because of the sparse solution obtained by MKL-SIP for L_1-MKL, which leads to less time on computing the combined kernels than MKL-SIP for L_p-MKL, as indicated in Tables 2.10 and 2.11.

As discussed in Section 2.5.3, we cannot compare MKL-SMO directly with the other baselines in terms of training times since they are not coded in the same platform. Instead, we use the code provided by the authors of MKL-SMO [70] to compare it to the C++ implementation of MKL-SIP, the fastest wrapper approach, which is available within the Shogun package [115]. We fix $p = 2$,

Table 2.7: Comparison of training time between MKL-SMO and MKL-SIP

		Number of training samples		
Caltech 101		$n = 10$	$n = 20$	$n = 30$
	MKL-SIP	3.6 ±0.2	6.5± 0.3	11.8 ± 0.7
	MKL-SMO	**0.2 ±0.1**	**2.3 ± 0.2**	**3.8 ± 0.5**
VOC 2007		25%	50%	75%
	MKL-SIP	15.5 ± 1.6	145.6 ± 3.9	360.7 ± 8.4
	MKL-SMO	**3.5 ± 0.7**	**14.2± 1.8**	**33.1± 3.0**
		Number of base kernels		
Caltech 101		$K = 48$	$K = 63$	$K = 108$
	MKL-SIP	6.5 ± 0.3	13.6 ± 2.9	19.8 ± 3.4
	MKL-SMO	**2.3 ± 0.2**	**3.2± 0.8**	**6.3± 1.0**
VOC 2007		$K = 15$	$K = 30$	$K = 75$
	MKL-SIP	145.6 ± 3.9	542.0 ± 32.8	1412.1 ± 63.4
	MKL-SMO	**14.2 ± 1.8**	**29.1± 2.8**	**77.8± 10.3**

vary the number of training samples for a fixed number of kernels (48 for Caltech 101 and 15 for VOC 2007) and the number of base kernels for a fixed number of samples (2,040 for Caltech 101 and 5,011 for VOC 2007). Table 2.7 shows that MKL-SMO is significantly faster than MKL-SIP on both data sets, demonstrating the advantage of a well-designed direct MKL optimization method against the wrapper approaches for L_p-MKL. We finally note that MKL-SMO cannot be applied to L_1-MKL which often demonstrates better performance with a modest number of training examples.

2.5.5.2 Experiment 5: Evaluation of Sparseness

We evaluate the sparseness of MKL algorithms by examining the sparsity of the solution for kernel combination coefficients. In Figures 2.7 and 2.8, we show how the size of active kernel set (i.e., kernels with non-zero combination weights) changes over the iterations for MKL-SIP with three types of regularizers: L_1-MKL, L_2-MKL and L_4-MKL. Note that it is difficult to distinguish the

results of L_2-MKL and L_4-MKL from each other as they are identical.

As expected, L_1-MKL method produces significantly sparser solutions than L_p-MKL. As a result, although L_p-MKL is more efficient for training because it takes a smaller number of iterations to train L_p-MKL than L_1-MKL, we expect L_1-MKL to be computationally more efficient for testing than L_p-MKL as most of the base kernels are eliminated and need not to be considered.

2.5.6 Large-scale MKL on ImageNet

To evaluate the scalability of MKL, we perform experiments on the subset of ImageNet consisting of $81,738$ images. Figure 3.10 shows the classification performance of MKL and baseline methods with the number of training images per class varied in powers of 2 ($2^1, 2^2, ..., 2^{11}$). Similar to the experimental results for Caltech 101 and VOC 2007, we observed that the difference between L_1-MKL and the average kernel method is significant only when the number of training examples per class is sufficiently large (i.e. ≥ 16). We also observed that the difference between L_1-MKL and the average kernel method starts to diminish when the number of training examples is increased over 256 per class. We believe that the diminishing gap between MKL and the average kernel method with increasing number of training examples can be attributed to the fact that all the 10 base kernels constructed for the ImageNet data set are strong kernels and provide informative features for image categorization. This is reflected in the kernel combination weights learned by the MKL method: most of the base kernels received significant non-zero weights.

Figure 2.10 shows the running time of MKL with a varied number of training examples. Similar to the experimental results for Caltech 101 and VOC 2007, we observe that L_2-MKL is significantly more efficient than L_1-MKL. We also observe that the running time for both L_1-MKL and L_2-MKL increases almost quadratically in the size of training data, making it difficult to scale to millions of training examples. We thus conclude that although MKL is effective in combining multiple image representations for image categorization, scalability of MKL algorithms is an open problem.

2.6 Summary and Conclusions

In this chapter, we have reviewed different formulations of multiple kernel learning and related optimization algorithms, with an emphasis on the application to image categorization. We highlighted the conflicting conclusions drawn by published studies on the empirical performance of different MKL algorithms. We have attempted to resolve these inconsistent conclusions by addressing the experimental setups in the published studies. Through our extensive experiments on three standard data sets used for image categorization, we are able to make the following conclusions:

- Overall, MKL is significantly more effective than the simple baselines for kernel combination (i.e., selecting the best kernel by cross validation or taking the average of multiple kernels), particularly when there are a large number of base kernels available, and the number of training examples is sufficiently large. However, MKL is not recommended for image categorization when the base kernels are strong, and the number of training examples are sufficient enough to learn a reliable prediction for each base kernel.

- Compared to L_p-MKL, L_1-MKL is overall more effective for image categorization and is significantly more robust to the weaker kernels with low classification performance.

- MKL-SMO, which is not a wrapper method but a direct optimization technique, is the fastest MKL baseline. However, it does not address the L_1-MKL formulation.

- Among various algorithms proposed for L_1-MKL, MKL-SIP is overall the most efficient for image categorization, because it produces sparse intermediate solutions throughout the optimization process.

- L_p-MKL is significantly more efficient than L_1-MKL because it converges in a significantly smaller number of iterations. But, neither L_1-MKL nor L_p-MKL scale well to very large data sets.

- L_1-MKL can be more efficient than L_p-MKL in terms of prediction time. This is because L_1-MKL generates sparse solutions and, therefore, will only use a small portion of the base kernels for prediction.

In summary, we conclude that MKL is an extremely useful tool for image categorization because it provides a principled way to combine the strengths of different image representations. Although MKL methods have demonstrated significant success for image categorization, there is still room for improvement. One of the most important directions for improving the accuracy of MKL methods is developing MKL algorithms that addresses the need of multi-label data, such as image categorization data sets. To this end, we propose a multiple kernel multi-label ranking method in Chapter 6. It is also very critical to improve the overall computational efficiency of MKL. The existing algorithms for MKL do not scale to large data sets with millions of images and thousands of classes. In the next chapter, we discuss our efforts on reducing the computational load of MKL for large-scale multi-label data sets.

Table 2.8: Total training time (seconds), number of iterations, and total time spent on combining the base kernels (seconds) for different MKL algorithms vs. number of training examples for Caltech 101.

	10 training instances per class		
baseline	training	#iter	KerComb
GMKL-L_1	34.6 ± 8.6	38.4 ± 2.0	27.9 ± 7.7
SimpleMKL-L_1	55.7 ± 25.3	17.2 ± 6.8	46.1 ± 22.0
VSKL-L_1	14.1 ± 2.3	38.3 ± 4.3	11.1 ± 1.7
MKL-GL-L_1	21.9 ± 0.8	40.0 ± 0.0	19.5 ± 0.8
MKL-GL-L_2	5.3 ± 0.6	8.8 ± 1.0	4.8 ± 0.6
MKL-GL-L_4	**3.5** $\pm$ **0.2**	**5.9** $\pm$ **0.4**	3.2 ± 0.2
MKL-Level-L_1	8.0 ± 2.3	33.0 ± 9.5	5.5 ± 1.4
MKL-SIP-L_1	5.4 ± 0.9	39.4 ± 2.6	**2.1** $\pm$ **0.3**
MKL-SIP-L_2	**3.8**$\pm$**1.2**	**5.6**$\pm$**0.9**	**2.4**$\pm$**1.1**
MKL-SIP-L_4	**3.3**$\pm$**0.6**	**4.4**$\pm$**0.5**	**1.8**$\pm$**0.6**
	30 training instances per class		
baseline	training	#iter	KerComb
GMKL-L_1	256.7 ± 47.7	38.6 ± 1.8	212.5 ± 42.3
SimpleMKL-L_1	585.6 ± 204.7	19.0 ± 7.5	494.4 ± 174.7
VSKL-L_1	121.9 ± 22.4	36.6 ± 5.1	103.5 ± 17.7
MKL-GL-L_1	197.1 ± 9.1	39.8 ± 1.0	178.3 ± 8.5
MKL-GL-L_2	50.8 ± 5.6	9.3 ± 1.0	46.3 ± 5.2
MKL-GL-L_4	32.5 ± 1.6	5.9 ± 0.3	29.6 ± 1.5
MKL-Level-L_1	63.3 ± 22.1	27.5 ± 11.1	47.9 ± 14.9
MKL-SIP-L_1	44.3 ± 6.1	39.7 ± 2.9	23.2 ± 2.7
MKL-SIP-L_2	30.4$\pm$4.2	6.3$\pm$1.0	25.2$\pm$3.9
MKL-SIP-L_4	**22.6**$\pm$**2.6**	**4.7**$\pm$**0.5**	**18.2**$\pm$**2.1**

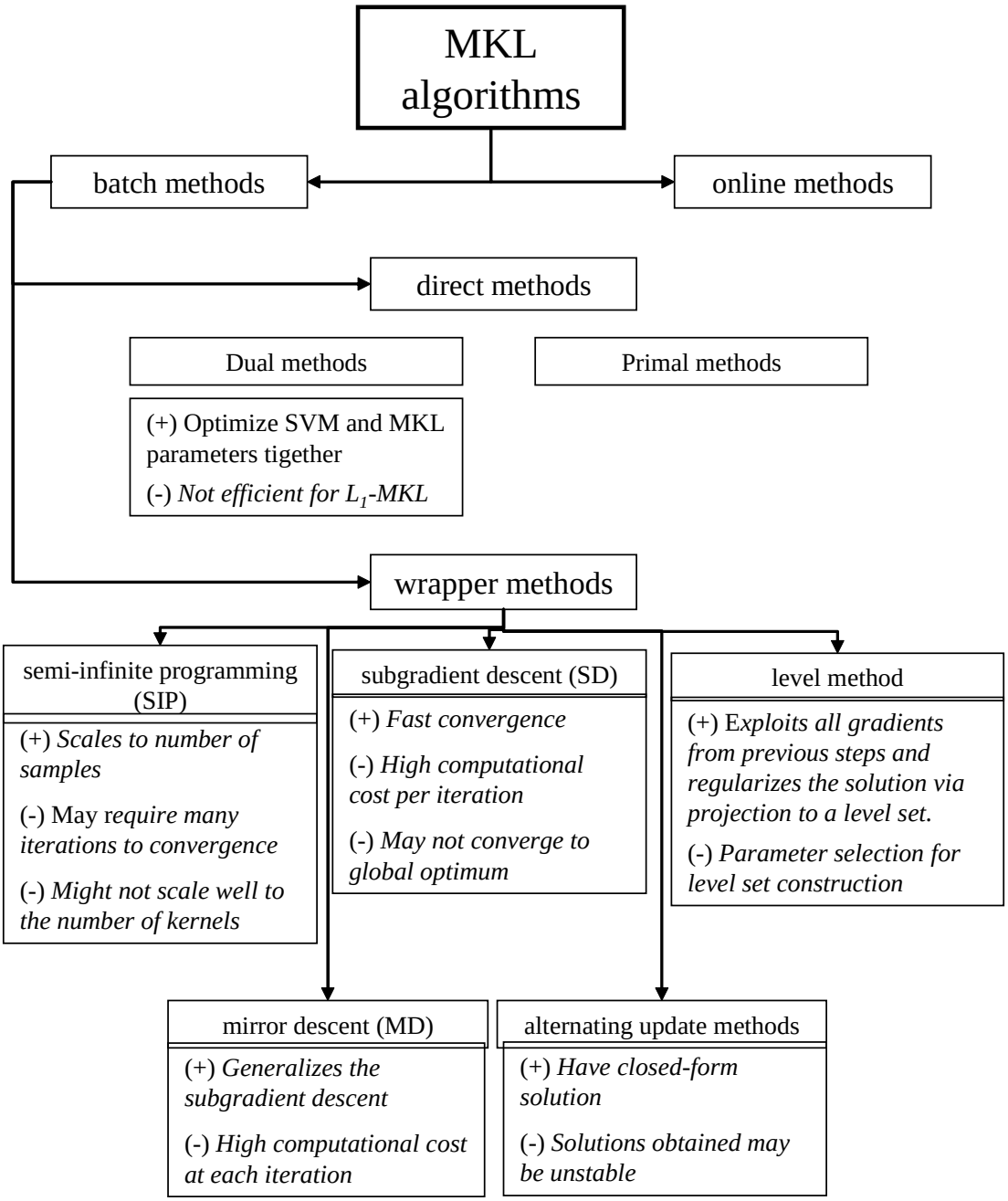

Figure 2.1: A summary of representative MKL optimization schemes

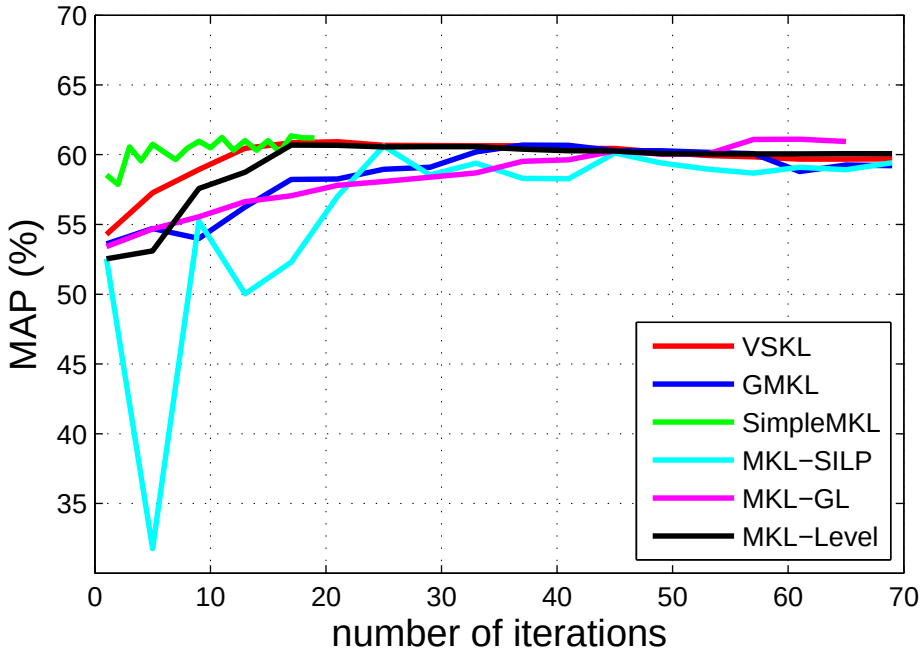

Figure 2.2: Mean average precision (MAP) scores of different L_1-MKL methods vs. number of iterations for the *anchor* class of the Caltech101 data set.

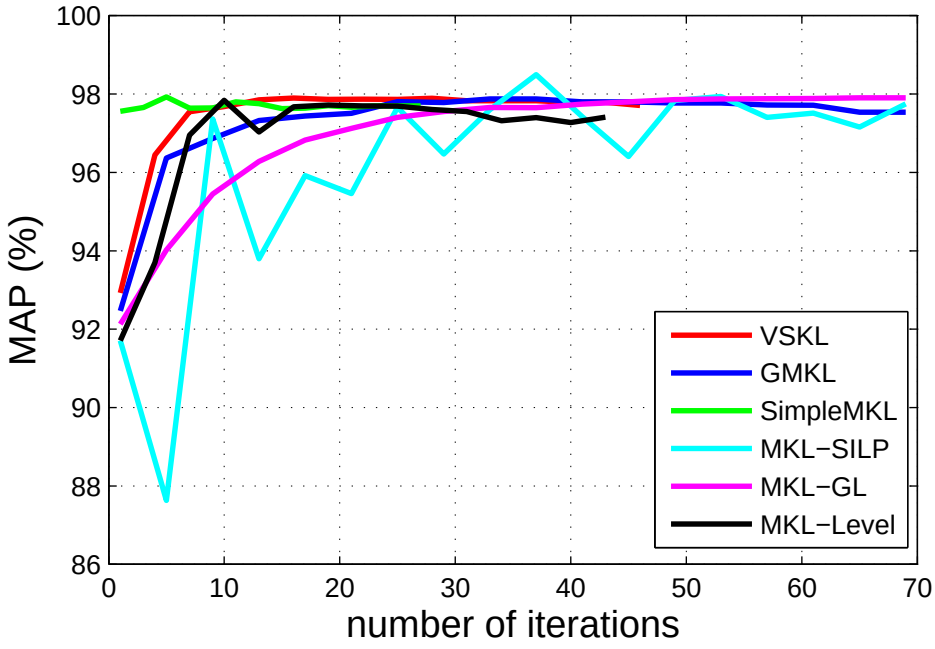

Figure 2.3: Mean average precision (MAP) scores of different L_1-MKL methods vs. number of iterations for the *bonsai* class of the Caltech101 data set.

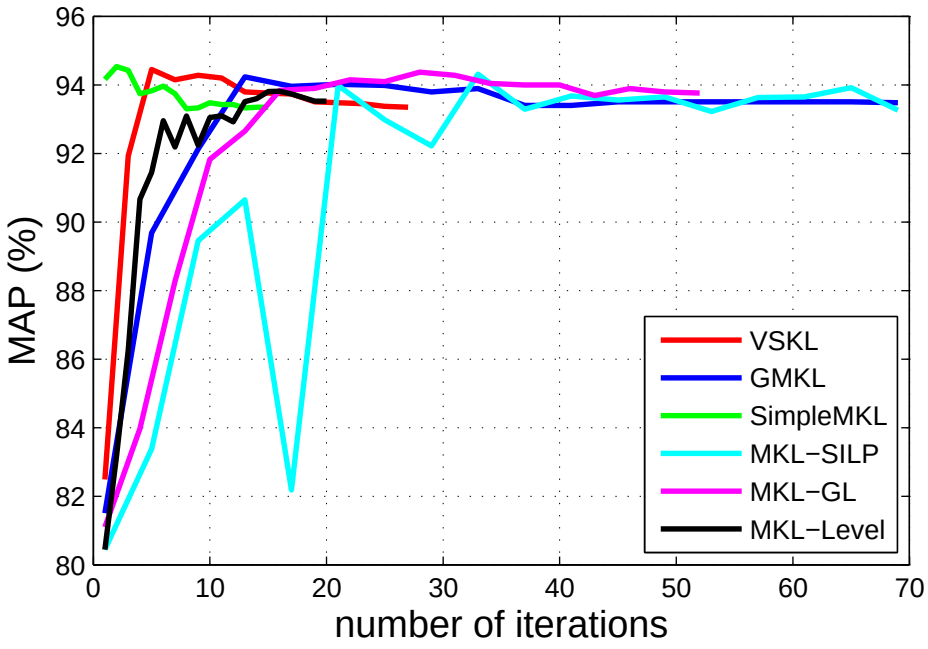

Figure 2.4: Mean average precision (MAP) scores of different L_1-MKL methods vs. number of iterations for the *camera* class of the Caltech101 data set.

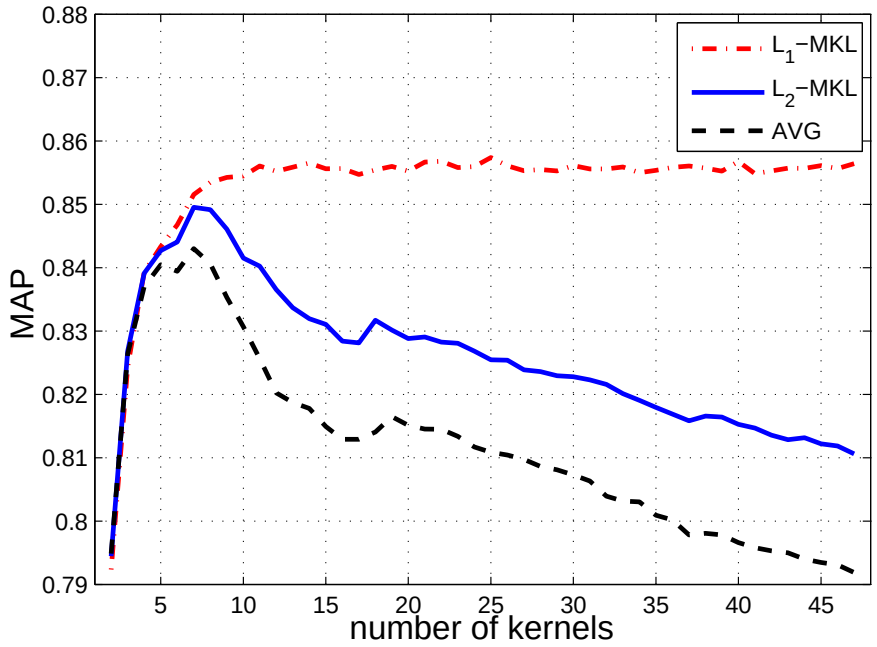

Figure 2.5: The change in MAP score with respect to the number of base kernels for the Caltech 101 data set.

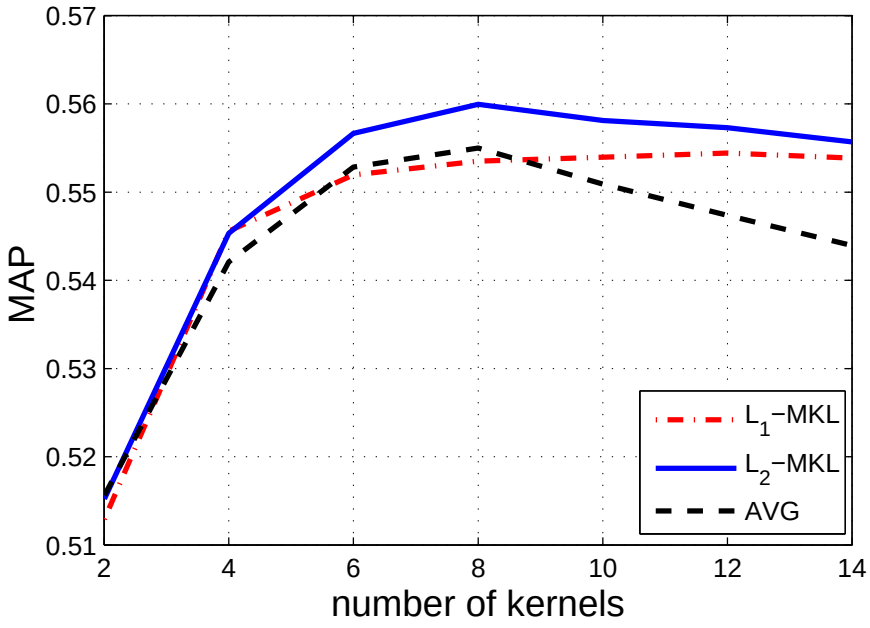

Figure 2.6: The change in MAP score with respect to the number of base kernels for the VOC 2007 data set.

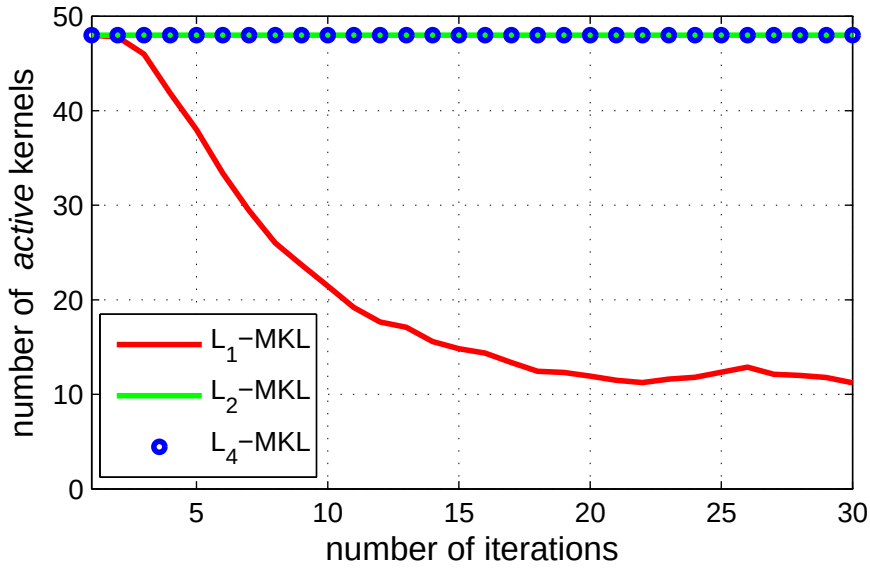

Figure 2.7: Number of active kernels learned by the MKL-SIP algorithm vs. number of iterations for the Caltech 101 data set. Note that it is difficult to distinguish the results of L_2-MKL and L_4-MKL from each other as they are identical.

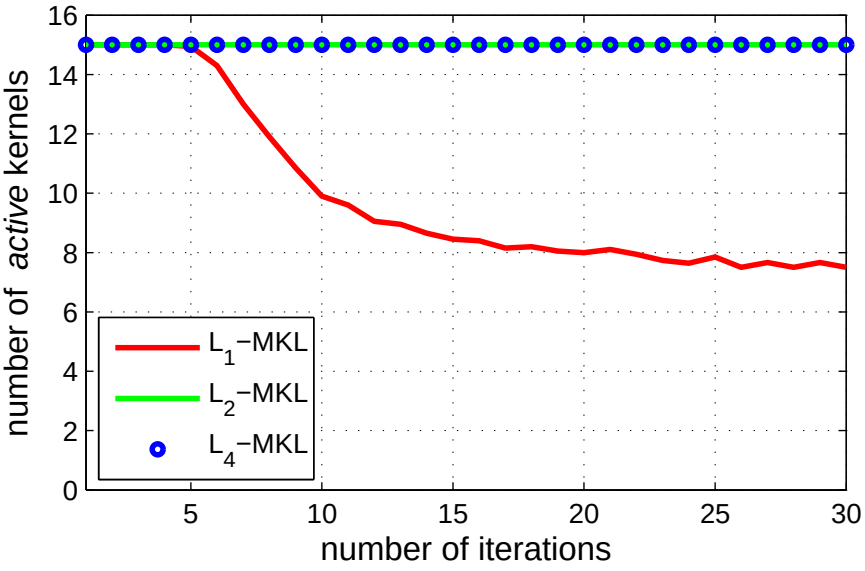

Figure 2.8: Number of active kernels learned by the MKL-SIP algorithm vs. number of iterations for the VOC 2007 data set. Note that it is difficult to distinguish the results of L_2-MKL and L_4-MKL from each other as they are identical.

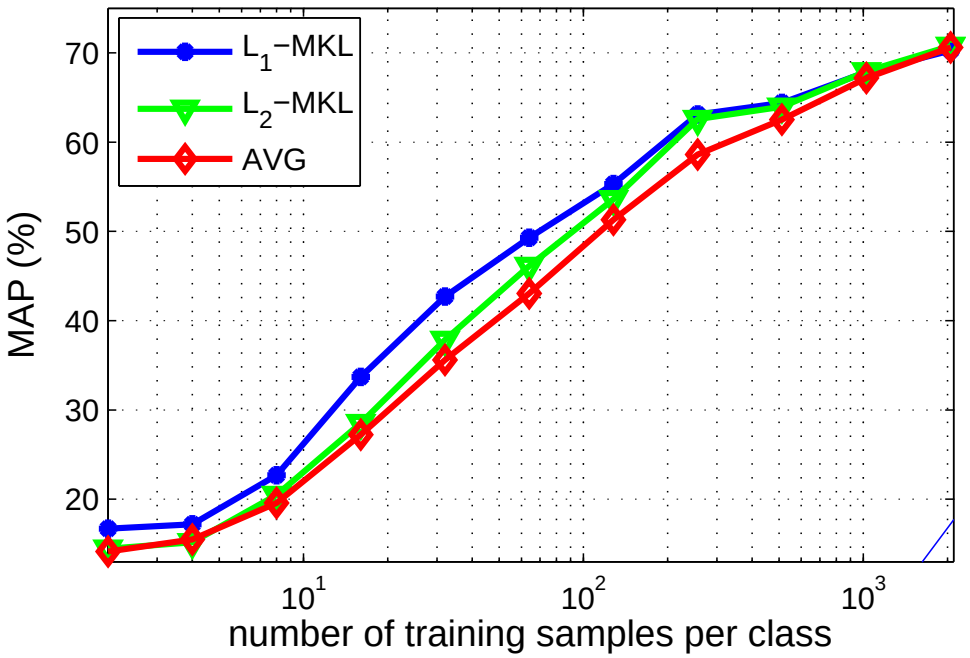

Figure 2.9: Classification performance for different training set sizes for the ImageNet data set.

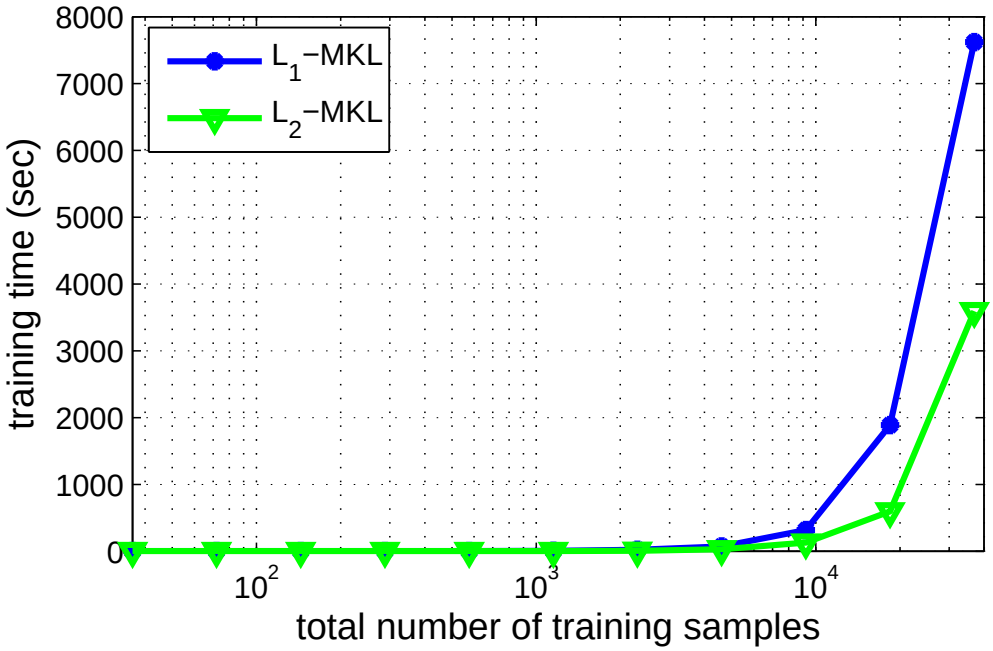

Figure 2.10: Training times for L_1-MKL and L_2-MKL on different training set sizes for the ImageNet data set.

Table 2.9: Total training time (seconds), number of iterations, and total time spent on combining the base kernels (seconds) for different MKL algorithms vs. number of training examples for the VOC 2007 data set.

	$2,500$ training instances		
baseline	training	#iter	KerComb
GMKL-L_1	117.6 ± 16.3	39.0 ± 0.0	67.4 ± 7.7
SimpleMKL-L_1	175.1 ± 77.4	16.7 ± 7.3	112.9 ± 48.3
VSKL-L_1	45.2 ± 6.1	37.0 ± 3.4	25.3 ± 2.2
MKL-GL-L_1	62.6 ± 4.7	40.0 ± 0.0	43.5 ± 0.6
MKL-GL-L_2	14.5 ± 1.3	9.3 ± 0.6	10.2 ± 0.7
MKL-GL-L_4	**8.0** $\pm$ **0.8**	5.2 ± 0.4	5.6 ± 0.5
MKL-Level-L_1	40.1 ± 10.8	35.0 ± 7.7	20.2 ± 4.0
MKL-SIP-L_1	34.6 ± 6.8	39.9 ± 0.5	12.7 ± 1.4
MKL-SIP-L_2	**9.6**$\pm$**1.9**	5.7$\pm$0.5	4.9$\pm$0.4
MKL-SIP-L_4	**7.1**$\pm$**1.1**	**4.0**$\pm$**0.0**	**3.5**$\pm$**0.1**
	$7,500$ training instances		
baseline	training	#iter	KerComb
GMKL-L_1	1133.2 ± 252.8	39.0 ± 0.0	646.9 ± 98.2
SimpleMKL-L_1	1671.3 ± 919.1	16.8 ± 6.4	1019.7 ± 424.8
VSKL-L_1	330.0 ± 49.2	29.9 ± 3.8	190.9 ± 22.8
MKL-GL-L_1	549.2 ± 79.8	40.0 ± 0.0	373.8 ± 4.2
MKL-GL-L_2	130.1 ± 17.7	9.5 ± 0.5	89.4 ± 6.1
MKL-GL-L_4	**74.9** $\pm$ **11.1**	5.3 ± 0.5	51.2 ± 4.5
MKL-Level-L_1	297.3 ± 95.2	31.1 ± 8.1	151.9 ± 31.0
MKL-SIP-L_1	309.0 ± 94.5	40.0 ± 0.0	117.0 ± 6.4
MKL-SIP-L_2	**84.3**$\pm$**24.5**	6.1$\pm$0.3	47.3$\pm$3.0
MKL-SIP-L_4	**56.4**$\pm$**14.7**	**4.1**$\pm$**0.3**	**31.5**$\pm$**2.2**

Table 2.10: Total training time (seconds), number of iterations, and total time spent on combining the base kernels (seconds) for different MKL algorithms vs. number of base kernels for the Caltech 101 data set.

	63 base kernels		
baseline	training	#iter	KerComb
GMKL-L_1	718.1 ± 169.8	38.8 ± 0.8	625.3 ± 152.9
SimpleMKL-L_1	1255.2 ± 350.9	17.3 ± 6.5	1047.6 ± 285.8
VSKL-L_1	398.1 ± 123.7	36.3 ± 5.2	345.6 ± 101.5
MKL-GL-L_1	397.1 ± 30.0	39.8 ± 1.0	351.9 ± 26.7
MKL-GL-L_2	118.8 ± 14.7	9.3 ± 1.0	108.5 ± 13.7
MKL-GL-L_4	**84.6** $\pm$ **5.8**	**6.0** $\pm$ **0.0**	**77.3** $\pm$ **4.8**
MKL-Level-L_1	204.1 ± 75.7	27.8 ± 10.4	167.2 ± 56.1
MKL-SIP-L_1	147.8 ± 29.8	39.8 ± 2.4	**85.3** $\pm$ **15.0**
MKL-SIP-L_2	**114.7**$\pm$**36.7**	7.9$\pm$0.7	**102.7**$\pm$**33.6**
MKL-SIP-L_4	**111.1**$\pm$**38.8**	7.5$\pm$0.8	**98.3**$\pm$**34.5** 9
	108 base kernels		
baseline	training	#iter	KerComb
GMKL-L_1	1170.5 ± 208.7	38.9 ± 0.8	1049.2 ± 190.7
SimpleMKL-L_1	2206.3 ± 580.1	17.2 ± 6.4	1960.3 ± 503.5
VSKL-L_1	569.9 ± 160.3	35.6 ± 5.9	491.8 ± 131.2
MKL-GL-L_1	604.6 ± 69.9	39.6 ± 1.6	546.6 ± 66.0
MKL-GL-L_2	226.3 ± 24.8	9.5 ± 1.0	212.0 ± 23.6
MKL-GL-L_4	**169.1** $\pm$ **16.0**	6.0 ± 0.1	**158.2** $\pm$ **14.5**
MKL-Level-L_1	405.8 ± 152.7	29.5 ± 9.5	343.7 ± 121.3
MKL-SIP-L_1	**192.1** $\pm$ **41.3**	39.9 ± 0.9	110.1 ± 18.1
MKL-SIP-L_2	634.1$\pm$107.2	6.8$\pm$1.3	582.1$\pm$106.3
MKL-SIP-L_4	407.2$\pm$80.2	**4.6**$\pm$**0.6**	368.4$\pm$67.9

Table 2.11: Total training time (seconds), number of iterations, and total time spent on combining the base kernels (seconds) for different MKL algorithms vs. number of base kernels for the VOC 2007 data set.

	30 base kernels		
baseline	training	#iter	KerComb
GMKL-L_1	1816.8 ± 405.8	37.8 ± 5.4	1186.9 ± 270.4
SimpleMKL-L_1	2335.3 ± 991.9	11.2 ± 7.1	1581.6 ± 626.4
VSKL-L_1	880.2 ± 128.5	30.6 ± 3.8	525.5 ± 75.3
MKL-GL-L_1	853.5 ± 206.1	40.0 ± 0.0	561.8 ± 107.3
MKL-GL-L_2	282.4 ± 64.2	9.6 ± 0.5	218.2 ± 46.3
MKL-GL-L_4	**190.1 ± 23.9**	**6.0 ± 0.0**	147.4 ± 11.0
MKL-Level-L_1	665.4 ± 114.7	36.8 ± 5.1	404.7 ± 40.2
MKL-SIP-L_1	460.0 ± 135.5	40.0 ± 0.0	170.6 ± 23.1
MKL-SIP-L_2	**240.8±62.5**	8.7±1.6	**154.5±43.5**
MKL-SIP-L_4	**170.1±16.5**	**6.2±0.4**	**115.1±15.4**
	75 base kernels		
baseline	training	#iter	KerComb
GMKL-L_1	3975.3 ± 890.0	34.2 ± 8.8	3072.5 ± 724.5
SimpleMKL-L_1	3416.3 ± 1299.7	**8.3 ± 7.8**	2776.4 ± 885.7
VSKL-L_1	1587.9 ± 238.8	29.4 ± 3.7	909.3 ± 122.2
MKL-GL-L_1	1500.4 ± 239.4	40.0 ± 0.0	1043.8 ± 87.6
MKL-GL-L_2	629.5 ± 84.0	9.8 ± 0.4	520.4 ± 47.7
MKL-GL-L_4	**346.2 ± 45.3**	6.0 ± 0.0	**286.2 ± 31.9**
MKL-Level-L_1	1136.8 ± 328.9	36.7 ± 3.1	702.2 ± 177.7
MKL-SIP-L_1	686.8 ± 262.9	40.0 ± 0.0	**228.5 ± 46.0**
MKL-SIP-L_2	**413.9±258.1**	**3.8±1.7**	**302.2±135.7**
MKL-SIP-L_4	**566.4±141.9**	**5.0±0**	424.2±81.5

Chapter 3

Multi-label Multiple Kernel Learning by Stochastic Approximation

3.1 Introduction

In Chapter 2, we provided a detailed review of MKL and a set of empirical analyses on image categorization data sets to demonstrate the effectiveness of MKL. The focus of Chapter 2 was the MKL methods for the binary classification problem, which constitutes the majority of the MKL literature. The application of MKL to multi-labeled data, such as image categorization data, is mostly limited to a use of one-vs-all framework for MKL, which has two main drawbacks. First, one-vs-all framework requires training a MKL algorithm separately for each class. Considering that there are thousands of training instances and hundreds of classes in recent image categorization data sets, training a one-vs-all MKL solver would be computationally demanding. Second, one-vs-all framework cannot exploit label correlations, since MKL solvers for each class are operated independently, meaning that no interaction of information transfer is available. It has been shown in many multi-label learning studies that learning independent classifiers for each class gives suboptimal performance compared to direct approaches which consider all classes together in the

learning process. In this chapter, we present an efficient algorithm for multi-label multiple kernel learning (ML-MKL). We assume that all the classes under consideration share the same combination of kernel functions, and the objective is to find the optimal kernel combination that benefits all the classes. Although several algorithms have been developed for ML-MKL, their computational cost is linear in the number of classes; therefore, they do not scale well when the number of classes increases, a challenge frequently encountered image categorization. We address this computational challenge by developing a framework for ML-MKL that combines the worst-case analysis with stochastic approximation. Our analysis shows that the complexity of our algorithm is $O(m^{1/3}\sqrt{lnm})$, where m is the number of classes.

This Chapter is organized as follows: in Section 3.2, we provide a brief literature review on MKL for multi-class and multi-label learning. Next, we introduce our multi-label MKL formulation and give an efficient algorithm to solve it. A convergence analysis for the proposed algorithm is provided in Section 3.3.2. In Section 3.4, we provide empirical analyses that demonstrate the strength of the proposed framework on benchmark data sets. We end the chapter with the concluding remarks and future directions in Section 3.5.

3.2 Previous Work

There is a large body of literature on MKL, and we provided a detailed review of binary MKL methods in Chapter 2. Although most efforts in MKL focus on binary classification problems, several studies have attempted to extend MKL to multi-class and multi-label learning [5, 68, 87, 116, 117]. Even though studies show that MKL for multi-class and multi-label learning can result in significant improvement in classification accuracy, the computational cost is often linear in the number of classes, making it computationally expensive when dealing with a large number of classes. Since most image categorization problems involve many image classes, whose number might go up to hundreds or sometimes even to thousands, it is important to develop an efficient

learning algorithm for multi-class and multi-label MKL that is sublinear in the number of classes.

In multi-class and multi-label learning, each instance can be simultaneously assigned to multiple classes. A straightforward approach for multi-label MKL (ML-MKL) is to decompose a multi-label learning problem into a number of binary classification tasks using either one-vs-all or one-vs-one approach. Varma et al. discussed and compared one-vs-all and one-vs-one schemes for MKL [69]. Tang et al. [116] evaluated three different strategies for multi-label MKL based for the one-vs-all approach: (i) learning one common kernel combination shared by all classes, (ii) learning a different kernel combination for each class independently, and (iii) a hybrid approach that allows partial sharing of kernel combination among different classes. Based on their empirical study, they concluded that learning one common kernel combination shared by all classes not only is computationally efficient but also yields classification performance that is comparable to choosing different kernel combinations for different classes.

One drawback of the decomposition based approaches for multi-label learning is that they are unable to take into account the dependency between different classes or the correlation between data points. To overcome this drawback, Ji et al. [68] proposed to encode the instance-class correlation into a hypergraph, which is then used to embed the multi-label data into a lower-dimensional space. Zien et al. proposed MKL for joint feature maps $\Phi(\mathbf{x}, \mathbf{y})$ and learns a single multi-class classification function $f_{\mathbf{w},b}(\mathbf{x}, y) = \langle \mathbf{w}, \Phi(\mathbf{x}, y) \rangle + b$ from training data [87]. They formulated the problem via several optimization methods including quadratically constrained quadratic programming (QCQP) and SILP.

Mei proposed a multi-label multi-kernel transfer learning method, which uses a one-vs-all classification scheme, for protein subcellular localization [118]. Gehler et al. proposed a two-step boosting approach that requires solving SVMs separately for each kernel, similar to wrapper approaches [43]. The method they presented learns nonlinear kernel combinations, which yield promising classification performance, but also leads to a high computational load. In another nonlinear MKL method [5], group information between the classes has been incorporated to multiple

kernel learning framework (GSMKL) in order to improve the classification accuracy. Getting use of class dependencies has been shown to improve the accuracy in multi-label learning task [13], and GSMKL also gets benefit of this to yield improved classification performance with a price of increased computational load. In addition to the high computational load, another limitation of this approach is that it assumes that there is a group structure within the classes, bringing the need of effective tools to find the group structure (if exists) within the classes.

In this chapter, we develop an efficient algorithm for Multi-Label MKL (ML-MKL) that assumes all the classifiers share the same linear combination of kernels. We note that although this assumption significantly constrains the choice of kernel functions for different classes, our empirical studies with image categorization show that the classification performance is not negatively affected. A naive implementation of ML-MKL with shared kernel combination will lead to a computational cost linear in the number of classes. We alleviate this computational challenge by exploring the idea of combining worst case analysis with stochastic approximation. Our analysis reveals that the convergence rate of the proposed algorithm is $O(m^{1/3}\sqrt{\ln m})$, which is significantly better than a linear dependence on m, where m is the number of classes. Our empirical studies show that the proposed MKL algorithm yields similar performance as the state-of-the-art algorithms for ML-MKL, but with a significantly shorter running time, making it suitable for multi-label learning with a large number of classes.

3.3 Multi-label Multiple Kernel Learning (ML-MKL)

In this chapter, we use the same notation as in Chapter 2 with only a change in the notation of the label vector $\mathbf{y}$, since the focus of this chapter is multi-label MKL. We introduce $\boldsymbol{\beta} = (\beta_1, \ldots, \beta_s)$, a probability distribution, for combining base kernels. We denote by $\mathbf{K}(\boldsymbol{\beta}) = \sum_{j=1}^{s} \beta_a \mathbf{K}_j$ the combined kernel matrices. We use the domain Δ_1 for the probability distribution $\boldsymbol{\beta}$, i.e., $\Delta_1 = \{\boldsymbol{\beta} \in \mathbb{R}_+^s : \boldsymbol{\beta}^\top \mathbf{1} = 1\}$. Our goal is to learn from the training examples the optimal kernel

combination $\boldsymbol{\beta}$ for all m classes.

The simplest approach for multi-label multiple kernel learning with a shared kernel combination is to find the optimal kernel combination $\boldsymbol{\beta}$ by minimizing the sum of regularized loss functions of all m classes, leading to the following optimization problem:

$$\min_{\boldsymbol{\beta}\in\Delta_1} \min_{\{f_k\in\mathcal{H}(\boldsymbol{\beta})\}_{k=1}^m} \left\{ \sum_{k=1}^m H_k = \sum_{k=1}^m \left\{ \frac{1}{2}|f_k|^2_{\mathcal{H}(\boldsymbol{\beta})} + \sum_{i=1}^n \ell\left(y_k^i f_k(\mathbf{x^i})\right) \right\} \right\}, \tag{3.1}$$

where $\ell(z) = \max(0, 1-z)$ and $\mathcal{H}(\boldsymbol{\beta})$ is a Reproducing Kernel Hilbert Space endowed with kernel $\kappa(\mathbf{x}, \mathbf{x}'; \boldsymbol{\beta}) = \sum_{j=1}^s \beta_j \kappa_j(\mathbf{x}, \mathbf{x}')$. H_k is the regularized loss function for the kth class. It is straightforward to verify the following dual problem of Eq. (3.1):

$$\min_{\boldsymbol{\beta}\in\Delta_1} \max_{\boldsymbol{\alpha}\in\mathcal{Q}_1} \left\{ \mathcal{L}(\boldsymbol{\beta}, \boldsymbol{\alpha}) = \sum_{k=1}^m \left\{ [\alpha_\mathbf{k}]^\top \mathbf{1} - \frac{1}{2}(\alpha_\mathbf{k} \circ \mathbf{y_k})^\top \mathbf{K}(\boldsymbol{\beta})(\alpha_\mathbf{k} \circ \mathbf{y_k}) \right\} \right\}, \tag{3.2}$$

where $\mathcal{Q}_1 = \{\boldsymbol{\alpha} = (\alpha_\mathbf{1}, \ldots, \alpha_\mathbf{m}) : \alpha_\mathbf{k} \in [0, C]^n, k = 1, \ldots, m\}$. To solve the optimization problem in Eq. (3.2), we can view it as a minimization problem, i.e., $\min_{\boldsymbol{\beta}\in\Delta_1} A(\boldsymbol{\beta})$, where $A(\boldsymbol{\beta}) = \max_{\boldsymbol{\alpha}\in\mathcal{Q}_1} \mathcal{L}(\boldsymbol{\beta}, \boldsymbol{\alpha})$. We then follow the subgradient descent approach in [53] and compute the gradient of $A(\boldsymbol{\beta})$ as

$$\partial_{\beta_j} A(\boldsymbol{\beta}) = -\frac{1}{2} \sum_{k=1}^m (\alpha_\mathbf{k}(\boldsymbol{\beta}) \circ \mathbf{y})^\top \mathbf{K}_j (\alpha_\mathbf{k}(\boldsymbol{\beta}) \circ \mathbf{y_k}),$$

where $\alpha_\mathbf{k}(\boldsymbol{\beta}) = \arg\max_{\boldsymbol{\alpha}\in[0,C]^n} [\alpha_\mathbf{k}]^\top \mathbf{1} - (\alpha_\mathbf{k} \circ \mathbf{y_k})^\top \mathbf{K}(\boldsymbol{\beta})(\alpha_\mathbf{k} \circ \mathbf{y_k})$. We refer to this approach as multi-label multiple kernel learning by sum, or ML-MKL-Sum. Note that this approach is similar to the one proposed in [116]. The main computational problem with ML-MKL-Sum is that by treating every class equally, in each iteration of subgradient descent, it requires solving m kernel SVMs, making it unscalable to a very large number of classes. Below we present a formulation for multi-label MKL whose computational cost is sublinear in the number of classes.

3.3.1 A Minimax Framework for Multi-label MKL

In order to alleviate the computational difficulty arising from a large number of classes, we search for the combined kernel matrix $K(\boldsymbol{\beta})$ that minimizes the worst classification error among m classes, i.e.,

$$\min_{\boldsymbol{\beta}\in\Delta_1} \min_{\{f_k\in\mathcal{H}(\boldsymbol{\beta})\}_{k=1}^m} \max_{1\le k\le m} H_k \tag{3.3}$$

Eq. (3.3) differs from Eq. (3.1) in that it replaces $\sum_{k=1}^m H_k$ with $\max_{1\le k\le m} H_k$. The main computational advantage of using $\max_k H_k$ instead of $\sum_k H_k$ is that by using an appropriately designed method, we may be able to figure out the most difficult class, the class that yields the worst classification performance, in a few iterations, and spend most of the computational cycles on learning the optimal kernel combination for the most difficult class. In this way, we are able to achieve a running time that is sublinear in the number of classes. Below, we present an optimization strategy for Eq. (3.3) based on the idea of stochastic approximation.

A direct approach is to solve the optimization problem in Eq. (3.3) by its dual form. It is straightforward to show that dual problem of Eq. (3.3) is Eq. (3.4) (see Proposition 4 in Section A.3 for the proof).

$$\min_{\boldsymbol{\beta}\in\Delta_1} \max_{\boldsymbol{\rho}\in B} \left\{ \mathcal{L}(\boldsymbol{\beta},\boldsymbol{\rho}) = \left\{ \sum_{k=1}^m \left\{ [\boldsymbol{\rho}_k]^\top \mathbf{1} - \frac{1}{2}(\boldsymbol{\rho}_k \circ \mathbf{y_k})^\top \mathbf{K}(\boldsymbol{\beta})(\boldsymbol{\rho}_k \circ \mathbf{y_k}) \right\}^{\frac{1}{2}} \right\}^2 \right\}, \tag{3.4}$$

where

$$B = \left\{ (\boldsymbol{\rho}_1,\ldots,\boldsymbol{\rho}_m) : \boldsymbol{\rho}_k \in \mathbb{R}_+^n, k=1,\ldots,m, \boldsymbol{\rho}_k \in [0, C\lambda_k]^n \text{ s.t. } \sum_{k=1}^m \lambda_k = 1 \right\}.$$

The challenge in solving Eq. (3.4) is that the solutions $\{\boldsymbol{\rho}_1,\ldots,\boldsymbol{\rho}_m\}$ in domain B are correlated

with each other, making it impossible to solve each $\boldsymbol{\rho}_k$ independently by an off-the-shelf SVM solver. Although a gradient descent approach can be developed for optimizing Eq. (3.4), it is unable to explore the sparse structure in $\boldsymbol{\rho}_k$ making it less efficient than state-of-the-art SVM solvers. In order to effectively explore the power of off-the-shelf SVM solvers, we rewrite Eq. (3.3) as follows

$$\min_{\boldsymbol{\beta}\in\Delta_1}\max_{\boldsymbol{\gamma}\in\Gamma}\left\{\mathcal{L}(\boldsymbol{\beta},\gamma)=\max_{\alpha\in\mathcal{Q}_1}\sum_{k=1}^{m}\gamma_k\left\{\boldsymbol{\alpha}_k^\top\mathbf{1}-\frac{1}{2}(\boldsymbol{\alpha}_k\circ\mathbf{y}_k)^\top\mathbf{K}(\boldsymbol{\beta})(\boldsymbol{\alpha}_k\circ\mathbf{y}_k)\right\}\right\}, \tag{3.5}$$

where $\Gamma=\{(\gamma_1,\ldots,\gamma_m)\in\mathbb{R}_+^m:\boldsymbol{\gamma}^\top\mathbf{1}=1\}$. In Eq. (3.5), we replace $\max_{1\leq k\leq m}$ with $\max_{\boldsymbol{\gamma}\in\Gamma}$. The advantage of using Eq. (3.5) is that we can resort to a SVM solver to efficiently find $\boldsymbol{\alpha}_k$ for a given combination of kernels $\mathbf{K}(\boldsymbol{\beta})$.

Given Eq. (3.5), we develop a subgradient descent approach for solving the optimization problem. In particular, in each iteration of subgradient descent, we compute the gradient $\mathcal{L}(\boldsymbol{\beta},\boldsymbol{\gamma})$ with respect to $\boldsymbol{\beta}$ and $\boldsymbol{\gamma}$ as follows

$$\begin{aligned}\nabla_{\beta_j}\mathcal{L}(\boldsymbol{\beta},\gamma)&=-\frac{1}{2}\sum_{k=1}^{m}\gamma_k(\boldsymbol{\alpha}_k\circ\mathbf{y}_k)^\top\mathbf{K}_j(\boldsymbol{\alpha}_k\circ\mathbf{y}_k),\\ \nabla_{\gamma_k}\mathcal{L}(\boldsymbol{\beta},\gamma)&=[\boldsymbol{\alpha}_k]^\top\mathbf{1}-\frac{1}{2}(\boldsymbol{\alpha}_k\circ\mathbf{y}_k)^\top\mathbf{K}(\boldsymbol{\beta})(\boldsymbol{\alpha}_k\circ\mathbf{y}_k),\end{aligned} \tag{3.6}$$

where $\boldsymbol{\alpha}_k=\arg\max_{\boldsymbol{\alpha}\in[0,C]^n}\boldsymbol{\alpha}^\top\mathbf{1}-(\boldsymbol{\alpha}\circ\mathbf{y}_k)^\top\mathbf{K}(\boldsymbol{\beta})(\boldsymbol{\alpha}\circ\mathbf{y}_k)/2$, i.e., a SVM solution to the combined kernel $\mathbf{K}(\boldsymbol{\beta})$. Following the mirror prox descent method [119], we define potential functions $\Phi_{\boldsymbol{\beta}}=\frac{\eta_\beta}{\eta_\gamma}\sum_{j=1}^{s}\beta_j\ln\beta_j$ for $\boldsymbol{\beta}$ and $\Phi_\gamma=\sum_{k=1}^{m}\gamma_k\ln\gamma_k$ for $\boldsymbol{\gamma}$, and have the following equations for updating $\boldsymbol{\beta}^t$ and $\boldsymbol{\gamma}^t$

$$\begin{aligned}\boldsymbol{\beta}_j^{t+1}&=\frac{\beta_j^t}{Z_\beta^t}\exp(-\eta_\beta\nabla_{\beta_j}\mathcal{L}(\boldsymbol{\beta}^t,\boldsymbol{\gamma}^t)),\\ \gamma_k^{t+1}&=\frac{\gamma_k^t}{Z_\gamma^t}\exp(-\eta_\gamma\nabla_{\gamma_k}\mathcal{L}(\boldsymbol{\beta}^t,\boldsymbol{\gamma}^t)),\end{aligned} \tag{3.7}$$

where Z_β^t and Z_γ^t are normalization factors that ensure ${\boldsymbol{\beta}^t}^\top\mathbf{1}={\boldsymbol{\gamma}^t}^\top\mathbf{1}=1$. $\eta_\beta>0$ and $\eta_\gamma>0$ are

the step sizes for optimizing $\boldsymbol{\beta}$ and $\boldsymbol{\gamma}$, respectively.

Unfortunately, the algorithm described above shares the same shortcoming as the other approaches for multiple label multiple kernel learning: it requires solving m SVM problems in each iteration; therefore, its computational complexity is linear in the number of classes. To alleviate this problem, we modify the above algorithm by introducing the stochastic approximation method. In particular, in each iteration t, instead of computing the full gradients that requires solving m SVMs. We sample one classification task according to the multinomial distribution $Multi(\gamma_1^t, \ldots, \gamma_m^t)$. Let a_t be the index of the sampled classification task. Using the sampled task a_t, we estimate the gradient of $\mathcal{L}(\boldsymbol{\beta}, \boldsymbol{\gamma})$ with respect to β_j and γ_k, denoted by $\widehat{g}_j^{\beta}(\boldsymbol{\beta}^t, \boldsymbol{\gamma}^t)$ and $\widehat{g}_k^{\gamma}(\boldsymbol{\beta}^t, \boldsymbol{\gamma}^t)$, as follows

$$\widehat{g}_j^{p}(\mathbf{p}_t, \gamma_t) = -\frac{1}{2}(\boldsymbol{\alpha}^{a_t} \circ \mathbf{y}^{a_t})^{\top} \mathbf{K_j} (\boldsymbol{\alpha}^{a_t} \circ \mathbf{y}^{a_t}), \tag{3.8}$$

$$\widehat{g}_k^{\gamma}(\boldsymbol{\beta}^t, \boldsymbol{\gamma}^t) = \begin{cases} 0 & k \neq a_t \\ \frac{1}{\gamma_k} \left({\boldsymbol{\alpha}^k}^{\top} \mathbf{1} - \frac{1}{2} (\boldsymbol{\alpha}^k \circ \mathbf{y}^k)^{\top} \mathbf{K}(\boldsymbol{\beta}) (\boldsymbol{\alpha}^k \circ \mathbf{y}^k) \right) & k = a_t \end{cases}. \tag{3.9}$$

The computation of $\widehat{g}_j^{\beta}(\boldsymbol{\beta}^t, \boldsymbol{\gamma}^t)$ and $\widehat{g}_k^{\gamma}(\boldsymbol{\beta}^t, \boldsymbol{\gamma}^t)$ only requires $\boldsymbol{\alpha}^{a_t}$; therefore, it only needs to solve one SVM problem, instead of m SVMs. The key property of the estimated gradients in Eqs. (3.8) and (3.9) is that their expectations are equal to the true gradients, as summarized by Proposition 1. This property is the key to the correctness of our algorithm.

Proposition 1. *We have*

$$\mathrm{E}_t[\widehat{g}_j^{\beta}(\boldsymbol{\beta}^t, \boldsymbol{\gamma}^t)] = \nabla_{\beta_j} \mathcal{L}(\boldsymbol{\beta}^t, \boldsymbol{\gamma}^t), \ \mathrm{E}_t[\widehat{g}_k^{\gamma}(\boldsymbol{\beta}^t, \boldsymbol{\gamma}^t)] = \nabla_{\gamma_k} \mathcal{L}(\boldsymbol{\beta}^t, \boldsymbol{\gamma}^t),$$

where $E_t[\cdot]$ stands for the expectation over the randomly sampled task a_t.

Given the estimated gradients, we will follow Eq. (A.12) for updating $\boldsymbol{\beta}$ and $\boldsymbol{\gamma}$ in each iteration. Since $\widehat{g}_k^{\gamma}(\boldsymbol{\beta}^t, \boldsymbol{\gamma}^t)$ is proportional to $1/\boldsymbol{\gamma}^t$, to ensure the norm of $\widehat{g}_k^{\gamma}(\boldsymbol{\beta}^t, \boldsymbol{\gamma}^t)$ to be bounded, we need to smooth $\boldsymbol{\gamma}^{t+1}$. In order to have a smoothing effect, without modifying $\boldsymbol{\gamma}^{t+1}$, we will sample

directly from $\hat{\boldsymbol{\gamma}}^{t+1}$,

$$\forall \boldsymbol{\gamma} \in \Gamma, \exists \hat{\boldsymbol{\gamma}} \in \hat{\Gamma}, \text{ s.t. } \hat{\gamma}_k^{t+1} \leftarrow \gamma_k^{t+1}(1-\delta) + \frac{\delta}{m}, k = 1, \ldots, m,$$

where $\delta > 0$ is a small probability mass used for smoothing and

$$\hat{\Gamma} = \left\{ \hat{\boldsymbol{\gamma}}^\top \mathbf{1} = 1, \hat{\gamma}_k \geq \frac{\delta}{m}, k = 1, \ldots, m \right\}.$$

We refer to this algorithm as multi-label multiple kernel learning by stochastic approximation, or ML-MKL-SA for short. Algorithm 3 gives the detailed description.

3.3.2 Convergence Analysis

Since Eq. (3.5) is a convex-concave optimization problem, we introduce the following citation for measuring the quality of a solution $(\boldsymbol{\beta}, \boldsymbol{\gamma})$

$$\bar{\Delta}(\boldsymbol{\beta}, \boldsymbol{\gamma}) = \max_{\boldsymbol{\gamma}' \in \Gamma} \mathcal{L}(\boldsymbol{\beta}, \boldsymbol{\gamma}') - \min_{\boldsymbol{\beta}' \in \Delta_1} \mathcal{L}(\boldsymbol{\beta}', \boldsymbol{\gamma}). \tag{3.11}$$

We denote by $(\boldsymbol{\beta}_*, \boldsymbol{\gamma}_*)$ the optimal solution to Eq. (3.5).

Proposition 2. *We have the following properties for* $\bar{\Delta}(\boldsymbol{\beta}, \boldsymbol{\gamma})$

1. $\bar{\Delta}(\boldsymbol{\beta}, \boldsymbol{\gamma}) \geq 0$ *for any solution* $\boldsymbol{\beta} \in \Delta_1$ *and* $\boldsymbol{\gamma} \in \Gamma$
2. $\bar{\Delta}\left(\boldsymbol{\beta}_*, \boldsymbol{\gamma}_*\right) = 0$
3. $\Delta(\boldsymbol{\beta}, \boldsymbol{\gamma})$ *is jointly convex in both* $\boldsymbol{\beta}$ *and* $\boldsymbol{\gamma}$

We have the following theorem for the convergence rate for Algorithm 3. The detailed proof can be found in Section A.3.

Theorem 1. *After running Algorithm 3 over* T *iterations, we have the following inequality for the*

solution $\bar{\boldsymbol{\beta}}$ and $\bar{\boldsymbol{\gamma}}$ obtained by Algorithm 3

$$\mathrm{E}\left[\bar{\Delta}\left(\bar{\boldsymbol{\beta}},\bar{\boldsymbol{\gamma}}\right)\right] \leq \frac{1}{\eta_\gamma T}\left(\ln m + \ln s\right) + \eta_\gamma\left(d\frac{m^2}{2\delta^2}\lambda_0^2 n^2 C^4 + n^2 C^2\right),$$

where d is a constant term, $\mathrm{E}[\cdot]$ stands for the expectation over the sampled task indices of all iterations, and $\lambda_0 = \max_{1\leq j\leq s}\lambda_{\max}(\mathbf{K}_j)$, where $\lambda_{\max}(\mathbf{Z})$ stands for the maximum eigenvalue of matrix $\mathbf{Z}$.

Corollary 2. *With $\delta = m^{\frac{2}{3}}$ and $\eta_\gamma = \frac{1}{n}m^{-\frac{1}{3}}\sqrt{(\ln m)/T}$, after running Algorithm 1 (on the original paper) over T iterations, we have $\mathrm{E}[\Delta(\bar{\boldsymbol{\beta}},\bar{\boldsymbol{\gamma}})] \leq O(nm^{1/3}\sqrt{(\ln m)/T})$ in terms of m,n and T.*

Since we only need to solve one kernel SVM at each iteration, we have the computational complexity for the proposed algorithm on the order of $O(m^{1/3}\sqrt{(\ln m)/T})$, sublinear in the number of classes m.

3.4 Experimental Results

In this section, we empirically evaluate the proposed multi-label multiple kernel learning algorithm by demonstrating its efficiency and effectiveness on the image categorization task.

3.4.1 Data Sets

Following the MKL experiments in Chapter 2, we use the same three benchmark data sets and the same base kernels as in this Chapter: Caltech 101 [3], Pascal VOC 2007 [94], and a subset of ImageNet. All the experiments conducted in this chapter are repeated five times, each with an independent random partition of training and testing data. Mean average precision scores along with the associated standard deviations are reported.

3.4.2 Baseline Methods

We compare four MKL methods and the average kernel baseline. The MKL baselines can be categorized into two groups. The first group is the one-vs-all MKL framework which requires solving one MKL problem for each class separately. For this group, we use two base MKL solvers that are shown to be the most efficient L_1-MKL methods in Chapter 2 : (i) MKL-SIP, a Semi-Infinite Programming (SIP) based method for MKL, [71] and (ii) MKL-Level, an extended level based method for MKL, [52]. We also use MKL-SIP-L_2 to include a non-sparse MKL solver into the comparison. The second group of methods requires learning a single kernel combination simultaneously for all classes. The two baseline methods that fall into this group are: (i) ML-MKL-Sum which learns a kernel combination shared by all classes as described in Section 3.3 using the optimization method in [116], (ii) the proposed ML-MKL-SA method.

ant butterfly ceiling fan chair

Figure 3.1: For the 4 classes (*ant, butterfly, ceiling fan, chair*) taken from the Caltech 101 data set, the first row gives images which produced false negatives for the single kernel baseline and true positives for ML-MKL-SA baseline. The second row gives images which produced false positives for the single kernel baseline and true negatives for the ML-MKL-SA baseline for the corresponding classes.

3.4.3 Implementation

The experiments with varied numbers of instances on the ImageNet data set were performed on a cluster of Sun Fire X4600 M2 nodes, each with 256 GB of RAM and 32 AMD Opteron cores, due to a need of high RAM capacity (over 100 GB). All other experiments were run on a different cluster, where each node has two four-core Intel Xeon E5620s at 2.4 GHz with 24 GB of RAM. We pre-compute all the kernel matrices and load the computed kernel matrices into the memory. This allows us to avoid re-computing and loading kernel matrices at each iteration of optimization.

All the baseline methods are coded in MATLAB For all the wrapper methods for MKL, LIBSVM [107] is used as off-the-shelf SVM solver. For MKL-SIP and MKL-Level, MOSEK [89] is used to solve the related optimization problems, as suggested in [52].

The same stopping criteria is applied to all the MKL algorithms when applicable. All the algorithms terminate when: (i) the relative change in the duality gap falls below a threshold ($1 - \frac{\Delta_t}{\Delta_{t-1}} < 10^2$), (ii) the change in the cost function falls below a threshold (10^{-3}), (iii) the difference in the kernel coefficients $\boldsymbol{\beta}$ between two consecutive iterations is small (i.e., $||\boldsymbol{\beta}^t - \boldsymbol{\beta}^{t-1}||_\infty < 10^{-4}$), and (iv) the maximum number of iterations is reached. A 2-fold cross-validation is applied to select the value of the regularization parameter $C \in \{10^{-2}, 10^{-1}, \ldots, 10^4\}$. The bandwidth of the RBF kernel is set to the average pair-wise χ^2 distances between image pairs.

Unless stated, the smoothing parameter δ is set to be 0.2 for the proposed method. For simplicity we take $\eta = \eta_\beta = \eta_\gamma$ in all the following experiments. Step size η is chosen as 0.01 for the Caltech 101 data set, 0.001 for the VOC 2007 and ImageNet data sets in order to achieve the best computational efficiency.

3.4.4 Classification Performance

To evaluate the effectiveness of different algorithms for multi-label multiple kernel learning, we report the category based mean averaged precision (MAP) over all the classes. We evaluate the

bird potted plant dining table train

Figure 3.2: For the 4 classes (*bird, potted plant, dining table, train*) taken from the VOC 2007 data set, the first row gives images which produced false negatives for the single kernel baseline and true positives by the GMKL baseline. The second row gives images which produced false positives for the single kernel baseline and true negatives for the ML-MKL-SA method for the corresponding classes.

efficiency of algorithms by their running times (seconds) for training.

Table 3.1 summarizes the classification accuracies (MAP) of all the baseline methods over the Caltech 101 data sets under three settings with 10, 20, and 30 training instances per class. MKL-SIP-L_2 and average kernel baselines yield the best performance for the first two settings, whereas MKL solvers with L_1 norm are superior for the last setting, where the number of training instances per class is 30. MKL-L_1 methods give sparse solutions by eliminating irrelevant base kernels.

Table 3.1: Classification results (MAP) for the Caltech 101 data set. We report the average values over five random splits and the associated standard deviation.

	Number of training instances per class		
Baseline	10	20	30
Average	**59.0** $\pm$ **0.7**	**69.7** $\pm$ **0.6**	77.2 $\pm$ 0.5
MKL-Level	54.7 $\pm$ 1.0	63.4 $\pm$ 0.6	84.4 $\pm$ 0.4
MKL-SIP-L_1	53.8 $\pm$ 0.6	63.8 $\pm$ 0.9	83.9 $\pm$ 0.7
MKL-SIP-L_2	**60.1** $\pm$ **0.6**	**70.7** $\pm$ **1.0**	79.1 $\pm$ 0.6
ML-MKL-Sum	55.1$\pm$ 1.3	65.0 $\pm$ 0.7	**85.6** $\pm$ **0.7**
ML-MKL-SA	54.5$\pm$ 0.7	66.1 $\pm$ 0.9	**85.3**$\pm$ **0.8**

Table 3.2: Classification results (MAP) for the VOC 2007 data set. We report the average values over five random splits and the associated standard deviation.

	Percentage of the samples used for training				
baseline	1%	5%	25%	50%	75%
Average	21.9 ± 0.5	42.4 ± 0.3	48.2 ± 0.8	54.5 ± 0.8	57.5 ± 0.8
MKL-Level	23.4 ± 0.6	**44.4 ± 0.4**	**51.5 ± 0.5**	**57.1± 0.6**	59.6 ± 0.9
MKL-SIP-L_1	22.6 ± 1.0	**44.2 ± 0.3**	**51.2 ± 0.33**	**56.6 ± 0.5**	59.5± 0.9
MKL-SIP-L_2	22.7 ± 0.4	42.6 ± 0.2	49.8 ± 0.2	**57.3 ± 0.2**	60.6± 0.5
ML-MKL-Sum	24.1 ± 0.4	**43.5 ± 0.5**	50.1 ± 0.4	55.8 ± 0.1	58.8±0.2
ML-MKL-SA	24.6 ± 0.9	**44.1 ± 0.6**	**50.6 ± 0.4**	56.1 ± 0.2	58.9±0.4

However, as discussed in Chapter 2, when the number of training examples is very small, it is difficult to determine the subset of kernels that are irrelevant to a given task. This is why MKL-L_1 methods give better results than MKL-L_2 methods on the Caltech 101 data set as the number of training instances increases.

Although the two multi-label MKL baselines, namely ML-MKL-Sum and ML-MKL-SA, are originally proposed as efficient approximations to one-versus-all MKL framework, they match and sometimes even outperform the one-vs-all MKL methods, MKL-SIP and Level-MKL, that learn one kernel combination for each class. These results justify the assumption of using the same kernel combination for all the classes for the Caltech 101 data set. Note that the average kernel baseline (AVG), which is similar in that it uses the same kernel combination for all classes, yields reasonable performance, although its classification performance is significantly worse than the proposed approach ML-MKL-SA when there is a sufficient number of training instances (30 instances per class for the Caltech 101 data set).

We provide some example images from the Caltech 101 data set in Figure 3.1 to visualize the advantage MKL brings over using a single kernel. For the 4 classes (ant, butterfly, ceiling fan, chair) taken from the Caltech 101 data set, the first row gives images which produced true positives for the ML-MKL-SA baseline and false negatives when a single kernel (the best performing base

kernel) is used. On the other hand, the second row gives images which produced false positives for the single kernel case and true negatives for the ML-MKL-SA baseline for the corresponding classes. Note the level of similarity in the shapes of each image on the same column, which is the possible cause of the errors for the single kernel case. On the other hand, by using different image representations, MKL avoids these errors on these sample images.

Table 3.2 summarizes the classification accuracies (MAP) for all the baseline methods on the VOC 2007 data set under five different settings, where 1%, 5%, 25%, 50%, and 75% of the whole data set is used as the training set. Table 3.2 confirms the conclusions that are drawn from Table 3.1: all the MKL methods, including ML-MKL-Sum and ML-MKL-SA outperform average kernel baseline as the number of training instances increase (for all settings except case-1%). The difference between the Caltech 101 and VOC 2007 results is that we do not see a significant performance difference between MKL-L_1 and MKL-L_2 methods. As discussed in Chapter 2, this is because the number of base kernels is smaller for the VOC 2007 experiments. Finally, we see that ML-MKL-Sum and ML-MKL-SA yield very close results compared to other MKL baselines, despite learning one shared kernel combination for all classes.

We also provide some example images from the VOC 2007 data set in Figure 3.2 to visualize the strength of MKL. We take four object categories and two different test images from each category to test. The first row gives images which produced true positives for the ML-MKL-SA baseline and false negatives when a single kernel (the best performing base kernel) is used. The second row gives images which produced false positives for the single kernel case and true negatives for the ML-MKL-SA baseline for the corresponding classes. These examples demonstrate that MKL methods are able to avoid false positives and negatives by successfully combine several image representations.

Table 3.3: Training time (seconds) for the Caltech 101 data set. We report the average values over five random splits and the associated standard deviation.

	Number of training instances per class		
Baseline	10	20	30
level-MKL	816.1± 125.6	3570± 519.0	6456.6± 664.2
MKL-SIP-L_1	550.8 ±91.8	2233.8±871.5	4518.6 ± 501.2
MKL-SIP-L_2	387.6 ±72.4	1275.0± 201.6	3100.8± 314.6
ML-MKL-Sum	302.7 ± 4.8	1053.8± 201.3	3817.9 ± 308.1
ML-MKL-SA	**119.2 ± 0.9**	**471.3 ± 16.9**	**1140.4 ± 276.5**

3.4.5 Training Time

We provide Tables 3.3 and 3.4 to compare the running times of the MKL baseline methods. Observe that ML-MKL-SA and ML-MKL-Sum are in general more efficient than the other MKL methods in the Caltech 101 experiments. This is not surprising as ML-MKL-SA and ML-MKL-Sum compute a single kernel combination for all classes. However, note that MKL-SIP-L_2 is faster than ML-MKL-Sum when the number of training instances is 30 per class for the Caltech 101 data set. This is because of the fast convergence of MKL-L_2 problem (see Chapter 2 for details). Moreover, we see that MKL-SIP-L_2 is faster than ML-MKL-Sum in most of the settings. The main reason for this is that, in addition to the fast convergence of MKL-SIP-L_2, the number of kernels and classes is smaller in the VOC 2007 data set. However, based on these observations, we expect ML-MKL-Sum to become faster as the number of classes and the number of kernels increase, since MKL-L_1 formulation often provides sparse solutions, which would significantly cut down the time spent on kernel computations.

The main advantage of the proposed algorithm is its computational efficiency. From Tables 3.3 and 3.4 we see that the proposed method requires less training time compared to the other baselines while providing comparable classification performance. Clearly, for the data sets with a high number of categories, the two methods that learn one shared kernel combination for all labels (ML-MKL-SA and ML-MKL-Sum) would be computationally more efficient than the methods that

Table 3.4: Training time (seconds) for the VOC 2007 data set. We report the average values over five random splits and the associated standard deviation.

Baseline	Percentage of the samples used for training				
	1%	5%	25%	50%	75%
level-MKL	4.5±0.5	43.3±7.1	802± 113.2	4332.6± 587.3	5946± 950.1
MKL-SIP-L_1	6.4±3.4	47.9±10.6	692 ± 67.8	4396.8±606.7	6180 ± 940.2
MKL-SIP-L_2	16.4±2.3	**34.3±7.4**	**192±21.3**	**706± 178.3**	1686± 246.5
ML-MKL-Sum	2.5± 0.3	57.4± 9.1	372.3± 26.6	2162.1± 175.3	3983 ± 402.2
ML-MKL-SA	**1.2± 0.3**	**39.8± 4.8**	**234.1± 21.1**	**886.5± 101.7**	**1224.3± 136.2**

learn a kernel combination separately for each class. In addition to this, the proposed method brings further improvement in efficiency compared to ML-MKL-Sum. The reduction in computation time is more significant for the Caltech 101 data set compared to the VOC 2007 data set. This is because the proposed algorithm employs an SVM solver for only one class per an iteration whereas ML-MKL-Sum has to train SVM solvers separately for all classes at each iteration. Since Caltech 101 has a larger number of classes, the proposed method shows a greater advantage for the Caltech 101 data set.

Figure 3.6 shows the change in the kernel weights over time for the proposed method (ML-MKL-SA) and Figures 3.3, 3.4, and 3.5 show the change in the kernel weights for three other baseline methods (ML-MKL-Sum, MKL-Level, and MKL-SIP-L_1) on the Caltech 101 data set with 30 training instances per class. We observe that, overall, ML-MKL-SA shares a similar pattern as Level-MKL in the evolution curves of kernel weights, but is much faster. We also have very similar curves when comparing MKL-SIP-L_1 and ML-MKL-Sum, as expected, since these two baselines use the same solver. When comparing ML-MKL-Sum and ML-MKL-SA, which are significantly more efficient than the other two baselines, we see that the kernel weights learned by ML-MKL-Sum vary significantly, particularly at the beginning of the learning process, making it a less stable algorithm than the proposed algorithm ML-MKL-SA.

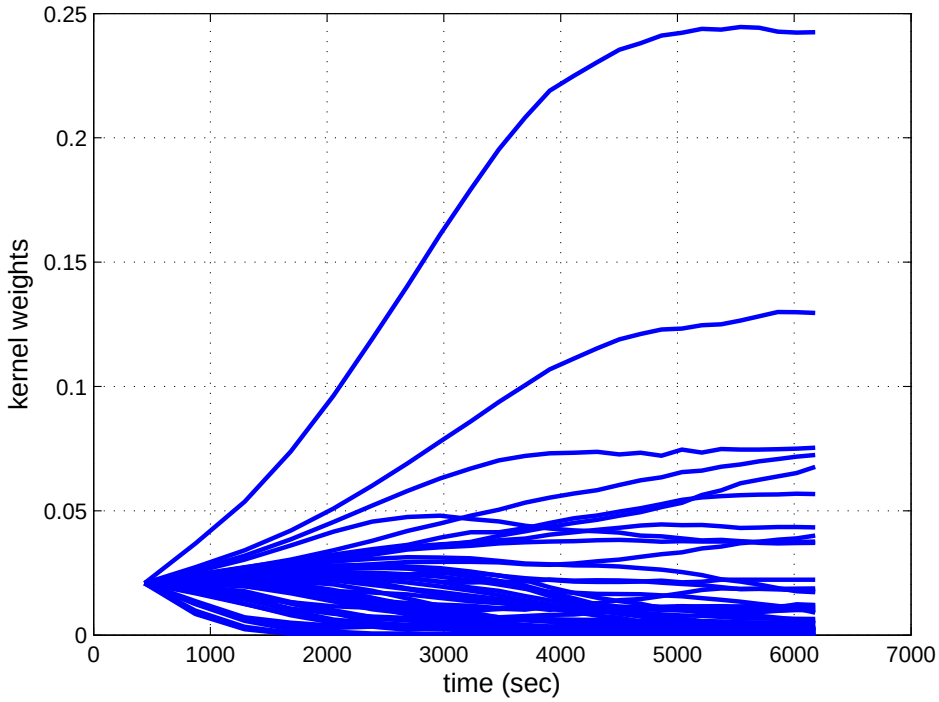

Figure 3.3: The evolution of kernel weights computed by the MKL-Level method over time for the Caltech 101 data set with 30 training instances per class.

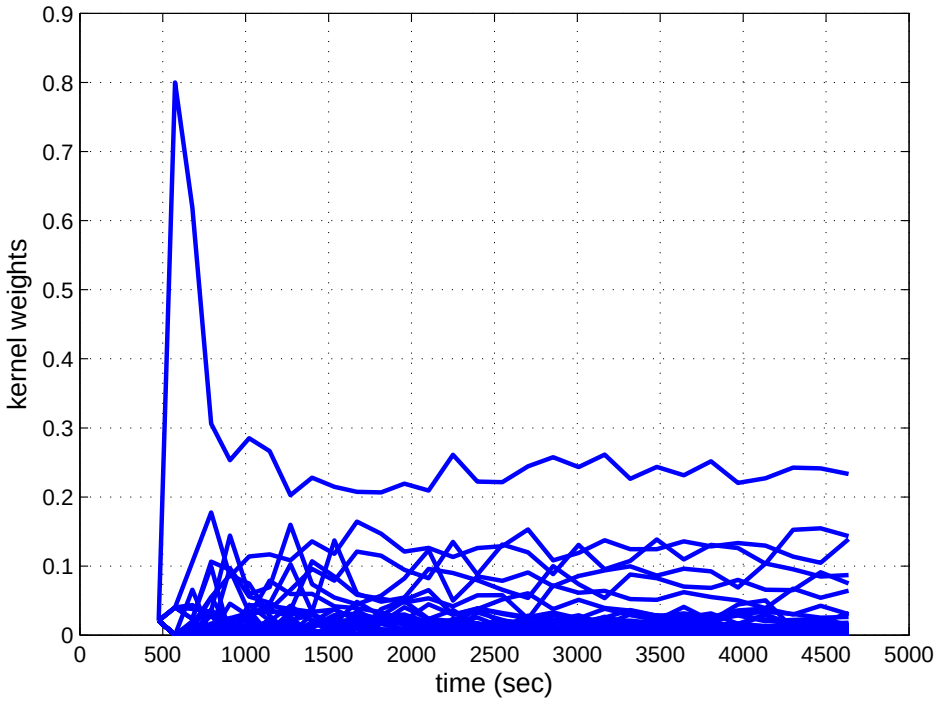

Figure 3.4: The evolution of kernel weights computed by the MKL-SIP-L_1 method over time for the Caltech 101 data set with 30 training instances per class.

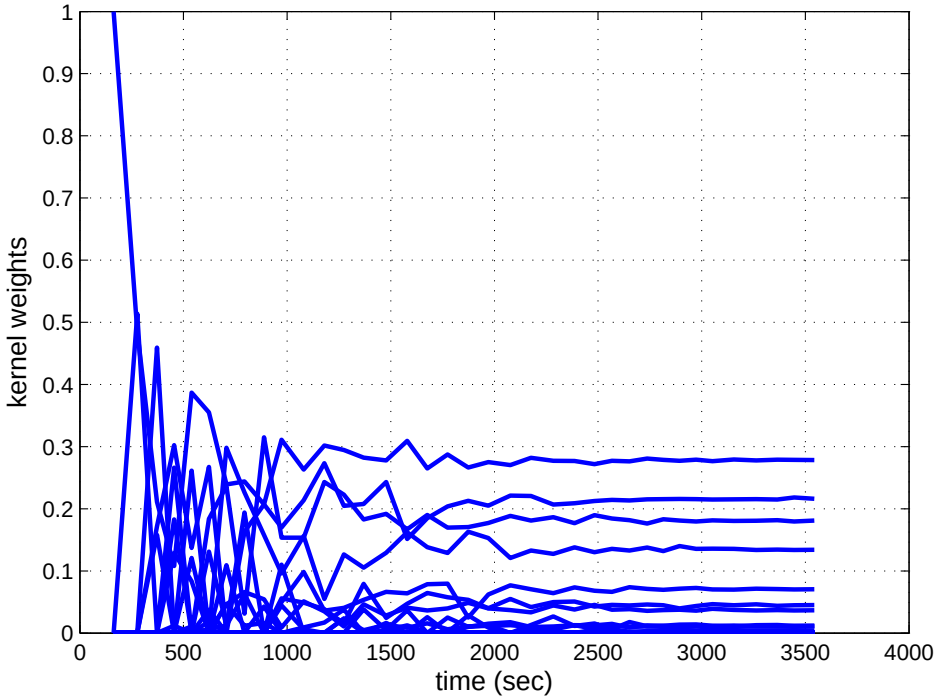

Figure 3.5: The evolution of kernel weights computed by the ML-MKL-Sum method over time for the Caltech 101 data set with 30 training instances per class.

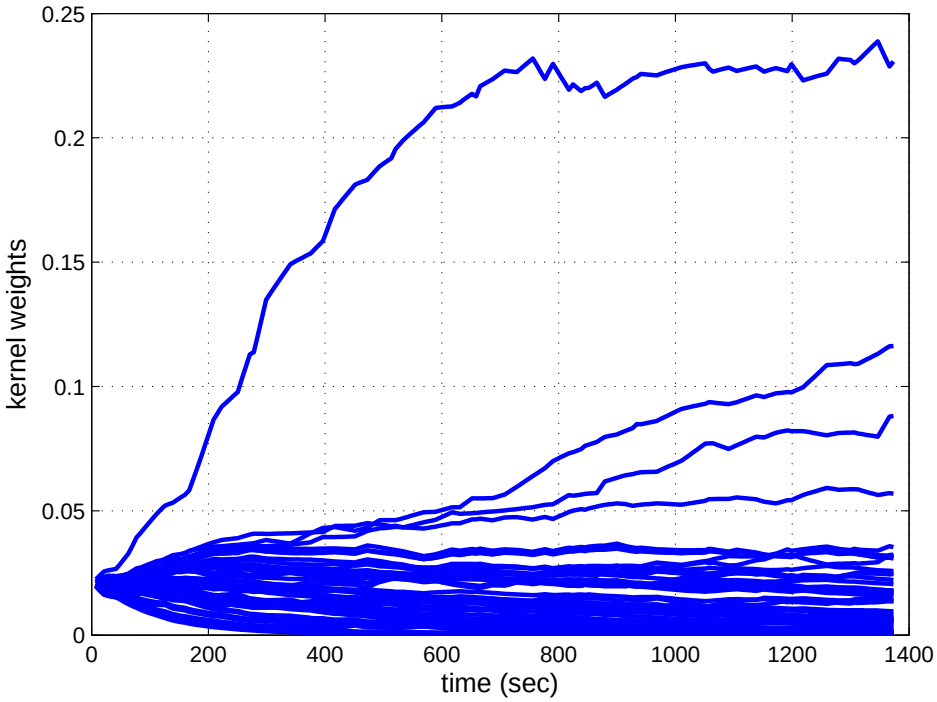

Figure 3.6: The evolution of kernel weights computed by the ML-MKL-SA method over time for the Caltech 101 data set with 30 training instances per class.

3.4.6 Sensitivity to Parameters

To evaluate the sensitivity of the proposed method to parameters δ, η_β and η_γ, we conducted experiments with varied values for these three parameters. Figure 3.7 shows how the classification performance (MAP) of the proposed algorithm changes over iterations on Caltech 101 (30 training instances per class) using six different values of δ: $\{0, 0.2, 0.4, 0.6, 0.8, 1\}$. We observe that the final classification accuracy is comparable for different values of δ, demonstrating the robustness of the proposed method to the choice of δ. However, we also note that the extreme case where $\delta = 0$ gives the worst performance, indicating the importance of adding the uniform sampling component for an increased stability.

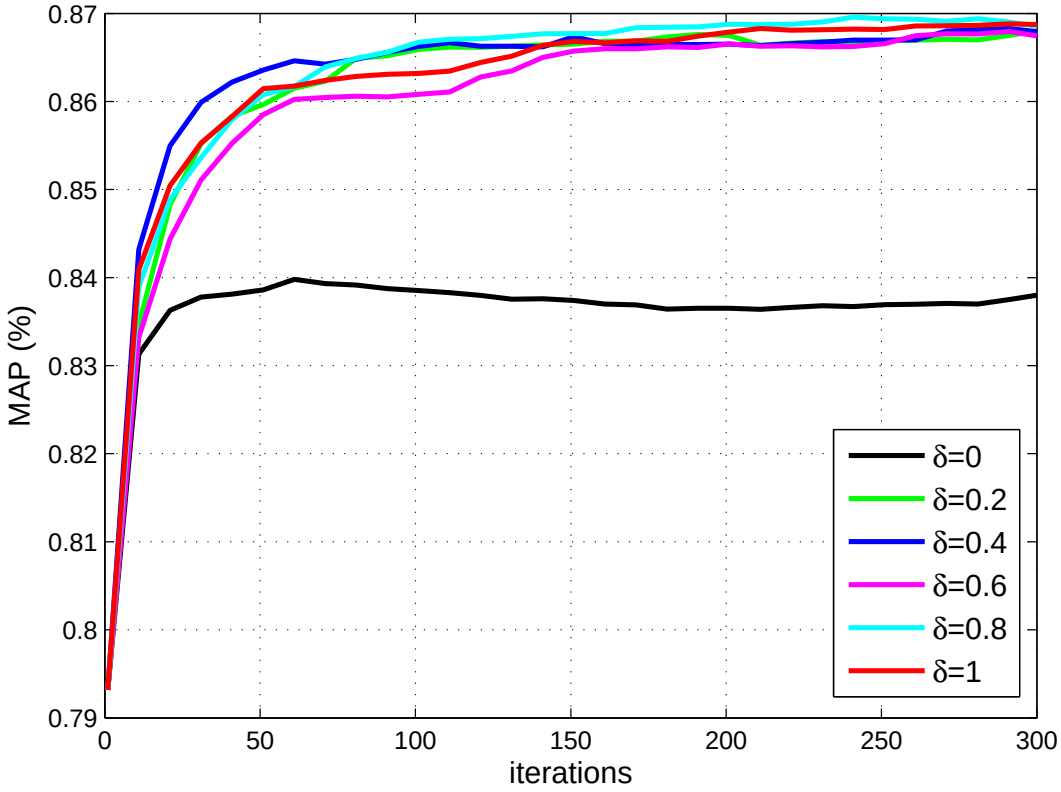

Figure 3.7: Classification performance (MAP) of the proposed algorithm ML-MKL-SA on Caltech 101 with 30 training instances per class using different values of δ (for $\eta_\beta = \eta_\gamma = 0.01$).

Figure 3.8 shows the change of classification performance (MAP) for three different values of η_β for a fixed η_γ whereas Figure 3.9 shows the change of classification performance (MAP) for

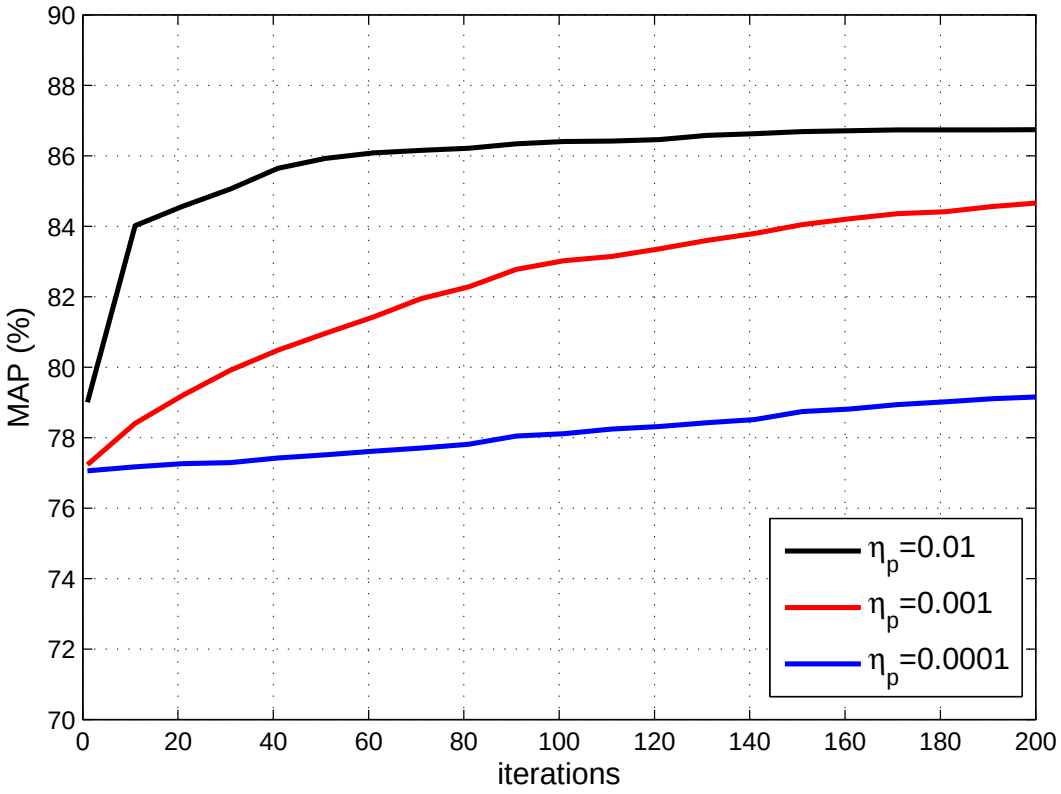

Figure 3.8: Classification performance (MAP) of the proposed algorithm ML-MKL-SA on Caltech 101 with 30 training instances per class using different values of η_β (for $\eta_\gamma = 0.0001$ and $\delta = 0.2$).

three different values of η_γ for a fixed η_β, when 30 samples per class are used from the Caltech 101 data set. Based on these plots we observe that a change in the value of η_β is more likely to have a greater impact on the convergence speed than a change in the η_γ value. Particularly, we see that $\eta_\gamma = 0.01$ and $\eta_\gamma = 0.001$ produce very similar plots. This result demonstrates that the proposed algorithm is in general insensitive to the choice of the step size η_γ. On the other hand, a more careful selection still needs to be done η_β in order to avoid slow convergence.

3.4.7 Large-scale MKL on ImageNet

To evaluate the scalability of MKL, we perform experiments on the subset of ImageNet consisting of $81,738$ images. Figure 3.10 shows the classification performance of ML-MKL-SA and

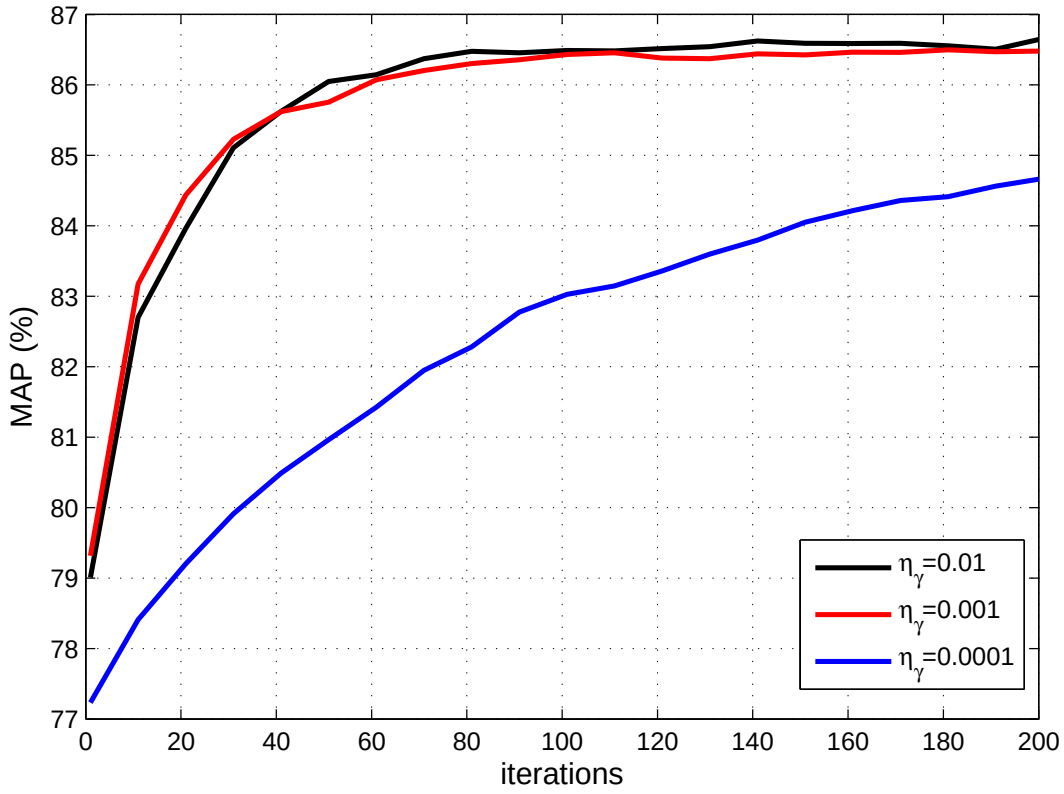

Figure 3.9: Classification performance (MAP) of the proposed algorithm ML-MKL-SA on Caltech 101 with 30 training instances per class using different values of η_γ ($\eta_\beta = 0.0001$ and $\delta = 0.2$).

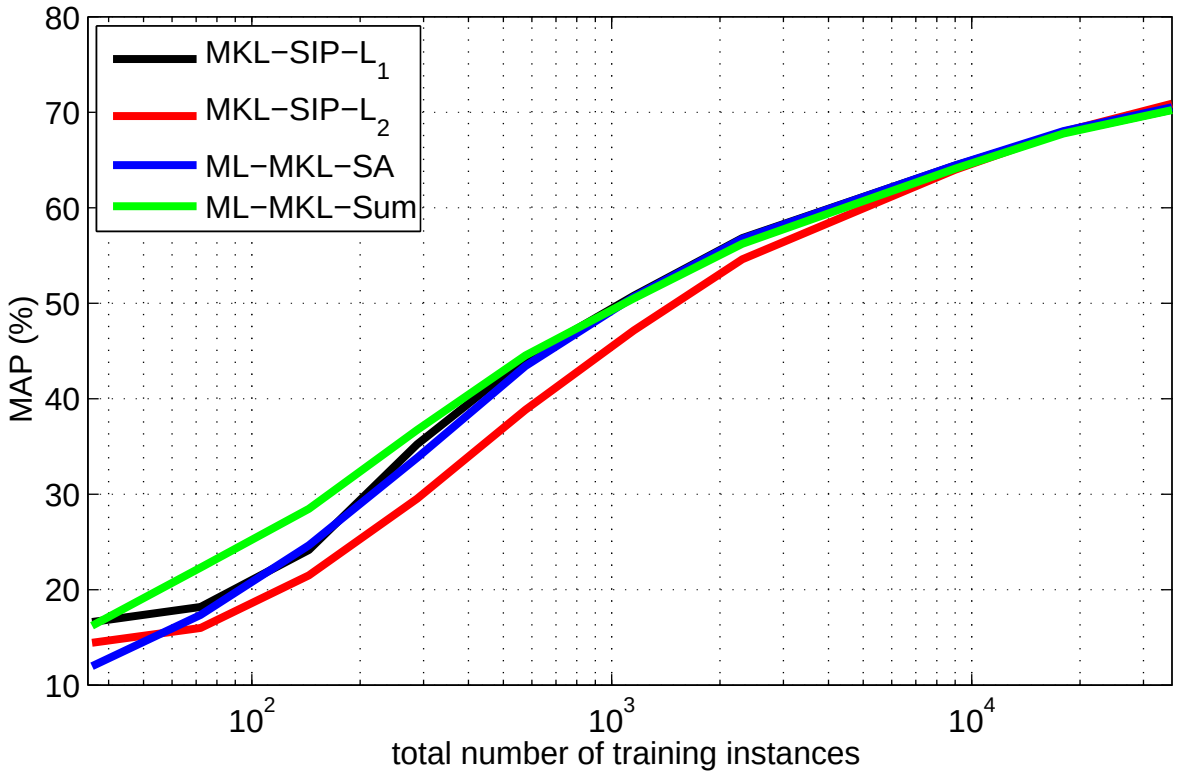

Figure 3.10: Comparison of the mean average precision scores for different training set sizes for the ImageNet data set.

the other baseline methods with the number of training images per class varied in powers of 2: $(2^1, 2^2, \ldots, 2^{11})$. We used MKL-SIP for both L_1 and L_2 norm MKL. Similar to the experimental results for Caltech 101 and VOC 2007, we observe that ML-MKL-SA and ML-MKL-Sum give comparable performance to the MKL solvers that learn a separate kernel combination for each class. In fact, for the settings with smaller number of instances (100 to 18000) ML-MKL-SA outperforms MKL-L_2 whereas ML-MKL-Sum outperforms both MKL-L_1 and MKL-L_2. However, the difference between the baseline performances starts to diminish when the number of training examples is increased over 256 per class. As discussed in Chapter 2, this is because all the 10 kernels constructed for the ImageNet data set are strong kernels and provide informative features for image categorization. In other words, the main strength of MKL-L_1, which is being able to remove irrelevant or weak kernels, does not bring any advantage.

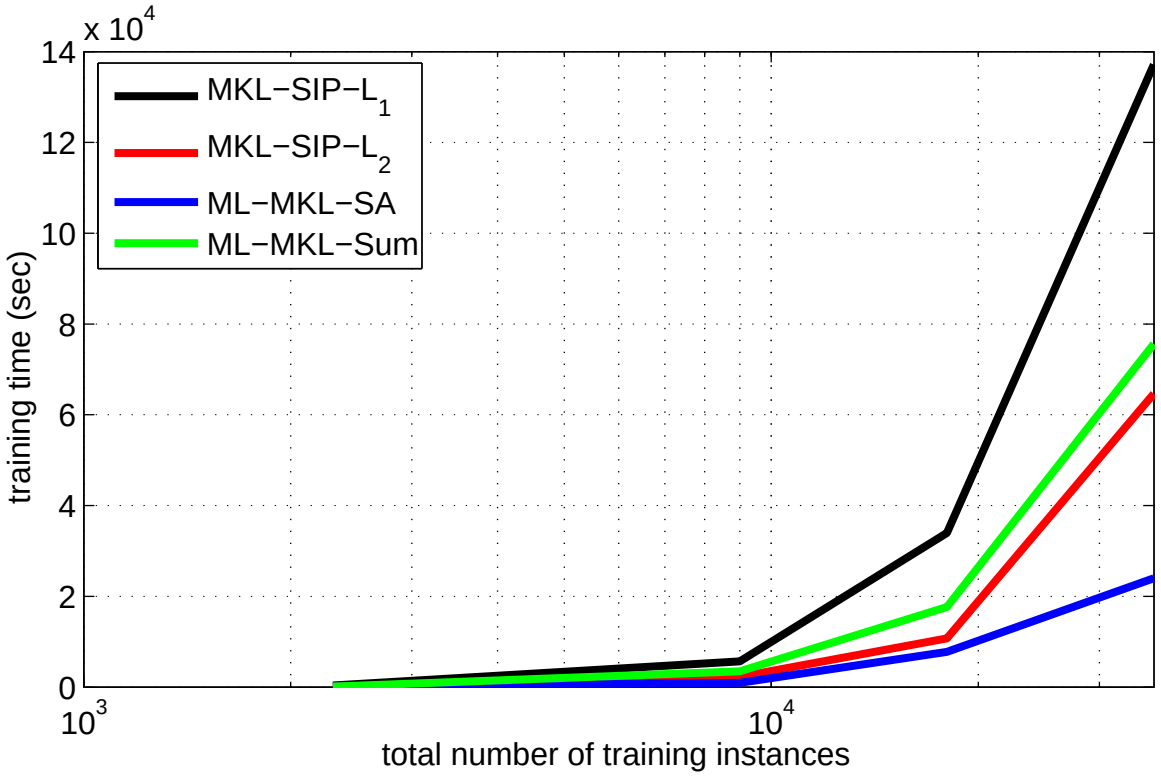

Figure 3.11: Comparison training times for different training set sizes for the ImageNet data set.

We also compare the training times of the baseline methods on the ImageNet data set. The comparison in Fig. 3.11 confirms our previous results and demonstrates the efficiency of the proposed ML-MKL-SA method.

3.5 Conclusions and Future Work

In this chapter, we present an efficient optimization framework for multi-label multiple kernel learning that combines a worst-case analysis with stochastic approximation. Compared to the other algorithms for ML-MKL, the key advantage of the proposed algorithm is that its computational cost is sublinear in the number of classes, making it suitable for handling a large number of classes. We verify the effectiveness of the proposed algorithm by experiments in image categorization

on several benchmark data sets. There are two main directions that we plan to explore in the future. The first one is improving the classification performance. Our experiments showed that, for OvA MKL framework, the proposed method improves the computational efficiency without causing a significant drop in the performance. However, the accuracy in image categorization can be improved by replacing the OvA framework by a multi-label learning formulation. To address this issue, we proposes a multiple kernel multi-label ranking method in Chapter 6. The second future direction is improving the prediction speed, which is in general more crucial than training speed in real world systems. To be able to cope with the increasing size of the image data sets, the prediction step needs to use sparse kernel combinations and classification functions. It is also desirable to have a sublinear dependency of prediction complexity on the number of classes.

Algorithm 1 The proposed Multi-label ranking algorithm

1: **Input**
- η_β, η_γ: step sizes
- $\mathbf{K}$: the kernel matrix
- $\mathbf{y}_1, \ldots, \mathbf{y}_m$: the assignments of m different classes to n training instances
- T: number of iterations
- n, m,s: number of instances, classes, and kernels, respectively
- δ: smoothing parameter

2: **Initialization**
- $\boldsymbol{\gamma}^1 = \mathbf{1}/m$ and $\boldsymbol{\beta}^1 = \mathbf{1}/s$

3: **for** $t = 1, \ldots, T$ **do**

4: Sample a classification task a_t according to the distribution $Multi(\gamma_1^t, \ldots, \gamma_m^t)$.

5: Compute $\boldsymbol{\alpha}^{a_t} = \arg\max_{\boldsymbol{\alpha} \in [0,C]^n} \boldsymbol{\alpha}^\top \mathbf{1} - (\boldsymbol{\alpha} \circ \mathbf{y}_{a_t})^\top \mathbf{K}(\boldsymbol{\beta})(\boldsymbol{\alpha} \circ \mathbf{y}_{a_t})/2$ using an off shelf SVM solver.

6: Compute the estimated gradients $\widehat{g}_j^\beta(\boldsymbol{\beta}^t, \boldsymbol{\gamma}^t)$ and $\widehat{g}_k^\gamma(\boldsymbol{\beta}^t, \boldsymbol{\gamma}^t)$ using Eqs. (3.8) and (3.9).

7: Update $\boldsymbol{\beta}^{t+1}$, $\boldsymbol{\gamma}^{t+1}$ and $\hat{\boldsymbol{\gamma}}^{t+1}$ as follows

$$\begin{aligned} \beta_j^{t+1} &= \frac{\beta_j^t}{Z_\beta^t} \exp(-\eta_\gamma \widehat{g}_j^\beta(\boldsymbol{\beta}^t, \boldsymbol{\gamma}^t)),\ j = 1, \ldots, s. \\ \left[\boldsymbol{\gamma}^{t+1}\right]^k &= \frac{\gamma_k^t}{Z_\gamma^t} \exp(\eta_\gamma \widehat{g}_k^\gamma(\boldsymbol{\beta}^t, \boldsymbol{\gamma}^t)),\ k = 1, \ldots, m \end{aligned}$$

$$\hat{\boldsymbol{\gamma}}^{t+1} = (1-\delta)\gamma^{t+1} + \frac{\delta}{m}\mathbf{1}.$$

8: **end for**

9: Compute the final solution $\bar{\boldsymbol{\beta}}$ and $\bar{\boldsymbol{\gamma}}$ as

$$\bar{\boldsymbol{\gamma}} = \frac{1}{T}\sum_{t=1}^T \boldsymbol{\gamma}^t, \quad \bar{\boldsymbol{\beta}} = \frac{1}{T}\sum_{t=1}^T \boldsymbol{\beta}^t. \tag{3.10}$$

Chapter 4

Image Categorization by Multi-label Ranking

4.1 Introduction

Image categorization requires an image to be assigned to a set of multiple classes, chosen from a large set of class labels. Therefore, image categorization can be cast into multi-label learning, in which each image can be simultaneously classified into more than one class. The most widely used approaches divide a multi-label learning problem into multiple independent binary labeling tasks. The division usually follows one-vs-all, one-vs-one, or the general error correction code framework [120, 121]. Most of these approaches suffer from imbalanced data distributions when constructing binary classifiers. This problem becomes more severe when the number of classes is large. Another limitation of these approaches is that they are unable to capture the correlation among classes [10]. In this chapter, we describe our multi-label ranking method, which addresses these two issues by simultaneously learning classifiers for each label.

Our method tackles the multi-label learning problem using a multi-label ranking approach. For a given example, multi-label ranking aims to rank all relevant classes higher than irrelevant

classes. By converting the classification problem into a ranking problem, multi-label ranking avoids constructing binary classifiers, which operate by distinguishing an individual class from the rest (one-vs-all) of a pair classes from each other (one-vs-one), thus alleviating the problem of imbalanced data distribution. In addition, by avoiding the binary decision regarding which subset of classes should be assigned to each example, multi-label ranking is usually more robust than the classification approaches, particularly when the number of classes is large.

We propose an efficient algorithm to solve the multi-label ranking problem which is based on a simple line search. One advantage of our method compared to the majority of the ranking methods is that the proposed algorithm has a linear dependency on the number of classes. On the other hand, most multi-label ranking methods have quadratic dependency because of the pair-wise class comparisons.

We show that our kernel based multi-label ranking problem formulation is closely related to one-vs-all dual SVM objective. However, unlike the one-vs-all formulation, the proposed cost function cannot be divided into independent components, i.e., one for each class, for optimization. Instead, two features of the proposed method enables exploiting the relationships between labels without making explicit assumptions on the structure of correlations. The first one is a balance constraint, which forces the sum of the dual variables that correspond to positive classes be equal to that of negative classes. The second feature of the proposed method is the optimization scheme it employs which solves the problem for all classes together and chooses the dual variables from a closed set.

4.2 Previous Work

The most widely used approach for multi-label learning is dividing the multi-label learning task into multiple independent binary classification tasks, i.e., learning a binary classifier for each label and deciding the label assignment of a test sample independently for each class. This method is

called binary relevance (BR) or one-vs-all classification. Once a multi-label learning problem is decomposed into multiple binary classification problems, any binary classification algorithm can be employed as a base solver. However, this straightforward approach has several shortcomings. Therefore, we see several attempts in the literature to develop algorithms that specifically address the needs of multi-labeled data, instead of simplifying the multi-label learning task by transforming the problem into an easier one.

There is a very rich literature on multi-label learning. We review multi-label learning methods in four subsections, which are not necessarily mutually exclusive. We also discuss related problems to multi-label learning in Section 4.2.5.

4.2.1 Label Set Transformation Methods

We categorize the methods that fall into this category into two groups: (i) Problem transformation, and (ii) label set projection methods.

4.2.1.1 Problem Transformation Methods

With binary decomposition techniques like one-vs-all and one-vs-one, label set transformation methods were the popular choice for early multi-label learning studies [121]. In a binary decomposition framework, a multi-label learning problem is decomposed into a set of binary classification tasks, which can be easily solved by using well-studied binary classifiers such as SVM or naive Bayes.

One of the shortcomings of the binary decomposition methods is that each classifier is trained independently, meaning that the correlation or dependencies between different classes are not exploited. Such dependencies can be very handy in many applications. Consider an example from automatic image annotation: if an image is tagged with the labels *sun* and *clouds*, it is very likely that the label *sky* is also a relevant label. Therefore, knowing the existence of the label *sun* in the image should be able to make detecting the label *sky* in the image easier. Another problem with

converting a multi-label learning problem into a set of binary classification tasks is the imbalanced (skewed) data distributions, particularly when the number of classes is large.

Another approach for label set transformation is to consider each possible combination of a binary label vector $\mathbf{y^i} = (y_1^i, \ldots, y_m^i) \in \{0,1\}^m$ as an individual class. This approach, which is named as the "label powerset" technique leads to a multi-class single label problem with a total of 2^K new labels, which are named as powerset labels. However, label powerset is not a practical method since the number of classes (powerset labels) in the transformed problem is exponential in the number of original labels.

Dietterich et al. proposed a technique for encoding classifier outputs in a multi-class single-label setting to increase the performance and robustness of the base learners [120]. The authors borrowed the idea of error-correcting coding (ECC) from the communication theory to create distributed output representations. Error-correcting coding is a robust coding scheme that makes detecting and correcting the errors in the output code possible. The main idea in the error correcting output codes (ECOC) scheme is to encode each class by a unique binary string (codeword) of length q. Then, a separate binary classifier is learned to calculate each of these q bits. Once the functions for each codeword digit are learned, the outputs of these q functions are evaluated for each test instance and an output binary string is constructed, which is then compared to all class codewords.

4.2.1.2 Label Set Projection Methods

The idea of projecting a label set into a lower dimensional space before the learning step is a frequently used idea in the multi-label learning literature. The main motivation of using a projected label set instead of the original assignment vector is to increase the computational efficiency by decreasing the number of classes.

The overall framework of the label set projection methods is illustrated by Figure 4.1.

We can summarize the overall process in 4 steps:

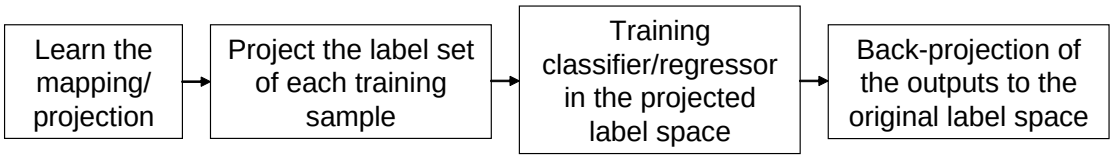

Figure 4.1: A diagram summarizing the label set projection schemes for multi-label learning.

1. Learn or construct the projection operation (matrix) to be used to project the original label vector into (possibly, but not necessarily) a lower dimensional space.

2. Perform the projections: $\psi(\cdot) : \mathbb{R}^m \rightarrow \mathbb{R}^{m'}$, s.t. $\check{\mathbf{y}}^{\mathbf{i}} = \psi(\mathbf{y}^{\mathbf{i}})$, where m is the number of original labels, m' the projected space dimension, $\mathbf{y}^{\mathbf{i}}$ the original label vector and $\check{\mathbf{y}}^{\mathbf{i}}$ is the new label vector in the projected space.

3. Learn a classification/regression model $f(\cdot) : \mathbb{R}^d \rightarrow \mathbb{R}^{m'}$, s.t. $f(\mathbf{x}^{\mathbf{i}}) = \check{\mathbf{y}}^{\mathbf{i}}$.

4. Perform back projection to the original label space from the projected space: $\psi'(\cdot) : \mathbb{R}^{m'} \rightarrow \mathbb{R}^m$, s.t. $\hat{\mathbf{y}}^{\mathbf{i}} = \psi'(\check{\mathbf{y}}^{\mathbf{i}})$, where $\hat{\mathbf{y}}^{\mathbf{i}}$ is the final label vector prediction.

Hsu et al. proposed to use the compressed sensing technique as a label set projection algorithm [122]. With the underlying assumption that label vectors are sparse, their scheme uses random projections for Step 2 and performs regression in Step 3. They show that if the label vectors are k-sparse (average number of nonzero entries is k), then the number of projections would be in the order of $k \log m$, where m is the number of classes. One drawback of this method is that Step 4, the mapping of the predictions back to the original label space, might be complicated since it requires solving an optimization problem for each test sample. Zhou et al. [123] proposed to use the sign of the random Gaussian projections instead of the projections themselves, thus making the projected label matrix $\mathbf{Y}'$ binary and allowing the use of binary classification, instead of regression, which

is employed by the other methods. The recovery step (Step 4) is also different from the original compressed based algorithm [122]. Zhou et al. proposed to use a technique they named as the label set distilling method.

In order to reduce the back-projection step's complexity, Tai et al. proposed a technique called principle label space transform (PLST) [124]. PLST differs from the compressed sensing approach in that its projection matrix is constructed by using the singular vectors of the label matrix $\mathbf{Y}$. Since singular vectors are orthonormal, the projection back to the original label space can be completed simply with a round-based reconstruction: multiplying lower dimensional predictions by the projection matrix and then performing element-wise rounding.

4.2.2 Supervised Algorithm Adaptation Methods

There are also methods that are specifically designed for multi-label learning by adapting the well-known supervised binary classification methods for handling multi-label data. For example, Zhang et al. proposed a maximum a posteriori estimation (MAP) multi-label K-nearest neighbors method (ML-KNN) [125]. In ML-KNN, the estimation of the label vector for a query sample depends on the label prior probability and the probability of assigning a label to an instance conditioned on the number of neighboring instances with the same label.

Schapire et al. proposed two extensions to the well-known Adaboost method for multi-label learning. The first one is Adaboost.MH, which minimizes the Hamming loss and uses a binary decomposition approach, in which each multi-labeled sample is replaced by m new binary sample, with m being the number of labels. The second extension, Adaboost.MR, enforces a bi-partite ranking of labels through a set of pair-wise comparisons [126].

Many well-know decision tree algorithms are adapted to multi-label learning with some modifications. Clare et al. modified the C4.5 decision tree algorithm [127] for multi-label learning by modifying the entropy definition for the information-gain criterion [128]. Alternating decision trees (ADT) [129] and predictive clustering trees (PCT) [130] are other methods that are extended

to multi-label learning [131, 132]. In their PCT based multi-label learning study, Blockeel et al. showed that learning one tree for all labels simultaneously is better in terms of both speed and accuracy when compared with learning an independent tree for each label [132].

4.2.2.1 Transfer learning for multi-label classification

Han et al. proposed a transfer learning scheme for multi-label learning that transfers knowledge between different domains via a linear projection of the data points; this projection is formulated by the use of Graph Laplacian of a label-induced hypergraph and elastic-net regularizer [133]. Claiming that most of the transfer learning methods focus on transferring knowledge between different sources or domains for the same class, Qi et al. presented a multi-label transfer learning approach that aims to perform inter-class knowledge transfer, which can perform within a single domain or multiple domains [134]. They defined a transfer function for each class; these functions depend on two types of similarity measures that are defined for the training samples. One of the similarities is based on a kernel function that strictly uses the input features. The second similarity measure involves both the input features and the corresponding label information through a use of label affinity matrix $\mathbf{S}$. The algorithm described in their study simultaneously optimizes the transfer function and the label affinity matrix.

4.2.3 Multi-label Ranking Methods

One of the earliest multi-label ranking algorithms was proposed in [27]. Constraints derived from the multi-labeled instances were used to enforce the relevant classes to be ranked higher than the irrelevant ones [27]. Crammer et al. [135] improved the computational efficiency of [27] by exclusively considering the most violated constraint: comparing only two labels per instances, with one being the positive label with the minimum output score and the other being the negative label with the maximum output score.

Elisseeff et al. proposed the RankSVM method, which uses pair-wise label ranking loss in the

SVM formulation [136]. Dekel et al. [137] and Shalev-Shwartz et al. [11] encoded the ranking problem using a *preference graph*. A boosting based algorithm was used in [137] to learn the classifiers from a set of given instances and the corresponding preference graphs. Although the described framework in [137] suits any type of ranking task, the multi-label learning problem is formulated as directed bipartite ranking. In [11] a generalization of the hinge loss function for the preference graphs was used for multi-label ranking.

In all these approaches, a ranking model is learned from pairwise constraints between the relevant and irrelevant classes. The number of pair-wise constraints has a quadratic dependence on the number of classes, making it computationally expensive when the number of classes is large. In contrast, our proposed framework for the multi-label ranking that we discuss in this chapter is computationally efficient and can handle a large number of classes (order of 100s).

4.2.4 Exploiting Label Correlation in Multi-label Learning

A number of approaches have been developed for multi-label learning that aim to capture dependencies among classes. In [10], the authors proposed to model the dependencies among the classes using a generative model. Ghamrawi et al. [12] tried to capture the dependencies by defining a conditional random field over all possible combinations of the labels. In [13], a multi-label matrix factorization approach that captures the class correlation via a class co-occurrence matrix was used. A hierarchical Bayesian approach was introduced in [29] to capture the dependency among classes.

There are several approaches [30, 138–141] for multi-label learning that encode the class dependencies under the assumption that some important features are shared among classes. Given the bag-of-words representation of documents, McCallum proposed an EM based scheme that not only estimates the source classes for each document, but also tries to find how the classes contribute to the generation process of the words [138]. By revealing word-class relationship, this method can benefit from label correlations when classifying a document based on its word content.

In their algorithm named MAHR, Huang et al. exploit the label correlations automatically by the hypothesis reuse principle: a hypothesis extracted for one label can be used on other labels [142]. Guo et al. proposed to use conditional dependency networks to model label correlations [143]. A hypergraph representation, in which each vertex is a training instance and each hyperedge for a category is a collection of relevant training samples, was also used to model higher-order label correlations [144, 145]. There are also stacking techniques (i.e., BR+) [146, 147] and classifier chains [148, 149] as feature set transformation methods that exploits class correlations.

We emphasize that our work does not focus on modeling the class correlations explicitly. While indirectly benefiting from dependencies between class labels, we do not make any assumptions regarding the type of relationships that exist between class labels. It should be noted that our proposed multi-label ranking method can be combined with many of the above approaches to further improve the classification performance in multi-label learning.

4.2.5 Related Problems

It is important to note that multi-label learning, despite having a similar goal, differs from a related task, multi-task learning [150]. Multi-task learning can be thought as a bridge between multi-label learning and binary decomposition methods. Similar to binary decomposition methods, a binary classifier is trained for each class. However, unlike binary decomposition methods, the classes are no longer assumed to be independent; rather they are trained using shared information between classes.

Multi-instance learning [151] is another task that can be confused with multi-label learning. The sole goal of multi-label learning is to find the relevant labels of an image. In contrast, multi-instance learning requires locating the concepts/objects in the image.

In this thesis, our understanding of the image categorization problem requires that all categories are pre-defined and have at least one corresponding instance (image) for each category in the training step. In other words, classifiers should be trained for all the classes that are going to be

used in the prediction/testing phase. However, there is a group of studies that are not restricted to this definition. For example, in zero-shot learning [152] and transfer learning [29,141] frameworks, the labels that do not have any corresponding training instances can be used in the prediction stage.

4.3 Maximum Margin Framework for Multi-label Ranking

Let $\mathbf{x^i}, i = 1, \ldots, n$ be the collection of training instances where each example $\mathbf{x^i} \in \mathbb{R}^d$ is a vector of d dimensions. Each training example $\mathbf{x^i}$ is annotated by a set of class labels, denoted by a binary vector $\mathbf{y^i} = (y_1^i, \ldots, y_K^i) \in \{-1, 1\}^m$, where m is the total number of classes, and $y_k^i = 1$ when $\mathbf{x^i}$ is assigned to class c_k and -1 otherwise. In multi-label ranking, we aim to learn m classification functions $f_k(\mathbf{x}) : \mathbb{R}^d \mapsto \mathbb{R}, k = 1, \ldots, m$, one for each class, such that for any example $\mathbf{x}$, $f_k(\mathbf{x})$ is larger than $f_l(\mathbf{x})$ when $\mathbf{x}$ belongs to class c_k and does not belong to class c_l. We define the classification error $\varepsilon_i^{k,l}$ for an example x_i with respect to any two classes c_k and c_l, as follows

$$\varepsilon_{k,l}^i = I(y_i^k \neq y_i^l)\ell\left(\frac{y_k^i - y_l^i}{2}\left(f_k(\mathbf{x^i}) - f_l(\mathbf{x^i})\right)\right), \tag{4.1}$$

where $I(z)$ is an indicator function that outputs 1 when z is true and zero, otherwise. The loss $\ell(z)$ is defined to be the hinge loss, where $\ell(z) = \max(0, 1 - z)$. Note that the above error function outputs 0 when $y_k^i = y_l^i$, namely when no classification error is counted, i.e., $\mathbf{x^i}$ either belongs to both c_k and c_l or $\mathbf{x^i}$ does not belong to either of the two classes.

Following the maximum margin framework for classification, we aim to search for the classification functions $f_k(\mathbf{x}), k = 1, \ldots, m$ that simultaneously minimize the overall classification error. This is summarized into the following optimization problem.

$$\min_{\{f_k \in \mathcal{H}_\kappa\}_{k=1}^m} \frac{1}{2}\sum_{k=1}^m |f_k|_{\mathcal{H}_\kappa}^2 + C\sum_{i=1}^n \sum_{k,l=1}^m \varepsilon_{k,l}^i, \tag{4.2}$$

where $\kappa(\mathbf{x}, \mathbf{x}') : \mathbb{R}^d \times \mathbb{R} \mapsto \mathbb{R}$ is a kernel function, $\mathcal{H}_\kappa$ is a Hilbert space endowed with a kernel

function $\kappa(\cdot,\cdot)$ and C is a regularization parameter. Theorem 3 provides the representer theorem for $f_k(\cdot), k = 1, \ldots, m$.

Theorem 3. *Classification functions $f_k(\mathbf{x})$, $k = 1, \ldots, m$ that optimize Eq. (4.2) are represented in the following form,*

$$f_k(\mathbf{x}) = \sum_{i=1}^{n} y_k^i [\Gamma^i]_k \kappa(\mathbf{x^i}, \mathbf{x}), \tag{4.3}$$

where $[\Gamma^i]_k = \sum_{l=1}^{m} \Gamma_{k,l}^i$. Note that $\Gamma^i \in \mathbf{S}^{m \times m}, i = 1, \ldots, n$ are symmetric matrices that are obtained by solving the following optimization problem:

$$\begin{aligned} \max \quad & \sum_{i=1}^{n}\sum_{k=1}^{m}[\Gamma^i]_k - \frac{1}{2}\sum_{k=1}^{m}\sum_{i,j=1}^{n}\kappa(\mathbf{x^i}, \mathbf{x^j}) y_k^i y_k^j [\Gamma^i]_k [\Gamma^j]_k \\ \textit{s. t.} \quad & \Gamma_{k,l}^i = \begin{cases} 0 \le \Gamma_{k,l}^i \le C & y_k^i \neq y_l^i \\ 0 & \textit{otherwise} \end{cases} \\ & \Gamma^i = [\Gamma^i]^\top, i = 1, \ldots, n;\ k, l = 1, \ldots, m. \end{aligned} \tag{4.4}$$

Proof. See the proof in Section A.4.1 □

The constraints in Eq. (4.4) explicitly capture the relationship between the classes. When an instance $\mathbf{x^i}$ belongs to class c_k, but does not belong to class c_l, the value of $\Gamma_{k,l}^i$ is positive, causing $\mathbf{x^i}$ to be a support vector. The positive terms $\Gamma_{k,l}^i$ are combined into $[\Gamma^i]_k$, which is used in computing the ranking function for class c_k.

4.4 Approximate Formulation

A straightforward approach that directly solves Eq. (4.4) by a standard quadratic programming approach is computationally expensive when the number of classes m is large because the number

of constraints is $O(m^2)$. We show that the relationship between multi-label ranking and the one-vs-all approach provides insight for deriving an approximate formulation for Eq. (4.4) that can be solved efficiently.

4.4.1 Relation to the One-vs-all Approach

Consider constructing $f_k(\mathbf{x})$ in Eq. (4.2) by the one-vs-all approach. The resulting representer theorem for $f_k(\mathbf{x})$ is

$$f_k(\mathbf{x}) = \sum_{i=1}^{n} y_k^i \alpha_k^i \kappa(\mathbf{x^i}, \mathbf{x}), k = 1, \ldots, m \tag{4.5}$$

where $\alpha_k^i, i = 1 \ldots, n; k = 1, \ldots, m$, are obtained by solving the following optimization problem

$$\begin{aligned} \max \quad & \sum_{i=1}^{n}\sum_{k=1}^{m} \alpha_k^i - \frac{1}{2}\sum_{k=1}^{m}\sum_{i,j=1}^{n} \kappa(\mathbf{x^i}, \mathbf{x^j}) y_k^i y_k^j \alpha_k^i \alpha_k^j \\ \text{s. t.} \quad & \alpha_k^i \in [0, C], \quad i = 1, \ldots, n; k = 1, \ldots, m. \end{aligned} \tag{4.6}$$

Comparing the above formulation to Eq. (4.4), we clearly see the mapping, i.e., $[\Gamma^i]_k \leftrightarrow \alpha_k^i$. Hence, the first simplification is to relax Eq. (4.4) by treating each $[\Gamma^i]_k$ as an independent variable, which approximates Eq. (4.4) into the following optimization problem

$$\begin{aligned} \max \quad & \sum_{i=1}^{n}\sum_{k=1}^{m} \alpha_k^i - \frac{1}{2}\sum_{k=1}^{m}\sum_{i,j=1}^{n} \kappa(\mathbf{x^i}, \mathbf{x^j}) y_k^i y_k^j \alpha_k^i \alpha_k^j \\ \text{s. t.} \quad & 0 \leq \alpha_k^i \leq C \sum_{l=1}^{m} I(y_k^i \neq y_l^i), \\ & i = 1, \ldots, n; k = 1, \ldots, m. \end{aligned} \tag{4.7}$$

Note that the constraint $\alpha_k^i \leq C \sum_{l=1}^m I(y_k^i \neq y_l^i)$ follows

$$[\Gamma^i]_k = \sum_{l=1}^m I(y_k^i \neq y_l^i)\Gamma_{k,l}^i \leq C \sum_{l=1}^m I(y_k^i \neq y_l^i).$$

While the problem in Eq. (4.7) can be decomposed into m independent problems, similar to an OvA SVM, this is not adequate for multi-label ranking as the dependence between the functions $f_k(\mathbf{x}), k = 1, \ldots, m$ cannot be captured.

4.4.2 Proposed Approximation

In this section, we present a better approximation of Eq. (4.4) compared to the one presented in Eq. (4.7). Without loss of generality, consider a training example $\mathbf{x^i}$ that is assigned to the first a classes, and is not assigned to the remaining $b = m - a$ classes. According to the definition of Γ^i in (4.4), we can rewrite Γ as

$$\Gamma = \begin{pmatrix} 0 & Z \\ Z^\top & 0 \end{pmatrix}, \tag{4.8}$$

where $Z \in [0, C]^{a \times b}$. Using this notation, variable $\tau_k = [\Gamma^i]_k$ is computed as

$$\tau_k = \begin{cases} \sum_{l=1}^b Z_{k,l} & 1 \leq k \leq a \\ \sum_{l=1}^a Z_{l,k} & a+1 \leq k \leq m \end{cases}$$

where $Z_{k,l}$ is an element in Z that is bounded by 0 and C. According to the above definition, for each instance, τ_k is the sum of either the k^{th} column or the k^{th} row of Z depending on whether the label k is relevant to that instance or not. Formulating τ_k by using Z brings several advantages. Firstly, it enables us to derive constraints for τ_k explicitly in the optimization. Secondly, all τ_k variables depend on each other in the optimization since the components of these variables are

taken from a closed domain Z. This relationship is in fact a special case of the constraint given in Eq. (4.4). The constraint in Eq. (4.4) intuitively forces a balance between the irrelevant and relevant labels of an instance by requiring the sum of the upper bounds of $[\Gamma^i]_k$ that correspond to relevant classes to be equal to that of $[\Gamma^i]_k$ that correspond to irrelevant classes. Obtaining τ_k from Z as formulated above introduces an additional constraint by forcing the sum of the weights corresponding to the relevant labels to be equal to the sum of the weights that are associated with irrelevant labels. This constraint is useful in dealing with the imbalance between the number of relevant and irrelevant labels as well as capturing the dependencies between the classes for that instance.

In order to convert $\tau_k, k = 1, \ldots, m$ into free variables, we need to derive explicit constraints on τ_k that will ensure that each solution of τ_k will result in a feasible solution for Z. Let us first consider a simple case in which we only require elements in Z to be non-negative. Theorem 4 provides the constraints on τ_k.

Theorem 4. *The following two domains Q_1 and Q_2 for vector $\tau = (\tau_1, \ldots, \tau_K)$ are equivalent*

$$
\begin{aligned}
Q_1 &= \{\tau \in \mathbb{R}^m : \exists Z \in \mathbb{R}_+^{a \times b} \textit{s. t.} \\
&\quad \tau_{1:a} = Z\mathbf{1}_b, \tau_{a+1:m} = Z^\top \mathbf{1}_a\} && (4.9) \\
Q_2 &= \left\{\tau \in \mathbb{R}_+^m : \sum_{k=1}^{a} \tau_k = \sum_{k=a+1}^{m} \tau_k\right\} && (4.10)
\end{aligned}
$$

Proof. See Section A.4.2 for the proof. □

Theorem 4 states that the two domains Q_1 and Q_2 are equivalent for vector τ and leads to the following corollary:

Corollary 5. *Consider the following two domains Q_1 and Q_2 for vector $\tau = (\tau_1, \ldots, \tau_m)$*

$$\begin{aligned} Q_1 &= \{\tau \in \mathbb{R}^m : \exists Z \in [0, C]^{a \times b} \text{ s. t.} \\ &\quad \tau_{1:a} = Z\mathbf{1}_b, \tau_{a+1:m} = Z^\top \mathbf{1}_a\} \qquad (4.11) \\ Q_2 &= \left\{\tau \in [0, C]^m : \sum_{k=1}^{a} \tau_k = \sum_{k=a+1}^{m} \tau_k\right\} \qquad (4.12) \end{aligned}$$

We have $\tau \in Q_2 \Rightarrow \tau \in Q_1$.

The above corollary becomes the basis for our approximation. Instead of defining matrix variables $\Gamma^i, i = 1, \ldots, n$ as in (4.4), we introduce the variable α_k^i for $[\Gamma^i]_k$. We furthermore restrict $\alpha^i = (\alpha_1^i, \ldots, \alpha_k^i)$ to be in the domain $\mathcal{G} = \left\{\tau \in [0, C]^m : \sum_{k=1}^{a} \tau_k = \sum_{k=a+1}^{m} \tau_k\right\}$ to ensure that feasible Γ^i can be recovered from a solution of α_k^i. The resulting approximate optimization is

$$\begin{aligned} \max \quad & \sum_{i=1}^{n}\sum_{k=1}^{m} \alpha_k^i - \frac{1}{2}\sum_{k=1}^{m}\sum_{i,j=1}^{n} \kappa(\mathbf{x^i}, \mathbf{x^j}) y_k^i y_k^j \alpha_k^i \alpha_k^j \\ \text{s. t.} \quad & \sum_{k=1}^{m} I(y_k^i = 1)\alpha_k^i = \sum_{k=1}^{m} I(y_k^i = -1)\alpha_k^i, \\ & \alpha_k^i \in [0, C], \quad i = 1, \ldots, n, k = 1, \ldots, m \qquad (4.13) \end{aligned}$$

Unlike Eq. (4.7), Eq. (4.13) cannot be solved as m independent problems since for each instance $\mathbf{x^i}$, the α_k^i from all the classes $c_k, k = 1, \ldots, m$ are involved in the constraint. According to these constraints, for each instance the sum of the weights corresponding to the relevant labels should be equal to the sum of the weights that are associated with irrelevant labels. Theorem 4 shows that by adding this constraint to the problem, the relationships between the classes can be exploited and used without explicitly determining the set Z and the matrices Γ^i. Another advantage of this formulation is that no assumptions on the form of these relationships (e.g., pairwise relationships between classes) are made.

4.5 Efficient Algorithm

We follow the work of Lin et al. [153] and solve Eq. (4.13) by coordinate descent. At each iteration, we choose one training example $(\mathbf{x^i}, \mathbf{y^i})$ and the related variables $\alpha^i = (\alpha_1^i, \ldots, \alpha_m^i)$, while fixing the remaining variables. The resulting optimization problem becomes

$$\begin{aligned} \max \quad & \sum_{k=1}^{m} \alpha_k^i - \frac{1}{2} \sum_{k=1}^{m} y_k^i f_k^{-i}(\mathbf{x^i}) \alpha_k^i - \frac{\kappa(\mathbf{x^i}, \mathbf{x^i})}{2} \sum_{k=1}^{m} (\alpha_k^i)^2 \\ \text{s. t.} \quad & \alpha^i \in [0, C]^m, \ {y^i}^\top \alpha^i = 0 \end{aligned} \tag{4.15}$$

where $f_k^{-i}(\mathbf{x^i})$ is the leave-one-out prediction that can be computed as $f_k^{-i}(\mathbf{x}) = \sum_{j \neq i} y_k^j \alpha_k^j \kappa(\mathbf{x^j}, \mathbf{x})$.

Theorem 6. *The optimal solution to (4.15) is written as*

$$\alpha_k^i = \pi_{[0,C]} \left(\frac{1 + \lambda y_k^i - \frac{1}{2} y_k^i f_k^{-i}(\mathbf{x^i})}{\kappa(\mathbf{x^i}, \mathbf{x^i})} \right), k = 1, \ldots, m \tag{4.16}$$

where λ is the solution to the following equation

$$g(\lambda) = \sum_{k=1}^{m} h \left(\frac{y_k^i + \lambda - \frac{1}{2} f_k^{-i}(\mathbf{x^i})}{\kappa(\mathbf{x^i}, \mathbf{x^i})}, y_k^i C \right) = 0. \tag{4.17}$$

Here $h(\mathbf{x}, y) = \pi_{[0,y]}(\mathbf{x})$ if $y > 0$ and $h(\mathbf{x}, y) = \pi_{[y,0]}(\mathbf{x})$ if $y \leq 0$. Function $\pi_G(\mathbf{x})$ projects $\mathbf{x}$ onto the region G.

Proof. See Section A.4.3 for the proof. □

The function $g(\lambda)$ defined in Eq. (4.17) is a monotonically increasing function of λ which can be solved using the bisection search. The lower and upper bounds for λ for the bisection search are shown in the proposition below.

Proposition 3. *The value of λ that satisfies Eq. (4.17) is bounded by $\lambda_{\min}$ and $\lambda_{\max}$. Define, $\kappa_{ii} = \kappa(\mathbf{x^i}, \mathbf{x^i})$ and $G = [0, C]$,*

$$\eta_{k+}^{-i} = 1 + \frac{1}{2} f_k^{-i}(\mathbf{x^i}) \qquad \eta_{k-}^{-i} = 1 - \frac{1}{2} f_k^{-i}(\mathbf{x^i})$$

$$\Delta = \sum_{k=1}^{m} \delta(y_k^i, 1) \pi_G \left(\frac{\eta_{k-}^{-1}}{k_{ii}} \right) - \sum_{k=1}^{m} \delta(y_k^i, -1) \pi_G \left(\frac{\eta_{k+}^{-i}}{\kappa_{ii}} \right)$$

$$a_{\min} = -C\kappa_{ii} + \min_{y_k^i = -1} \eta_{k+}^{-i} \qquad b_{\min} = -\max_{y_k^i = 1} \eta_{k-}^{-i}$$

$$a_{\max} = Ck_{ii} - \min_{y_k^i = 1} \eta_{k-}^{-i} \qquad b_{\max} = \max_{y_k^i = -1} \eta_{k+}^{-i}$$

If $\Delta < 0$, we have $\lambda_{\min} = 0$ and $\lambda_{\max} = \min(a_{\max}, b_{\max})$. If $\Delta > 0$, we have $\lambda_{\max} = 0$ and $\lambda_{\min} = \max(a_{\min}, b_{\min})$.

Proof. See Section A.4.4 for the proof. □

Once λ is calculated by applying the bisection search between the bounds $\lambda_{\min}$ and $\lambda_{\max}$, it is straightforward to calculate the coefficients α_k^i and finally the ranking functions $f_k(\mathbf{x})$ for any new instance $\mathbf{x}$.

4.6 Experimental Results

In this section, we empirically evaluate the proposed multi-label ranking algorithm by demonstrating its efficiency and effectiveness on the image categorization task.

4.6.1 Data Sets

In order to compare our proposed multi-label learning method to state-of-the-art methods, we use three benchmark data sets: VOC 2007, ESP Game and MIR Flickr25000.

For the VOC 2007 data set, we use the default partitioning suggested by the Pascal VOC Challenges: 5,011 training images and 4,952 test images. We follow [101] and use dense-SIFT features.

Note that the majority of the images in the VOC 2007 data set are labeled by a single class. In fact, the average number of labels per image is only 1.5. Because of this property, VOC 2007 is not an ideal data set for evaluating multi-label learning algorithms. Nevertheless, the performance on the VOC 2007 data set will allow us to examine if the proposed algorithm is effective for image categorization, since VOC 2007 is the most-widely used image categorization benchmark.

The MIR Flickr25000 [154] data set is a subset of the MIR Flickr-1M data set. The original data set contains 25,000 images from 457 classes. However, to be able to create a better test bed for multi-label learning, we remove the images that are assigned to fewer than three classes and take 75% of the instances to form the training set by random sampling. The bag-of-words model based on dense-SIFT features, provided by [101] and [155], are used for image representation.

We also use a subset of the ESP data set, in which the average number of labels per image is 8.3. To study the influence of the number of training samples and labels on multi-label learning performance, we vary the number of training samples and number of labels. In total, we have 20 settings: four training sets with 10,000, 20,000, 30,000, and 40,000 images and five different cases for the number of categories $\{20, 50, 100, 200, 500\}$. After ranking the categories in terms of their frequency (number of images annotated with them) in the data set, we pick the top 20, 50, 100, 200, and 500 categories to create these five different test settings. The number of test images is 10,000. We use dense-SIFT based BoW representation.

4.6.2 Baseline Methods

The following methods are evaluated:

- SVM: We use LIBSVM [156] implementation of the one-vs-all SVM classifier, which is shown to outperform other multi-class SVM methods in [121].

- PLATT: We apply Platt's method to convert SVM scores to posterior probabilities [157]. This conversion makes it easy to compare the output scores of different SVM classifiers,

leading to better performance for multi-label ranking in some cases.

- MLKNN: A nearest neighbor based multi-label classification method [125]. The number of nearest neighbors is chosen by cross-validation. MLKNN is a very popular baseline in the multi-label learning literature due to its simplicity.

- Multiple label shared space model with least square loss (MLLS): A direct multi-label learning method [139] that makes use of the class correlations via a feature space transformation under the assumption of a shared subspace between the categories. MLLS is reported to outperform other state-of-the-art methods that explore class correlations [139].

- MLR-L_1: Our proposed multi-label ranking method that is described in this chapter.

- MLR-GL: Our proposed group lasso based multi-label ranking method that is described in Chapter 5. The approximation parameter η is chosen by cross-validation.

For kernel based methods, we use the RBF kernel with χ^2 distance in our experiments, which has shown to outperform other kernels for image categorization. The regularization parameter C is chosen with a grid search over $\{10^{-2}, 10^{-1}, \ldots, 10^{4}\}$. The bandwidth of the RBF kernel is set to the average pair-wise χ^2 distance between the training image pairs.

4.6.3 Multi-label Ranking Performance

We first compare the ranking performance of the baseline methods in terms of the AUC-ROC and MAP scores. We start by comparing the baselines on the VOC 2007 data set. According to [94, 158], SVM classifier with RBF kernel with χ^2 distance, one of the baselines (SVM) used in our study, yields a comparable performance with the state-of-art methods in the PACAL VOC evaluations. Table 4.1 shows that the proposed algorithm yields a better performance than the one-vs-all SVM method in terms of AUC-ROC and MAP, indicating that the proposed method is effective for image categorization.

Input Image				
True objects	*people, motorbike, car*	*car, people, dog*	*people, motorbike, car*	*car, people, bike*
SVM	*people, car, bus*	*people, car, horse*	*people, cow, motorbike*	*motorbike, people, horse*
PLATT	*people, car, bus*	*people, car, horse*	*people, cow, car*	*motorbike, people, bus*
MLKNN	*people, horse, bus*	*car, people, cat*	*people, car, cat*	*people, motorbike, car*
MLLS	*people, car, bus*	*people, dog, cat*	*people, car, bus*	*bike, people, car*
MLR-L_1	*people, motorbike, car*	*car, people, dog*	*people, motorbike, car*	*car, people, bike*
MLR-GL	*car, people, motorbike*	*people, car, dog*	*people, motorbike, car*	*car, bike, people*

Figure 4.2: For four images from the VOC 2007 data set, the original labels are given in addition to the outputs of baseline methods.

Table 4.1: AUC-ROC and MAP results for the VOC 2007 data set

	SVM	PLATT	MLKNN	MLLS	MLR-L_1	MLR-GL
AUC-ROC	90.7	90.5	89.4	90.7	**91.0**	**91.0**
MAP	65.6	65.6	63.7	66.0	**67.2**	**67.2**

As an illustration, Figure 4.2 shows examples of test images from VOC 2007 data set and the categories predicted by different methods. This figure supports the claim that the categories identified by the proposed ranking method are more relevant to the visual content of images than the baseline methods, particularly for the images that contain several objects.

It is important to note that the evaluation measure we are using in this chapter is not the same as the one used in the VOC competition. In the VOC challenge, the performance is evaluated for each object class separately (category-based), based on the confidence values obtained by binary classifiers. This is not applicable for our case, as we propose a label ranking scheme. We rank the categories for each image in the descending order of their scores, and our AUC measure evaluates how successful is label-ranking. See Section A.1.5 for a detailed discussion on evaluation measures we are using in this dissertation.

Tables 4.2 and 4.3 provide the AUC-ROC and MAP results for the baselines on the ESP Game

Table 4.2: AUC-ROC (%) for the ESP Game data set with 10,000 training images

	20	50	100	200	500
SVM	79.3	79.5	79.8	80.2	80.4
PLATT	79.1	79.4	79.5	79.9	80.0
MLKNN	78.6	78.8	79.5	81.3	83.5
MLLS	79.7	79.4	79.4	79.8	80.2
MLR-L_1	**81.7**	**81.5**	**82.1**	82.9	83.9
MLR-GL	80.2	80.5	81.8	**83.8**	**85.4**

data set when 10,000 images are used for training. From the Tables 4.2 and 4.3, we reach the following conclusions:

- The proposed ranking methods, MLR-L_1 and MLR-GL, consistently and significantly outperforms the other baseline methods.
- Converting SVM scores to posterior probabilities does not improve the performance on this data set.
- MLLS method performs better than SVM, PLATT, and MLKNN baselines when the number of categories is small (20, 50, 100). However, this gap decreases as the number of categories increases, possible because the assumption of a shared subspace that covers all categories is too restrictive when the number of categories is large.
- The relative performances of MLKNN and MLR-GL against the other baselines are better when evaluated by AUC-ROC than MAP. This is because these two baselines do not focus on optimizing the performance for the top ranks (i.e., rank-1, rank-2, etc.).
- MLKNN, which is a very popular baseline in the multi-label learning literature due to its simplicity, is significantly outperformed by the other baselines.
- The methods that are specifically designed for multi-label learning, MLLS and the two ranking methods, outperform one-vs-all SVM in majority of the settings.

Table 4.3: MAP (%) for the ESP Game data set with 10,000 training images

	20	50	100	200	500
SVM	59.3	49.5	43.4	38.0	32.4
PLATT	59.1	49.3	43.4	37.9	32.2
MLKNN	57.0	45.9	39.7	35.2	28.5
MLLS	59.9	48.5	43.3	38.0	32.4
MLR-L_1	**62.2**	**52.0**	**45.5**	**40.0**	**34.2**
MLR-GL	59.4	48.6	42.9	38.2	32.1

Figure 4.3 plots the change of the AUC-ROC score with respect to the number of training images ($10,000$, $20,000$, $30,000$, and $40,000$), when the number of classes is 200. It should be noted that although increasing the number of samples boost the performance of all the baselines, the relative performance of the baselines with respect to each other does not change. The only change in the relative performance is seen for MLLS method. When compared to the one-vs-all SVM baselines and ML-KNN, MLLS takes a better advantage of the increased number of training instances and outperforms them for the settings where the number of training instances is greater than 10,000.

We also check the AUC-ROC and MAP results on the MIR Flickr25000 data set. We see from Table 4.4 that the proposed ranking methods and MLLS give better results compared to one-vs-all SVM baselines, showing again the superiority of direct multi-label learning methods compared to problem transformation approaches like one-vs-all decomposition. Furthermore, the MLR-GL method, which is another multi-label ranking approach, significantly outperforms the other baselines in terms of AUC-ROC score. However, similar to the case in the ESP Game experiments, its relative performance drops in terms of MAP. The proposed MLR-L_1 technique gives comparable results to MLLS, which seems to perform well for the MIR Flickr25000 data set, indicating that the shared subspace assumption is valid for this data set. This indicates that the multi-label learning methods that make strong assumptions when exploiting label correlations have potential to perform well when their assumptions hold, as shown by our MIR Flickr25000 experiments. However,

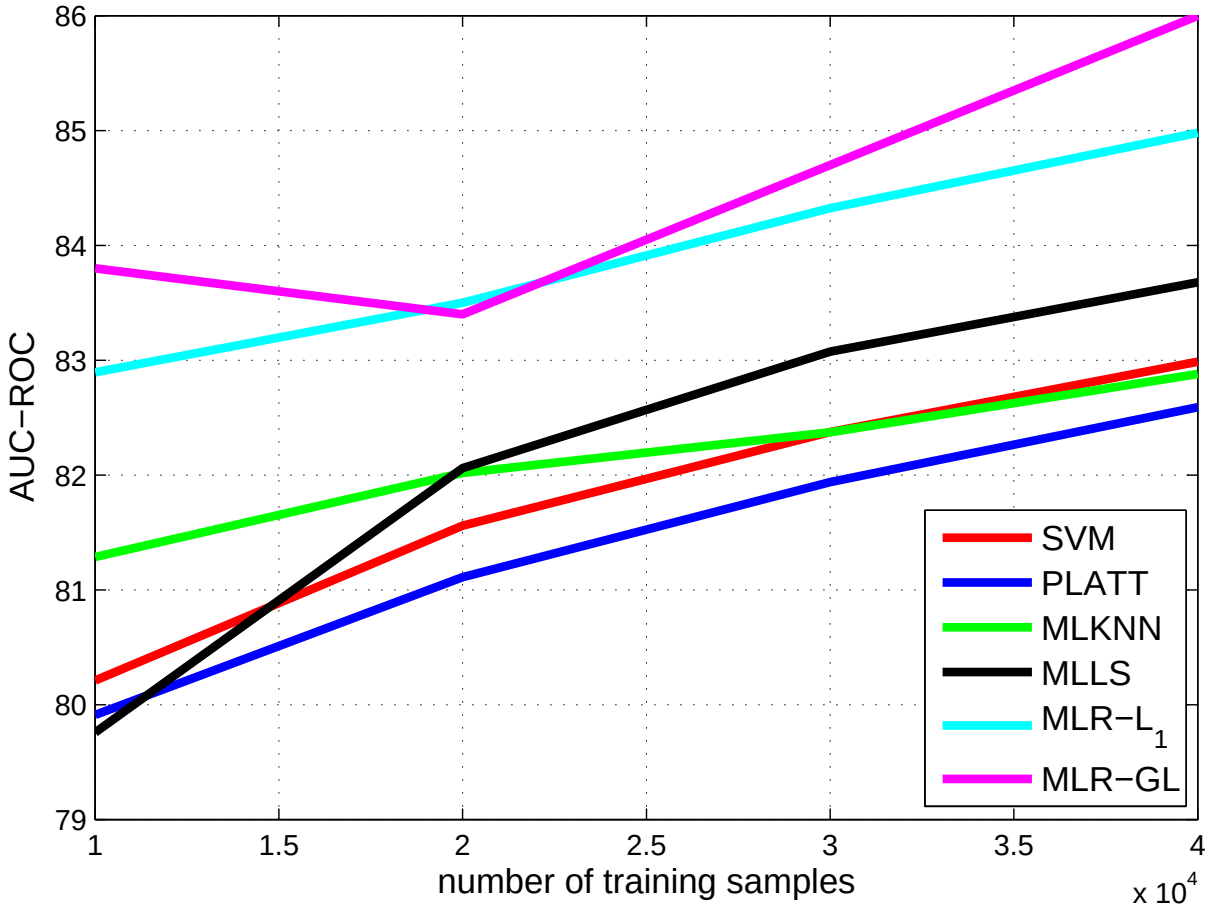

Figure 4.3: Change of the AUC-ROC score with respect to the number of training images.

Table 4.4: AUC-ROC and MAP results for the MIR Flickr25000 data set

	SVM	PLATT	MLKNN	MLLS	MLR-L_1	MLR-GL
AUC-ROC	70.2	68.7	68.6	75.9	75.4	**76.2**
MAP	31.5	31.4	28.9	**33.2**	**33.3**	32.8

these methods might not perform well for the data sets where the underlying assumption does not hold (e.g., ESP Game for the MLLS method).

Finally, we show some example images from the ESP data set and the predicted labels by each baseline method in Table 4.5. The first row under the images gives the true image class labels. For each baseline, we provide the top six returned labels ranked from left to right. The hits are written with bold characters. For example, for the left-most image, SVM provides the following outputs, in the descending relevance order: *ad, computer, screen, book, woman, sign*. Among these six labels, only the labels *ad* and *sign* are correct, meanwhile the other four labels, which are irrelevant for the image, are ranked above the two labels, *logo* and *sign*, causing them to become false negatives. On the other hand, the proposed ranking method MLR-L_1 successfully ranks the labels *ad*, *logo*, and *sign* above all other labels.

4.6.4 Training Time

Figure 4.4 plots the change of the training time of the three baselines (MLR-L_1 [1], SVM, and MLLS) for a fixed number of categories (100) with respect to the number of training samples for the ESP Game data set. In this experiment, we vary the number of training examples from 10,000 to 40,000. We observe that the MLLS method gets computationally more efficient compared to SVM and MLR-L_1 because of its subspace assumption, which allows learning in a lower dimensional space. The main bottleneck of the MLLS algorithm is the SVD operation on the data matrix. However, when the number of samples, n, is large ($n >> d$), the algorithm only calculates d

[1] we will analyse the computational efficiency of the MLR-GL method in Chapter 5

Table 4.5: The label predictions by the baselines for four images from the ESP Game data set. The first row under the images gives the true image class labels. For each baseline, we provide the top six returned labels (three in the top row, and three in the lower row) ranked from left to right. The hits are written with bold characters.

			
labels	**ad logo sign**	**man sky**	**sky tee cloud**
SVM	**ad** computer screen book woman **sign**	**sky** window people light cloud gun	**tee sky** water building rock light
MLKNN	**ad logo** sport **sign** man screen	hair face **sky** **man** tree smile	hair face man **sky** girl woman
MLLS	**logo sign** ocean sea silver book	**sky** window light cloud people **man**	**sky tee** water light rock **cloud**
MLR	**logo sign ad** woman man paper	**sky man** people woman window cloud	**sky tee cloud** building water dark
MLR-GL	**ad sign logo** man picture computer	**sky man** woman girl people hair	**sky cloud tee** water light dark

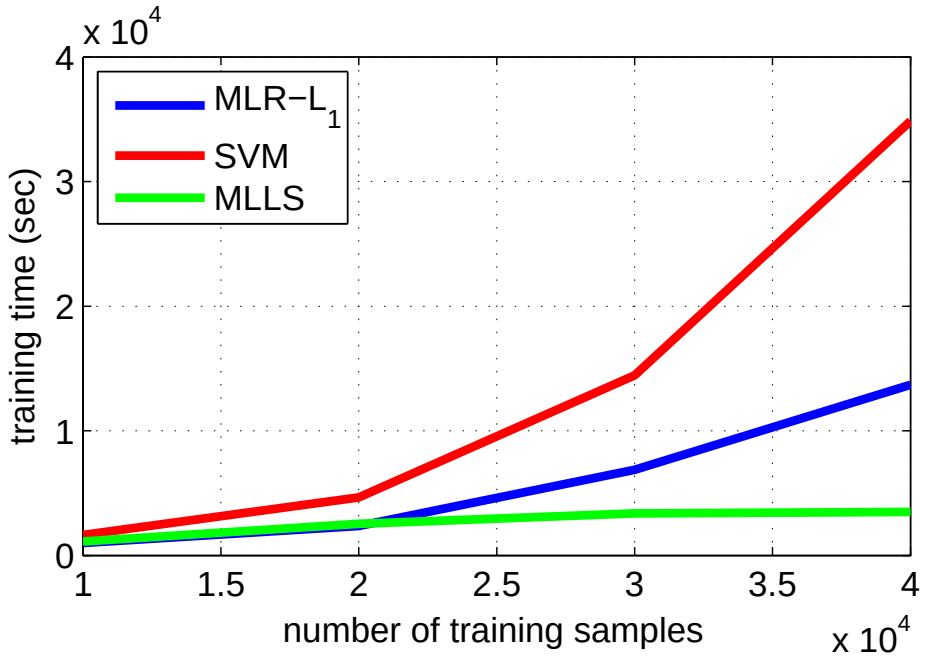

Figure 4.4: Training time of the three baselines for a fixed number of categories (100) with respect to the number of training samples for the ESP Game data set.

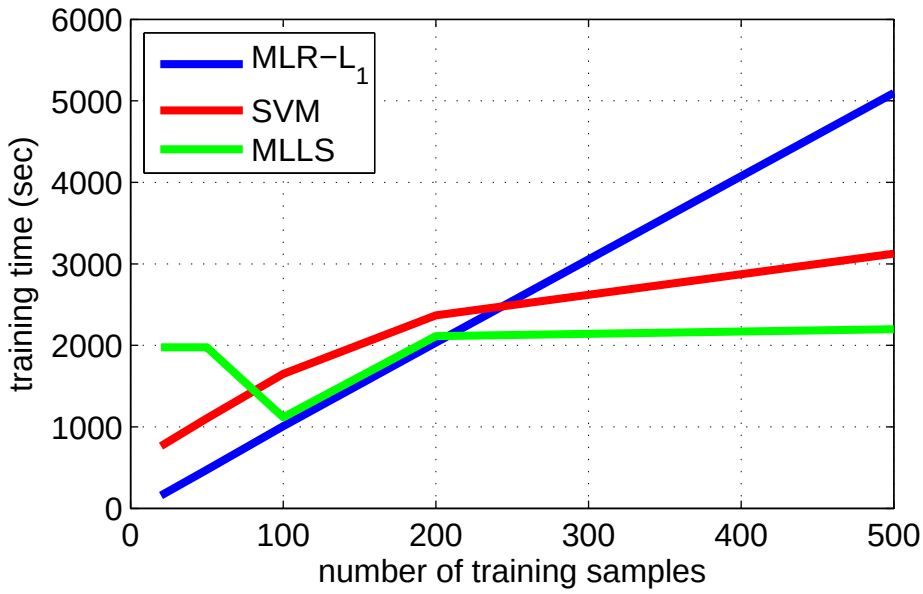

Figure 4.5: Training time of the three baselines for a fixed number of training samples (10,000) with respect to the number of categories for the ESP Game data set.

singular values, where d is the dimension of the feature vectors, and the corresponding d singular vectors. This is why we see that the computational time of MLLS does not increase significantly when the number of samples increases. On the other hand, SVM and MLR-L_1 have a quadratic dependency on the number of samples because they are both kernel-based methods. The difference between the speeds of these two methods increases in favor of MLR-L_1 as the number of samples increases.

Figure 4.5 plots the change of the training time of the three baselines (MLR-L_1, SVM, and MLLS) for a fixed number of training samples (10,000) with respect to the number of categories for the ESP Game data set. This time, we vary the number of categories by using 20, 50, 100, 200, and 500 classes. Similar to the previous case, the MLLS method is the most efficient method among the compared baselines. The training time of the MLR-L_1 method has a linear dependency on the number of categories. Although we would expect SVM to show a very similar characteristic as well, it actually becomes more efficient than MLR-L_1 as the number of categories increase. This

is because the SVM optimization terminates early for the classes that cause the long-tail problem: the classes that have a very small number of positive samples.

To conclude, it is important to note that the proposed ranking method is comparable to, if not more efficient than, one-vs-all SVM in terms of training time. Considering that many researchers are employing one-vs-all SVM in their image categorization studies, MLR-L_1 emerges as a strong alternative. As our empirical analyses showed, shared subspace methods, or label set projection methods (i.e., compressed sensing based multi-label learning), which rely on a similar idea, are computationally more efficient for large scale data sets. However, the proposed baseline can be combined with such approaches to significantly reduce the training time.

4.7 Conclusions and Future Work

We have introduced an efficient multi-label ranking scheme which offers a direct solution to multi-label ranking unlike the conventional methods that use a set of binary classifiers for multi-label learning. Our direct approach enables us to capture the relationships between the class labels without making any assumptions on how these relationships should be modeled. The strength of the proposed approach lies in establishing the relationships between the classifiers by treating them as ranking functions. An efficient algorithm is presented for solving the proposed multi-label ranking approach. Our empirical study of image categorization with three benchmark data sets demonstrated that the proposed method outperforms state-of-the-art methods. Yet, there are some future directions that can be followed to improve the proposed method:

- *Improving the computational efficiency*: The computational efficiency of the proposed method can be improved by combining it with label set space projection methods such as compressed labeling [123] to have a sublinear dependency on the number of labels.

- *Exploiting label correlations*: If the data being used have a label structure that can be modeled explicitly, e.g., hierarchical structure or existing pair-wise correlations between classes,

such structures can be integrated into the proposed multi-label learning framework.

- *Robustness to incomplete label assignments*: The label annotations for training images are often incomplete due to the high cost of the annotation process and the ambiguities between the class labels. It is important to develop multi-label learning methods that are robust to incomplete label assignments. One possible solution is the method we present in Chapter 5.

- *Multiple kernel learning for multi-label ranking*: The proposed multi-label ranking method is limited to a single kernel use. We discussed how considering labels together in a multiple kernel learning task can improve both the computational efficiency and classification accuracy for multi-label data in Chapter 3. The next step in this direction is to extend the proposed multi-label ranking method to multiple kernel learning. To address this issue, we extend the proposed MLR-L_1 algorithm to multiple kernel setting in Chapter 6.

Algorithm 2 Multi-label ranking algorithm:

Input

- $\mathbf{x}^1, \ldots, \mathbf{x}^n$; $\mathbf{x}^i \in \mathbb{R}^d$: Training instances
- $\mathbf{y}^1, \ldots, \mathbf{y}^n$; $\mathbf{y}^i \in \{-1, 1\}^m$: the assignments of m different classes to n training instances
- $\mathbf{K}$: $n \times n$ kernel matrix
- T: number of iterations

Initialization

- $\boldsymbol{\alpha}_k^i = 0$, $i = 1, \ldots, n$, $k = 1, \ldots, K$

for $t = 1, \ldots, T$ **do**

  for $i = 1, \ldots, n$ **do**

    $\Delta = 0$

    Calculate the leave one out prediction:
    $\mathbf{f}^{-\mathbf{i}}(\mathbf{x}^\mathbf{i}) = \mathbf{y}^{i\top}\boldsymbol{\alpha}^i\mathbf{K}_{:,i} - (\mathbf{y}^{i\top}\boldsymbol{\alpha}^i)K_{i,i}$

    Compute Δ

$$\eta_{k+}^{-i} = 1 + \frac{1}{2}f_k^{-i}(\mathbf{x}^\mathbf{i}) \qquad \eta_{k-}^{-i} = 1 - \frac{1}{2}f_k^{-i}(\mathbf{x}^\mathbf{i})$$

$$\Delta = \sum_{k=1}^{m} \delta(y_k^i, 1)\pi_{[0,C]}\left(\frac{\eta_{k-}^{-1}}{K_{ii}}\right) - \sum_{k=1}^{m} \delta(y_k^i, -1)\pi_{[0,C]}\left(\frac{\eta_{k+}^{-i}}{K_{ii}}\right)$$

    where function $\pi_G(x)$ projects x onto the region G.

    Calculate the bounds $\lambda_{\max}$ and $\lambda_{\min}$ for the line search
    $a_{\min} = -C\kappa_{ii} + \min_{y_k^i=-1} \eta_{k+}^{-i}$ $\qquad b_{\min} = -\max_{y_k^i=1} \eta_{k-}^{-i}$
    $a_{\max} = Ck_{ii} - \min_{y_k^i=1} \eta_{k-}^{-i}$ $\qquad b_{\max} = \max_{y_k^i=-1} \eta_{k+}^{-i}$
    if $\Delta < 0$, we have $\lambda_{\min} = 0$ and $\lambda_{\max} = \min(a_{\max}, b_{\max})$
    if $\Delta > 0$, we have $\lambda_{\max} = 0$ and $\lambda_{\min} = \max(a_{\min}, b_{\min})$

    Solve for λ by using a line search and the bounds $\lambda_{\max}$ and $\lambda_{\min}$

$$g(\lambda) = \sum_{k=1}^{K} h\left(\frac{y_k^i + \lambda - \frac{1}{2}f_k^{-i}(\mathbf{x}^\mathbf{i})}{K_{i,i}}, y_k^i C\right) = 0.$$

    where $h(\mathbf{x}, y) = \pi_{[0,y]}(\mathbf{x})$ if $y > 0$ and $h(\mathbf{x}, y) = \pi_{[y,0]}(\mathbf{x})$ if $y \leq 0$.

    Compute

$$\alpha_k^i = \pi_{[0,C]}\left(\frac{1 + \lambda y_k^i - \frac{1}{2}y_k^i f_k^{-i}(\mathbf{x}^\mathbf{i})}{K_{i,i}}\right), k = 1, \ldots, m \tag{4.14}$$

  end for

end for

Chapter 5

Multi-label Ranking for Image Categorization with Incomplete Class Assignments

5.1 Introduction

In Chapter 4, we have discussed the multi-label learning problem in detail, and presented our multi-label ranking approach, MLR-L_1. Our empirical analyses showed that the proposed MLR-L_1 method outperforms the state-of-the-art multi-label learning techniques. The strength of the MLR-L_1 method is its label ranking formulation, which implicitly considers the pair-wise comparisons between the relevant and irrelevant labels for each training image. Simultaneously solving this formulation for all class labels enables an exploitation of the label correlations, one of the main research directions in the multi-label learning literature. However, the performance of multi-label learning techniques, including the MLR-L_1 method, depends on the quality of the training set and the label supervision. It is unclear if strong multi-label learning algorithms would work well in practice. One of the main concerns about the real world systems is that the labeling process

is very expensive and often inaccurate. In image categorization systems, the image annotations for the training data set are provided mostly by online users from online services like Amazon Mechanical Turk. As a result, the retrieved annotations are often incomplete; only a subset of the true image labels are given by the annotators.

person, horse, **grass**, **tail**

bus, car, **grass**, **tail**

tree, animal, ear, sand, **sky**

blue, pink, cartoon, animal, **ear**, **tail**

Figure 5.1: Some example images from the VOC 2007 (top row) and ESP Game (bottom row) data sets with their annotations. The labels written in italic are provided with the images, whereas the ones written in bold fonts are the missing labels. These images, with their missing annotations, are examples of incomplete labeled data.

In this chapter, we consider the image categorization problem from incomplete labeled data. As an example, an image, whose true class assignment is (c_1, c_2, c_3), is only presented with class c_1 when it is used for training. Our goal is to learn a multi-label learning model from training examples that have incomplete class assignments. We refer to this problem as multi-label learning with incomplete class assignments, and the training data as incompletely labeled data. Multi-

label learning with incomplete class assignments is frequently encountered in automatic image annotation when the number of classes is very large, and it is only feasible for users to provide a limited number of class labels for a given instance, as seen in Figure 5.1.

Incompletely labeled data also arise when there is a large ambiguity between class labels, making it difficult for annotators to decide appropriate class assignments for given training instances. Figure 5.2 shows two examples of annotated images from the ESP Game data set. Some of the annotated words used to describe these two images can cause ambiguity. For example, the keywords *baby*, *kid*, and *boy* can be used interchangeably; therefore, an annotator who picks any of these labels would probably not include the other two to the annotation set. Also, note that these annotations are often generated by collapsing annotated words from multiple users. Therefore, it is very likely that some of the labels that cause the label ambiguity problem might be missing from the final list of annotations. Both scenarios, missing labels and label ambiguity, are frequently encountered in the image categorization problem.

It is important to distinguish the learning scenario studied in this work from the related ones in the previous studies such *partial labeling* [159, 160] and *weakly labeled data* [161]. We provide in Table 5.1 some of the related concepts that can be confused with the *multi-label learning with incomplete class assignments* task and briefly highlight the differences.

There is a rich body of literature on multi-label learning, ranging from simple approaches that divide multi-label learning into a set of binary classification problems [162], to more sophisticated approaches that explicitly explore the correlation among classes [10–13]. But none of these approaches directly address the challenge of multi-label learning from incompletely labeled data, which is a more realistic scenario. To this end, we present a multi-label learning framework based on the idea of multi-label ranking [11,15,27,137]. Unlike the classification approaches that make a binary decision about the class assignment for a given instance, multi-label ranking methods rank classes for a given instance such that the relevant classes are ranked before the irrelevant ones. In order to handle the problem of incomplete class assignment, we extend multi-label ranking by

baby, *boy*, **child**, *eye, face, girl, hair, house*, **kid**, *mouth, nose, pink, smile*

anime, *ball, boy,* **cartoon**, **drawing**, *girl, group, hair, kid, man, people, play, red*

Figure 5.2: Example images from the ESP Game data set and their annotations. The annotations highlighted by bold font, which are used to annotate the same concept/object in the corresponding images, are examples of the label ambiguity problem.

Table 5.1: Some concepts that can be confused with the incomplete label assignment problem

problem	**bib**	**definition**
partial labeling	[159, 160]	Only one of the positive class assignments is correct
weakly labeled data	[161]	A value indicating correctness of predictions is given
weakly tagged images	[164]	Some of the class assignments are incorrect
partially labeled data	[165]	Another name for semi-supervised learning
bandit multi-class learning	[166, 167]	Learner receives partial feedback, e.g., click data.

exploiting the group lasso technique [163] to combine the errors in ranking the assigned and unassigned classes for each image. As will be seen in the following discussions, by using group lasso to combine ranking errors, the proposed framework is able to automatically detect the missing class assignments in the training set and consequentially improve the classification accuracy.

We present an efficient learning algorithm for the proposed framework. The efficiency of a multi-label ranking method is important, since a naive implementation would result in performing a pairwise comparison between all possible image pairs, making it difficult to scale to a large number of classes and training instances. Our empirical studies on two benchmark data sets for image categorization indicate that (i) our framework is robust to the missing class assignments

problem and performs better than the state-of-the-art multi-label learning approaches in the case of incompletely labeled data, and (ii) the proposed approach is computationally efficient and scales well to large numbers of training examples and classes.

5.2 A Framework for Multi-label Learning from Incompletely Labeled Data

In order to handle incompletely labeled data, we consider exploring the group lasso regularizer when estimating the error in ranking the assigned classes against the unassigned ones. The key idea is to selectively penalize the ranking errors. To facilitate our discussion, we follow the notation in Chapter 4 and consider an instance $\mathbf{x}$ that is assigned to classes $c_1, \ldots, c_a$. Consequently, classes $c_{a+1}, \ldots, c_m$ are remained as the unassigned classes for $\mathbf{x}$. If example $\mathbf{x}$ is fully labeled, following [15], the ranking error for the classification functions $f_k(\mathbf{x}), k \in [m]$ is expressed as

$$\sum_{k=1}^{a} \sum_{l=a+1}^{m} \max(0, f_l(\mathbf{x}) - f_k(\mathbf{x}) + 1). \tag{5.1}$$

However, given the data is only partially labeled, some of the unassigned class labels may indeed be the true classes, and the above loss function for $\mathbf{x}$ may overestimate the classification error. To address this issue, we introduce a slack variable, denoted by $\varepsilon_{k,l}$, to account for the error of ranking an unassigned class l before the assigned class k. This introduces the following constraint

$$\varepsilon_{k,l} + f_k(\mathbf{x}) \geq 1 + f_l(\mathbf{x}). \tag{5.2}$$

Now, instead of adding all the errors together for an instance $\mathbf{x}$, i.e., $\sum_{k=1}^{a}\sum_{l=a+1}^{m}\varepsilon_{k,l}$, we combine the ranking errors $\varepsilon_{k,l}$ via a group lasso regularizer, i.e.,

$$\sum_{l=a+1}^{m}\sqrt{\sum_{k=1}^{a}\varepsilon_{k,l}^2} \tag{5.3}$$

The motivation of using group lasso for aggregating ranking errors is two fold: first, as stated in the general theory, group lasso is able to select a group of variables, which in our case, is to select the group of ranking errors $\{\varepsilon_{k,l}, k = 1,\ldots,a\}$ for each unassigned class c_l. In particular, an unassigned class c_l is likely to be a missing class assignment for an instance x when many of its ranking errors $\{\varepsilon_{k,l}\}_{k=1}^{a}$ are non-zero, which coincides with the criterion of group selection by group lasso. Thus, by using the group lasso regularizer, we may be able to decide which unassigned classes are indeed the missing correct class assignments. Second, group lasso usually results in a sparse solution in which most of the group variables are zero and only a small number of groups are assigned non-zero values. In our case, the sparse solution implies that most of the unassigned classes for $\mathbf{x}$ are indeed correct, and only a few unassigned classes are the true class assignments for $\mathbf{x}$ that are missed during annotation.

Let $\mathbf{x^1},\ldots,\mathbf{x^n}$ be the collection of training instances that are labeled by $Y_1,\ldots,Y_n$, where each $Y_i \subset \mathcal{Y}$. For the convenience of presentation, we represent each class assignment Y_i by a binary vector $y^i = (y_1^i,\ldots,y_m^i) \in \{-1,+1\}^m$, where $y_k^i = +1$ if $k \in Y_i$ and $y_k^i = -1$ if $k \notin Y_i$. Using the group lasso regularizer described above, we have the following optimization problem:

$$\min_{f_k\in\mathcal{H}_\kappa}\frac{1}{2}\sum_{k=1}^{m}|f_k|_{\mathcal{H}_\kappa}^2 + C\sum_{i=1}^{n}\sum_{l\notin Y_i}\sqrt{\sum_{k\in Y_i}\ell^2(f_k(\mathbf{x^i}) - f_l(\mathbf{x^i}))}, \tag{5.4}$$

where $\ell(z) = \max(0, 1-z)$ is the hinge loss function that assesses the error in ranking two classes c_k and c_l. In the next section, we discuss the strategy for efficiently optimizing Eq. (5.4).

5.3 Optimization Algorithm

First, we have the following representer theorem for $f(\mathbf{x})$ that optimizes Eq. (5.4).

Theorem 7. *The optimal solution to Eq. (5.4) admits the following expression for $f(\mathbf{x})$, i.e.,*

$$f_k(\mathbf{x}) = \sum_{i=1}^{n} y_k^i \alpha_k^i \kappa(\mathbf{x}, \mathbf{x^i}), \quad k = 1, \ldots, m,$$

where $\alpha_k^i, i = 1, \ldots, n$ are the combination weights.

It is straightforward to verify the above representer theorem. Next, in order to solve Eq. (5.4) efficiently, we aim to linearize the objective function in Eq. (5.4) by using the following lemma.

Lemma 1. $\sum_{l=a+1}^{m} \sqrt{\sum_{k=1}^{a} \ell^2(f_k(\mathbf{x^i}) - f_l(\mathbf{x^i}))}$ *is equivalent to the following expression:*

$$\begin{aligned} \max_{\gamma^i \in \mathbb{R}^{a \times (m-a)}} \quad & \left\{ \sum_{l=a+1}^{m} \sum_{k=1}^{a} \gamma_{k,l}^i \ell(f_k(\mathbf{x^i}) - f_l(\mathbf{x^i})) \right\} \\ \textit{s.t.} \quad & \max_{1 \le l \le m-a} |\boldsymbol{\gamma}_{\cdot,l}^i|_2 \le 1, \end{aligned} \tag{5.5}$$

where $\boldsymbol{\gamma}_{\cdot,l}$ stands for the lth column vector of matrix $\boldsymbol{\gamma}^i$.

Lemma 1 follows directly from the fact that $\sum_{l=a+1}^{m} \sqrt{\sum_{k=1}^{a} \ell^2(f_k(\mathbf{x^i}) - f_l(\mathbf{x^i}))}$ is a $L_{1,2}$ norm of the loss function $\ell(f_k(\mathbf{x}) - f_l(\mathbf{x}))$ and the dual norm of $L_{1,2}$ is $L_{\infty,2}$. See Section A.5.1 for a detailed proof.

Using lemma 1, we turn Eq. (5.4) into a convex-concave optimization problem as revealed in the following theorem.

Theorem 8. *The problem in Eq. (5.4) is equivalent to the following convex-concave optimization*

problem

$$\max_{\{\gamma^i \in \Delta_i\}_{i=1}^n} \min_{\{f_k \in \mathcal{H}_\kappa\}_{k=1}^m} L = \frac{1}{2}\sum_{k=1}^{m} |f_k|^2_{\mathcal{H}_\kappa} \tag{5.6}$$
$$+C\sum_{i=1}^{n}\sum_{l\notin Y_i}\sum_{k\in Y_i}\gamma^i_{k,l}\ell(f_k(\mathbf{x^i}) - f_l(\mathbf{x^i})),$$

where $\gamma^i = [\gamma^i_{k,l}]_{m\times m}$ *and*

$$\Delta_i = \left\{ \begin{array}{c} \gamma^i_{k,l} \geq 0, k, l = 1, \ldots, m, \\ \gamma^i \in \mathbb{R}^{m\times m} : \ \gamma^i_{k,l} = 0 \textit{ if } l \in Y_i \textit{ or } k \notin Y_i, \ , \\ \max\limits_{1\leq l\leq m} |\boldsymbol{\gamma}^i_{\cdot,l}|_2 \leq 1 \end{array} \right\}$$

The above theorem follows by directly plugging the result of Lemma 1 into Eq. (5.4). As indicated by the above theorem, the introduction of the group lasso is equivalent to introducing a different weight $\gamma^i_{k,l}$ for each comparison between an assigned class and an unassigned class. It is the introduction of these weights that allows us to determine which unassigned class is missed in the annotation process.

Theorem 9. *The optimal solution* $f(\mathbf{x})$ *to Eq. (5.6) can be expressed as follows:*

$$f_k(\mathbf{x}) = \sum_{i=1}^{n} y^i_k \alpha^i_k \kappa(\mathbf{x}, \mathbf{x^i}),$$

where $\boldsymbol{\alpha}^i = (\alpha^i_1, \ldots, \alpha^i_m)^\top, i = 1 \ldots n$ *is the optimal solution to the following optimization problem:*

$$\max_{\{\boldsymbol{\alpha}^i \in \Omega_i\}_{i=1}^n} \sum_{k=1}^{m}\left(\sum_{i=1}^{n}\alpha^i_k - \sum_{i,j=1}^{n}\alpha^i_k\alpha^j_k y^i_k y^j_k K_{i,j}\right), \tag{5.7}$$

where

$$\Omega_i = \left\{ \boldsymbol{\alpha}^i \in \mathbb{R}^m : \exists \gamma^i \in \Delta_i \textit{ s. t. } \boldsymbol{\alpha}^i = C\gamma^i \mathbf{1} + C[\gamma^i]^\top \mathbf{1} \right\}.$$

The proof of this theorem can be found in the Section A.5.2. Note that although the objective function in Eq. (5.7) is similar to that of SVM, it is the constraints specified in domain Ω_i that makes this problem computationally more challenging.

Algorithm 3 Multi-label ranking algorithm with Group Lasso

Input
- $\mathbf{x}^1, \ldots, \mathbf{x}^n$; $\mathbf{x}^i \in \mathbb{R}^d$: Training instances
- $\mathbf{y}^1, \ldots, \mathbf{y}^n$; $\mathbf{y}^i \in \{-1, 1\}^m$: the assignments of m different classes to n training instances
- $\mathbf{K}$: $n \times n$ kernel matrix
- T: number of iterations

Initialization
- $\alpha_j^i = 0$, $i = 1, \ldots, n$, $j = 1, \ldots, m$

for $t = 1, \ldots, T$ **do**
  for $i = 1, \ldots, n$ **do**
    $\Delta = 0$
    Calculate the leave one out prediction vector:
    $\mathbf{f^{-i}} = \mathbf{y}^{i\top} \boldsymbol{\alpha}^i \mathbf{K}_{:,i} - (\mathbf{y}^{i\top} \boldsymbol{\alpha}^i) K_{i,i}$
    $a = \sum_{j=1}^m I(y_j^i == 1)$ & $b = \sum_{j=1}^m I(y_j^i = -1)$,
    where $I(z)$ is an indicator function that outputs 1 when z is true and zero, otherwise.
    Construct $\mathbf{f_a^{-i}}$ and $\mathbf{f_b^{-i}}$ such that $\mathbf{f^{-i}}(\mathbf{x^i})$=$\mathbf{f_a^{-i}} \bigcup \mathbf{f_b^{-i}}$
    $\mathbf{f_a^{-i}}$: components of $\mathbf{f^{-i}}$ that corresponds to positive labels, i.e., $y_j^i = 1$.
    $\mathbf{f_b^{-i}}$: components of $\mathbf{f^{-i}}$ that corresponds to negative labels, i.e., $y_j^i = -1$.
    Compute matrix $\mathbf{H} \in \mathbb{R}^{a \times b}$: $\mathbf{H} = \frac{1}{2}((\mathbf{1}_b^\top \mathbf{1}_a) - \mathbf{f_b^{-i}} \mathbf{1}_a^\top - \mathbf{1}_b \mathbf{f_a^{-i}})^\top$
    Construct matrix $\boldsymbol{\gamma} \in \mathbb{R}^{a \times b}$
    for $s = 1, \ldots, b$ **do**
      $\gamma_{:,s} = \frac{\pi_+(\mathbf{H}_{:,\mathbf{s}})}{|\pi_+(\mathbf{H}_{:,\mathbf{s}})|_2} \min(1, \frac{|\pi_+(\mathbf{H}_{:,\mathbf{s}})|_2}{\eta C K_{i,i}})$
      where function $\pi_+(\mathbf{z})$ projects $\mathbf{z}$ onto the region $\mathbb{R}_+^a$.
    end for
    Calculate $\boldsymbol{\alpha}$
    $\boldsymbol{\alpha}_a = C\gamma \mathbf{1_b}$
    $\boldsymbol{\alpha}_b = C\gamma^\top \mathbf{1_a}$
    $\boldsymbol{\alpha} = \boldsymbol{\alpha}_a \bigcup \boldsymbol{\alpha}_b$
  end for
end for

In order to efficiently solve Eq. (5.7), we consider the block coordinate descent method. In

particular, we aim to optimize $\boldsymbol{\alpha}^i$ with the other dual variables, $\{\boldsymbol{\alpha}^j, j \neq i\}$, being fixed. Without a loss of generality, we assume that example $\mathbf{x^i}$ is assigned to the first a classes and is not assigned to the remaining $b = m - a$ classes. For the convenience of presentation, we drop the index i and write $\boldsymbol{\alpha}^i$ as $\boldsymbol{\alpha}$. We thus have the following optimization problem for $\boldsymbol{\alpha}^i$.

$$\max_{\boldsymbol{\alpha}\in\Omega} \sum_{k=1}^{m} \alpha_k - K_{i,i} \sum_{k=1}^{m} \alpha_k^2 - 2 \sum_{k=1}^{m} y_k \alpha_k \sum_{j\neq i} \alpha_k^j y_k^j K_{i,j}, \tag{5.8}$$

where Ω is defined as

$$\Omega = \big\{\boldsymbol{\alpha} \in \mathbb{R}^m : \exists \boldsymbol{\gamma} \in \mathbb{R}_+^{a\times b}, |\boldsymbol{\gamma}_{\cdot,l}|_2 \leq 1, l \in [b]$$
$$\text{s.t. } \boldsymbol{\alpha}_{1:a} = C\boldsymbol{\gamma}\mathbf{1}_b,\ \boldsymbol{\alpha}_{a+1:a+b} = C\boldsymbol{\gamma}^\top \mathbf{1}_a\big\}.$$

In the above, we use the notation $\boldsymbol{\alpha}_{i:j} = (\alpha_i, \ldots, \alpha_j)$ to represent a subset of vector $\mathbf{b}$ whose index ranges from i to j. $\mathbf{1}_a$ represents a vector of a dimensions with all its elements being one. We now aim to simplify the problem in Eq. (5.8). First, we have for any $\alpha \in \Omega$

$$\sum_{k=1}^{m} \alpha_k = 2C(\mathbf{1}_a^\top \boldsymbol{\gamma} \mathbf{1}_b). \tag{5.9}$$

Second, we have

$$\sum_{k=1}^{m} \alpha_k^2 = \sum_{k=1}^{a} \alpha_k^2 + \sum_{k=a+1}^{a+b} \alpha_k^2 = C^2 \left(\mathbf{1}_b^\top \boldsymbol{\gamma}^\top \boldsymbol{\gamma} \mathbf{1}_b + \mathbf{1}_a^\top \boldsymbol{\gamma}\boldsymbol{\gamma}^\top \mathbf{1}_a\right). \tag{5.10}$$

To simplify the last term in Eq. (5.8), we define

$$f_k^{-i}(\mathbf{x^i}) = y_k \sum_{j\neq i} \alpha_k^j y_k^j \kappa(\mathbf{x^i}, \mathbf{x^j}), \tag{5.11}$$

and vector $\mathbf{f}^{-i} = (f_1^{-i}(\mathbf{x^i}), \ldots, f_i^{-i}(x_i)) = (\mathbf{f}_a^{-i}, \mathbf{f}_b^{-i})$. Using these notations, the third term in

Eq. (5.8) becomes

$$\sum_{k=1}^{m} \alpha_k f_k^{-i}(x_i) = \boldsymbol{\alpha}^\top \mathbf{f}^{-i} = C\text{tr}\left(\left(\mathbf{1}_b[\mathbf{f}_a^{-i}]^\top + \mathbf{f}_b^{-i}\mathbf{1}_a^\top]\right)\boldsymbol{\gamma}\right). \tag{5.12}$$

Thus, we have the following optimization problem to solve

$$\begin{aligned}\max_{\gamma\in\Delta} \mathbf{1}_a^\top\boldsymbol{\gamma}\mathbf{1}_b - \frac{1}{2}CK_{i,i}\left(\mathbf{1}_b^\top\boldsymbol{\gamma}^\top\boldsymbol{\gamma}\mathbf{1}_b + \mathbf{1}_a^\top\boldsymbol{\gamma}\boldsymbol{\gamma}^\top\mathbf{1}_a\right) \\ -\text{tr}\left(\left(\mathbf{f}_b^{-i}\mathbf{1}_a^\top + \mathbf{1}_b[\mathbf{f}_a^{-i}]^\top\right)\boldsymbol{\gamma}\right),\end{aligned} \tag{5.13}$$

where $\Delta = \{\boldsymbol{\gamma} \in \mathbb{R}_+^{a\times b} : |\boldsymbol{\gamma}_{\cdot,l}|_2 \leq 1, l = 1,\ldots,b\}$. The problem in Eq. (5.13) is indeed a Second Order Cone Programming (SOCP) problem [168]. Although a SOCP problem can be solved by a standard tool like SeDuMi [88], it can still be computationally expensive to solve a large-scale SOCP problem. We thus further simplify Eq. (5.13) by the following approximation

$$\mathbf{1}_b^\top\boldsymbol{\gamma}^\top\boldsymbol{\gamma}\mathbf{1}_b + \mathbf{1}_a^\top\boldsymbol{\gamma}\boldsymbol{\gamma}^\top\mathbf{1}_a \approx \eta\text{tr}(\boldsymbol{\gamma}^\top\boldsymbol{\gamma} + \boldsymbol{\gamma}\boldsymbol{\gamma}^\top) = 2\eta\text{tr}(\boldsymbol{\gamma}^\top\boldsymbol{\gamma}), \tag{5.14}$$

where $\eta > 1$ is a parameter introduced for approximation. Using the approximation in Eq. (5.14), we have

$$\max_{\boldsymbol{\gamma}\in\Delta} \mathbf{1}_a^\top\boldsymbol{\gamma}\mathbf{1}_b - CK_{i,i}\eta\text{tr}(\boldsymbol{\gamma}^\top\boldsymbol{\gamma}) - \text{tr}\left(\left(\mathbf{f}_b^{-i}\mathbf{1}_a^\top + \mathbf{1}_b[\mathbf{f}_a^{-i}]^\top\right)\boldsymbol{\gamma}\right), \tag{5.15}$$

where we define

$$\left((\mathbf{1}_b\mathbf{1}_a^\top) - \mathbf{f}_b^{-i}\mathbf{1}_a^\top - \mathbf{1}_b[\mathbf{f}_a^{-i}]^\top\right)^\top = 2\mathbf{H} = (2\mathbf{h}_1,\ldots,2\mathbf{h}_b). \tag{5.16}$$

Lemma 2 shows a closed form solution to Eq. (5.15).

Lemma 2. *The optimal solution to Eq. (5.15) is*

$$\boldsymbol{\gamma}_{\cdot,s} = \frac{\pi_{\mathcal{G}}(\mathbf{h}_s)}{|\pi_{\mathcal{G}}(\mathbf{h}_s)|_2} \min\left(1, \frac{|\pi_{\mathcal{G}}(\mathbf{h}_s)|_2}{CK_{i,i}\eta}\right), \quad s = 1, \ldots, b, \tag{5.17}$$

where $\mathcal{G} = \{\mathbf{z} : \mathbf{z} \in \mathbb{R}_+^a\}$ *and* $\pi_{\mathcal{G}}(\mathbf{h})$ *projects vector* $\mathbf{h}$ *into the domain* $\mathcal{G}$*.*

The proof of this lemma can be found in Section A.5.3.

5.4 Experimental Results

5.4.1 Data Sets

In order to evaluate the proposed method for multi-label learning with incomplete class assignments, we use two multi-label data sets that were used in Chapter 4: subsets of the ESP Game and MIR Flickr25000 data sets.

For MIR Flickr25000, we remove the images that are assigned to fewer than three classes. This procedure gives us 10,199 images from 457 classes. We take 75% of the examples to form a training set by random sampling. The bag-of-words model based on dense-SIFT features, provided by [101] and [155], are used for image representation.

We use a subset of the ESP data set, in which the average number of labels per image is 8.3. To study the influence of the number of training samples and labels on multi-label learning performance, we vary the number of training samples and labels. We follow the protocol in Chapter 4 to vary the number of training instances and classes. The number of test images is 10,000. We use dense-SIFT based BoW representation to construct image features.

To simulate the situation of incomplete class assignment, we conduct experiments in four different settings for the ESP Game and MIR Flickr25000 data sets. In the first setting, termed *case-1*, there is no missing class assignment for any training image. In the next three settings, termed *case-2*, *case-3*, and *case-4*, for each training image, we randomly choose 20%, 40%, and

Table 5.2: AUC-ROC (%) for the ESP Game data set with 10,000 training images and 200 classes.

	case-1	**case-2**	**case-3**	**case-4**
SVM	80.2	79.2	77.5	75.2
PLATT	80.1	79.5	77.9	75.9
MLKNN	81.3	72.5	72.3	72.1
MLLS	79.8	78.9	77.3	75.0
MLR-L_1	82.3	82.2	81.1	79.4
MLR-GL	**83.8**	**83.4**	**82.8**	**82.1**

Table 5.3: MAP (%) for the ESP Game data set with 10,000 training images and 200 classes.

	case-1	**case-2**	**case-3**	**case-4**
SVM	38.0	36.2	34.0	31.0
PLATT	37.9	36.5	34.5	31.8
MLKNN	35.2	26.4	25.8	25.6
MLLS	38.0	37.0	35.5	33.1
MLR-L_1	**40.0**	**38.0**	**37.1**	35.2
MLR-GL	38.2	37.5	36.8	**35.4**

60% of the assigned class labels, respectively, and remove them from the training data. During the label removal process, we make sure that each image has at least one positive class label.

5.4.2 Baseline Methods

We use the same baselines as in Chapter 2: SVM [121], PLATT [157], MLKNN [125], MLLS [139], MLR-L_1, and MLR-GL, the proposed group lasso based multi-label ranking method that is described in this chapter and specifically addresses the multi-label learning with incomplete class assignment problem.

When calculating the kernel matrix, a modified chi-squared kernel with $d(\mathbf{x}, \mathbf{x}') = |\mathbf{x} - \mathbf{x}'|_2^2/|\mathbf{x} + \mathbf{x}'|_2^2$, is used for the ESP GAME and MIR Flickr25000 data sets because it yields significantly better performance than the standard version. σ is set to be chi-squared kernel is chosen as the mean of the pair-wise distances $d(x, y)$ [69]. The optimal values for parameters C and the

Table 5.4: The label predictions by the baselines for four images from the ESP Game data set, when 40% of the training labels are missing. The first row under the images gives the true image class labels. For each baseline, we provide the top nine returned labels (three in the top row, and three it the lower row) ranked from left to right. The hits are written with bold characters.

			
labels	**silver circle round**	**sky orange dark**	**tree road wood**
SVM	**silver** diamond **circle** jewelry metal wrist tree time wood	**dark** man cloud computer face wave metal space dance	water sidewalk ride ocean man boat wall animal fish
MLKNN	man ad woman metal face girl logo people sky	man hair face girl **sky** people woman water smile	ad hair sky girl man **tree** water smile screen
MLLS	**silver circle** tree **round** dark line wood hand orange	face **dark** night sea eyes ocean teeth computer **orange**	water sea sky ocean man cloud **tree** wall street
MLR-L_1	**silver circle round** dark woman line orange logo wood	man **dark** light lights cloud **orange** shadow night sun	ocean sky man water sea wall men people **wood**
MLR-GL	**round circle silver** ad logo square line face woman	light **dark** man woman night girl **sky orange** people	sky water man **wood** sea people **tree road** woman

Table 5.5: AUC-ROC results for the MIR Flickr data set

	case-1	**case-2**	**case-3**	**case-4**
SVM	70.2	69.1	67.6	65.7
PLATT	70.0	68.8	67.3	65.0
MLKNN	68.7	67.6	66.1	64.3
MLLS	75.9	74.6	72.7	71.5
MLR-L_1	75.4	72.7 4	71.7	69.1
MLR-GL	**76.2**	**75.7**	**75.0**	**74.1**

approximation parameter η are selected by cross validation.

The parameter η approximates $2\text{tr}(\boldsymbol{\gamma}^\top\boldsymbol{\gamma})/(\mathbf{1}_b^\top\boldsymbol{\gamma}^\top\boldsymbol{\gamma}\mathbf{1}_b + \mathbf{1}_a^\top\boldsymbol{\gamma}\boldsymbol{\gamma}^\top\mathbf{1}_a)$, for the matrix $\boldsymbol{\gamma} \in \mathbf{R}^{a\times b}$, where a and b are respectively the number or relevant and irrelevant labels for a training image. As the number of classes increases, we would expect both a and b to increase. Consequently, larger values of a and b would require a larger η value for a better approximation. This is confirmed by the cross validation operation that we performed to choose the η value in our experiments. For example, the selected η value was 50 when we set the number of labels 50 in the training data set, whereas $\eta = 150$ gave the best performance among different values of η tried for the data subset with 500 image labels (for the experiments in Chapter 4). Therefore, we can conclude that the optimal value of η depends on factors like the number of image labels and the nature of the data set (i.e., the average number of labels per image).

5.4.3 Multi-label Ranking Performance on Incompletely Labeled Data

Tables 5.2 and 5.3 show the results for the ESP Game data set in terms of AUC-ROC and MAP, respectively, for a training set with $10,000$ images. We note that the classification results are consistent among experiments with different training set sizes, and only report the results for the $10,000$ images setting results for brevity. From the tables, we first observe that the baseline PLATT, which converts SVM output scores into probabilistic scores, improves the performance of SVM in the missing label settings. This is consistent with [169], where the conversion procedure makes the outputs from different SVM classifiers more comparable and consequently may lead to better performance for multi-label ranking. On the other hand, both SVM and PLATT are outperformed by the direct multi-label learning methods, namely MLR-GL, MLR-L_1, and MLLS; this stresses the importance of developing multi-label ranking methods for multi-label learning.

Second, we observe a significant decrease in classification accuracy for all the methods when moving from case-1 to case-4, proving that the missing class assignment could significantly affect the classification performance. On the other hand, compared to the other baseline methods, the

proposed method (MLR-GL) is more resilient to the missing class labels: it only experiences a drop less than 2% in terms of AUC-ROC metric when 60% of the assigned class labels are removed (case-4), whereas the other methods experience drops from 3% to 5%. Similarly, the performance drop from case-1 to case-4 is less than 3% for MLR-GL, whereas it is more than 5% for the other baselines in terms of MAP score. These results indicate the robustness of the proposed method in handling missing class assignments.

In Table 5.4, we provide results for sample test images from the ESP Game data set for the case-3 experiments, where 40% of the assigned class labels are missing from the training images. We give the label predictions by the baselines for four three images, and the first row under the images gives the true image class labels. For each baseline, we provide the top nine returned labels ranked from left to right and top to bottom. The correct matches are written with bold characters. In addition to the clear superiority of the proposed method's predictions over the other baselines, there is another point that needs to be emphasized. The analysis of the left-most image, whose labels are *silver, circle,* and *round,* shows how using label correlations help to address the label ambiguity problem. We see that the three direct multi-label learning methods, MLR-GL, MLR-L_1, and MLLS, successfully retrieve the label *round* in addition to *circle,* whereas SVM baselines cannot. This is because certain label pairs, such as *circle-round, girl-woman* and *logo-ad,* are mostly retrieved together by the direct multi-label learning methods. This makes these methods more robust for the label ambiguity problem.

We also report the results on the MIR Flickr25000 data set in terms of AUC-ROC score in Table 5.5. Similar to the ESP Game data set, we observe (i) a significant drop in AUC-ROC score for all the methods when some class assignments are missing from training examples, and (ii) MLR-GL experiences the least degradation, together with the MLLS method, in terms of AUC-ROC score compared to the other baseline methods. We also notice that unlike the ESP Game data set, the baseline SVM slightly outperforms the baseline PLATT for the MIR Flickr25000 data set, showing that the probabilistic score conversion does not improve the SVM outputs for this data

set.

To better understand the reasons as to why the proposed MKL-GL is more robust, we observe the outcomes for the training samples after the training/learning step. Table 5.6 shows how different methods perform in finding the missing true labels for training examples, where only the underlined true labels are provided to the learning algorithms. We observe that MLR-GL is able to find more missing labels than the other baselines. Unlike the other baselines, when ranking the label scores for the training images, MLR-GL does not always put the assigned labels at the top of the ranking. In contrast, it ranks some categories that are initially labeled as irrelevant higher than the relevant ones, meaning that MKL-GL does not overfit. This is why the proposed method outperforms the baselines in this task. Table 5.7 shows examples of annotations generated for test images for case-4, where 60% of the positive labels are removed from the training data set. These examples confirm that the proposed method gives better annotation results than the baseline methods.

Based on the above results, we conclude that the proposed method for multi-label learning (i) is effective for image categorization, and (ii) is more effective in handling incompletely labeled data than the state-of-the-art methods for multi-label learning.

5.4.4 Training Time

In Chapter 4, we observed that the MLLS baseline is computationally more efficient than one-vs-all SVM and the MLR-L_1 multi-label ranking method when the number of samples, n, is greater than the number of feature dimensions, d. Therefore, when comparing the proposed MLR-GL method to SVM and MLR-L_1 in terms of training time, we exclude the MLLS baseline from the evaluations. Moreover, we are also not including MLKNN algorithm, which is significantly faster than the other baselines, because it only requires simple and fast operations, such as calculating label prior probabilities. However, MLKNN's efficiency comes with a price of lower classification performance.

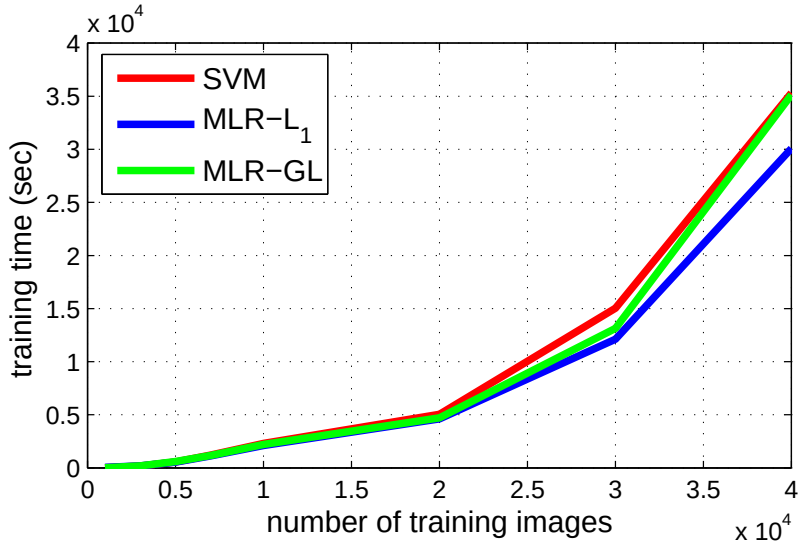

Figure 5.3: The change in the baseline training times (seconds) with respect to the number of training images from the ESP Game data set.

Figures 5.3 and 5.4 plot the change in the baseline training times with respect to the number of training images and labels, respectively. We use the ESP Game data set, and the three baselines that we are comparing are MLR-GL, MLR-L_1, and one-vs-all SVM. All these three methods are implemented with C. In this experiment, we vary the number of training examples from $1,000$ to $40,000$ and labels from 10 to 500. Overall, we observe that the methods in comparison have similar running times. The computational complexity of MLR-L_1 and MLR-GL per iteration is $O(mn^2)$, where n is the number of training examples and m is the number of classes.

Note that the time spent on kernel matrix construction is not included in this study because it is shared by all the three methods in comparison. However, when the RAM capacity is not large enough to store the whole kernel matrix, using a pre-computed kernel matrix would not be possible. This would have a larger negative impact on one-vs-all SVM, since the computational complexity would become $O(dmn^2)$. This is because the kernel function computations need to be performed separately for each class. On the other hand, the computational complexity of the proposed multi-label ranking methods would be $O(dn^2 + mn^2)$, since the classifiers for all labels

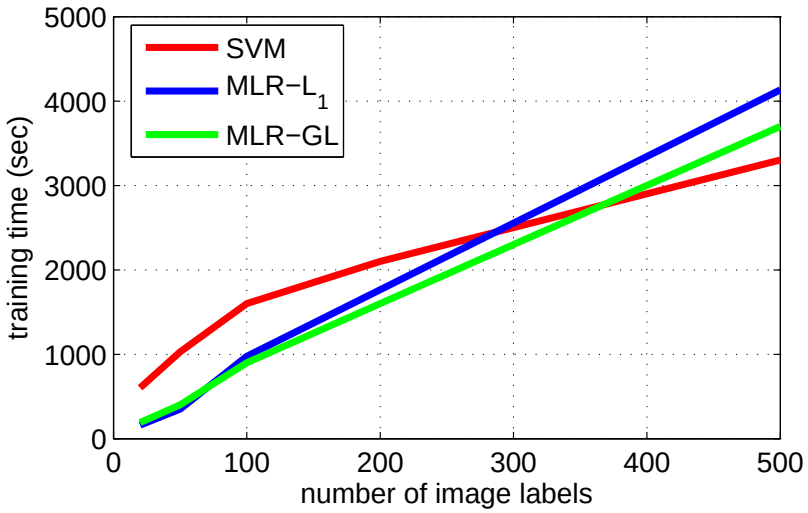

Figure 5.4: The change in the training time (seconds) for the proposed multi-label ranking algorithms and one-vs-all SVM with respect to the number of image labels (m).

are learned together by using a single kernel.

5.5 Conclusions and Future Work

In this chapter, we have presented our multi-label ranking approach which addresses the incomplete class assignment problem. By using the group lasso technique [163] to combine the errors in ranking the assigned classes and unassigned classes, our method is able to use the relationships between the class labels to detect the missing class assignments, making it more robust for incompletely labeled data. Our empirical study of image categorization with two benchmark data sets demonstrated that the proposed method outperforms state-of-the-art methods, particularly when the number of missing label assignments increases in the training set. We can list our contributions as follows:

- We have proposed a multi-label ranking approach which offers a direct solution to multi-label learning unlike the conventional methods that use a set of binary classifiers. Our experiments have shown that the proposed method outperforms the multi-label learning techniques from the literature.

- The proposed method is robust to incomplete class assignment problem. The performance difference

between the proposed method and the multi-label learning baselines increases in favor of the proposed approach as the number of missing class labels in the training set increases.

- We have proposed an efficient algorithm that involves using a closed form solution. The computational complexity is linear with respect to the number of class labels. The computational load of the proposed algorithm is comparable to that of one-vs-all SVM, which is one of the most efficient multi-label learning algorithms. The proposed algorithm can efficiently handle the majority of the available image categorization data sets with tens of thousands of images and hundreds of classes.

The proposed algorithm efficiently and effectively tackles the incomplete class assignment problem. However, there are three main issues that need to be addressed to improve this work further. The first one is extending the proposed framework to multiple kernel learning. Similar to the multi-label ranking approach presented in Chapter 4, the multi-label ranking method we describe in this chapter is limited to a single kernel function usage. Extension of this work to multiple kernel learning setting can bring a significant improvement in classification performance. The second issue is the computational complexity. The current algorithm can handle tens of thousands of samples and hundreds of classes. However, since the computational complexity is linear in the number of class labels and quadratic in terms of training instances, training the proposed algorithm in recent large scale image categorization data sets (millions of images and thousands of class labels) would not be practical. One way to improve the training efficiency of the proposed multi-label ranking algorithm would be incorporating label set space projection methods like compressed labeling [123]. Finally, the proposed method can be extended to the scenario where not only some of the "true" class assignments are missing, but some of the class labels are incorrectly assigned to the training instances. This is a more challenging problem in which we need to address the uncertainty arising from missing class assignment as well as from noisy class assignments. This scenario often encountered in the problem of image tagging/annotation [155].

Table 5.6: Examples of training images from the ESP Game data set with true labels and annotations generated by different multi-label learning methods. Only the underlined true labels are provided to the methods for training. For each method, the correct (returned) keywords are highlighted by bold font whereas the incorrect ones are highlighted by italic font.

Images		
Labels	**brown girl grass green hair picture smile tree**	**blue, building car city cloud sky street white window**
MLR-GL	*man black* **green** *people white* red *woman* **tree** *blue sky* **girl hair picture grass brown** *water light yellow old hat face* **smile** *house shirt eye*	**white** *man* **sky blue** *green red black woman water* **window** *tree people grass hair picture house yellow brown girl* **cloud building** *mountain smile face* **car**
LIBSVM+Platt	**girl green** *blue black face hair woman people white glasses man group* **tree grass** *sky light pink chinese eye red plant dress hand flower forest*	**window city** *black hair man* **white** *water yellow smile chinese line tree* **sky** *lake mountain pink* **blue** *computer wood green table woman boy house hat*
LIBSVM	**green girl** *space drink sky point face woman shop metal family pot machine light truck forest star guy sit glasses white night* **hair** *black usa*	**city window** *metal truck* **car** *ball lake lake* **building** *room fly line wing roof water website mountain road helmet* **white** *tent chinese chair pink silver small*
MLR-L1	**green girl** *black* **tree** *people light* **hair** *man white metal dark band leaf star glasses sky space woman red night truck face street pot group*	**window city** *black* **sky** *water metal mountain pink wing* **car building** *hair boy computer lake truck insect person roof room man tree silver road ocean*

Table 5.7: Examples of test images from the ESP Game data set with annotations generated by different multi-label learning methods. The correct keywords are highlighted by bold font whereas the incorrect ones are highlighted by italic font.

Images		
Labels	**tree water black picture drawing sea art blue boat green city**	**man woman people hair girl picture smile group photo kid family**
MLR-GL	*man white* **black** *woman people* **blue green** *red* **tree** *girl sky* **water** *hair* **picture** *old brown grass yellow face mountain*	**man woman** *black white* **people** *blue green red* **girl** *tree* **hair** *sky water* **picture** *old brown face yellow grass* **smile**
LIBSVM	*book smile gray sun flag computer brick man yellow street machine* **sea** *leaf road ocean couple forest fly purple toy*	**man hair** *black movie face food fire boy* **smile** *lady metal statue dance couple red table toy arm bike gold*
LIBSVM+Platt	*book man smile white* **blue** *sky* **black** *woman red* **green** *people* **tree water** *computer girl face old hair yellow leaf*	*movie food* **man hair** *white* **smile woman** *blue face black* **people** *green red* **girl** *fire tree sky boy table eye*
MLR-L1	**tree green** *hair movie white* **black** *people grass statue leaf orange old bike red flower mountain* **picture** *dance eye dirt*	**hair** *tree black movie green* **man** *eye* **woman** *white hand face* **girl people smile** *dance red hat orange statue brown*

Chapter 6

Multiple Kernel Multi-label Ranking

6.1 Introduction

In this chapter, we present a multiple kernel multi-label ranking (MK-MLR) algorithm for image categorization. The algorithm we propose combines different image representations to make the best multi-label prediction for a query image, by learning to rank relevant labels over irrelevant labels. To achieve this goal and build our algorithm, we combine several conclusions drawn in the previous chapters:

- The experimental results in Chapter 2 showed that, given a sufficient number of training samples, learning a sparse combination of base kernels (MKL-L_1) is advantageous for image categorization. Not only does it often improve accuracy when compared to the average kernel or MKL-L_2 frameworks, but the sparse solutions also lead to a computationally efficient prediction step. Using a smaller number of base kernels as a result of sparsity brings a significant time gain in terms of feature extraction cost; one of the main bottlenecks of the prediction step.

- Among the MKL-L_1 baselines we evaluated in Chapter 2, MKL-SILP (Semi-Infinite Linear Programming) [71] is the most computationally efficient method. MKL-SILP is a wrapper approach, meaning that learning the kernel weights and classification functions can be separated in each iteration. Because of this, the inner SVM-solver can be replaced by other learning algorithms without modifying the linear programming solution that is used for updating the kernel weights.

- In Chapter 4, we formulated image categorization as a multi-label ranking problem. Our experimental results showed that learning classification functions for all the classes in a single framework (i.e., direct multi-label approaches) gives better prediction results compared to decomposing the problem into individual binary classification tasks, i.e., one-vs-all SVM. However, the algorithm we presented in Chapter 4 (MLR-L_1) is designed for using a single kernel. In this chapter, our goal is not only finding the optimal multi-label ranking solution, but also the best linear kernel combination that would maximize multi-label prediction performance.

- The experimental results provided in Chapter 3 showed that, for image classification, there is not a significant performance difference between using one shared kernel combination for all classes and learning a different kernel combination for each class. Therefore, in order to improve the computational efficiency of training and prediction steps, we propose to learn a single kernel combination that would benefit all the classes in a multiple kernel multi-label ranking framework.

Based on these stated conclusions, we extend the MLR-L_1 method by integrating it into a wrapper SILP MKL framework. The goal of developing a multiple kernel multi-label ranking method is to address the two essential factors for improving the performance of image categorization: (i) heterogeneous information fusion, and (ii) exploiting label correlation of multi-label data. The main difference between the algorithm proposed in Chapter 3, ML-MKL-SA, and the MK-MLR algorithm we present in this chapter is that the former aims to improve the training efficiency of MKL for one-vs-all framework. On the other hand, the goal of the MK-MLR algorithm is to improve the image categorization performance by exploiting label dependencies in multi-label data and optimizing the use of different image representations.

This Chapter is organized as follows: in Section 6.2, we provide a literature review on MKL methods that are proposed for multi-label learning. Next, in Section 6.3, we introduce our multiple kernel multi-label ranking formulation and provide a computationally efficient algorithm, which is based on semi-infinite linear programming (SILP), to solve it. In Section 6.4, we provide empirical analyses that demonstrate the strength of the proposed framework on benchmark data sets. We end the chapter with the concluding remarks and future directions in Section 6.5.

6.2 Previous Work

MKL is a very useful tool for the image categorization problem, since an image can be represented in many ways depending on the methods used for key-point detection, descriptor/feature extraction, and key-point quantization; each image representation has different strengths and weaknesses. MKL offers a systematic solution to image feature selection and combination for the image representation and learning problems. However, a vast majority of MKL studies in the literature address the binary classification task. Therefore, the use of MKL for image categorization is mostly limited to one-vs-all framework, which gives suboptimal performance (see Chapter 4). A detailed survey of binary MKL methods is presented in Chapter 2.

We presented a multi-label multiple kernel method (ML-MKL-SA) in Chapter 3. Unlike the one-vs-all scheme, the proposed ML-MKL-SA method does not decompose the multi-label problem into individual binary problems. By learning a common kernel for all classes, ML-MKL-SA takes advantage of multi-label data by sharing information between the classes. However, the classification functions for each class are still trained independently, meaning that label correlations are not used when the classifiers are trained.

One of the main conclusions of Chapter 4 is that direct methods for multi-label learning, which optimize classification functions together, are superior to decomposition based methods such as one-vs-all and one-vs-one. However, there is a limited number of works that extend a direct multi-label learning method to multiple kernel setting in the literature. Kernel multiple linear regression (KMLR) and canonical correlation analysis (CCA) are two techniques that are employed in multi-label learning literature to compute a mapping between data samples and data labels [170]. Yakhnenko et al. extended the kernel regression model and canonical correlation analysis methods to the multiple kernel setting [171]. The authors proposed a reduced gradient method to solve for the optimal linear kernel combination for multi-label learning with KMLR and CCA. Ji et al. [68] proposed a multi-label multiple kernel learning method that can be considered as a generalization to KCCA. The goal of the method they proposed is to embed the data into a low-dimensional space by using a hypergraph, which encodes instance-label correlations. In addition to proposing a SILP solver, they also approximated the problem in order to use Nesterov's method [85].

Zhang et al. used concept networks to model inter-label dependencies and similarity diversities [172]. Inter-label dependencies exploit the similarity between images that share a common label. For a pair of

images that share some common labels but also contain different labels from each other, similarity diversity is used to measure the dissimilarity between these two images. The authors proposed to learn an optimal kernel not only for each label, but also for each label pair in order to utilize the concept networks.

Our method, MK-MLR is the first attempt of extending multi-label ranking to multiple kernel setting. One of the main advantages of MK-MLR compared to other multi-label MKL methods is that MK-MLR exploits label correlations without making explicit assumptions on the data. Moreover, learning one shared kernel combination for all classes is advantageous for classes with small number of positive samples. Since MKL-L_1 methods require a sufficient number of training samples to perform well, sharing a kernel combination, which also means sharing information among different classes, benefits classes with a small number of samples. Finally, by imposing sparsity on the kernel combination vector, the proposed method improves the computational efficiency of training and prediction.

6.3 Multiple Kernel Multi-Label Ranking (MK-MLR)

In this chapter, we use the same notation as in Chapter 3. We introduce $\boldsymbol{\beta} = (\beta_1, \ldots, \beta_s)$, a probability distribution, for combining base kernels. We use the domain $\mathbb{B}_1$ for the probability distribution $\boldsymbol{\beta}$, i.e., $\mathbb{B}_1 = \{\boldsymbol{\beta} \in \mathbb{R}^s_+ : \boldsymbol{\beta}^\top \mathbf{1} = 1\}$. Our goal is to learn from the training examples the optimal kernel combination $\boldsymbol{\beta}$ for all m classes while simultaneously optimizing the corresponding ranking functions.

6.3.1 A Minimax Framework for Multiple kernel Multi-label Ranking

In multiple kernel multi-label ranking, we aim to learn m classification functions $f_k(\mathbf{x}; \boldsymbol{\beta}) : \mathbb{R}^{d_1 \times d_2 \times \ldots d_s} \mapsto \mathbb{R}, k = 1, \ldots, m$, one for each class, such that for any example $\mathbf{x}$, $f_k(\mathbf{x}; \boldsymbol{\beta})$ is larger than $f_l(\mathbf{x}; \boldsymbol{\beta})$ when $\mathbf{x}$ belongs to class c_k and does not belong to class c_l. Note that $f_k(\mathbf{x}; \boldsymbol{\beta})$ is computed by using the kernel function $\kappa(\cdot, \cdot; \boldsymbol{\beta}) = \sum_{s=1}^{K} \beta_s \kappa_s(\cdot, \cdot)$. We define the classification error $\varepsilon_i^{k,l}$ for an example $\mathbf{x^i}$ with respect to any two classes c_k and c_l, as follows

$$\varepsilon_{k,l}^i = I(y_i^k \neq y_i^l) \ell \left(\frac{y_k^i - y_l^i}{2} \left(f_k(\mathbf{x^i}; \boldsymbol{\beta}) - f_l(\mathbf{x^i}; \boldsymbol{\beta}) \right) \right), \tag{6.1}$$

where $I(z)$ is an indicator function that outputs 1 when z is true and zero, otherwise. The loss $\ell(z)$ is defined to be the hinge loss, where $\ell(z) = \max(0, 1 - z)$.

Following the framework in Chapter 4 and the multiple kernel learning problem, we aim to search for the classification functions $f_k(\mathbf{x}; \boldsymbol{\beta}), k = 1, \ldots, m$ that simultaneously minimize the overall classification error. This is summarized into the following optimization problem.

$$\min_{\boldsymbol{\beta} \in \mathbb{B}_1} \min_{\{f_k \in \mathcal{H}_\kappa(\boldsymbol{\beta})\}_{k=1}^m} \frac{1}{2} \sum_{k=1}^{m} |f_k|^2_{\mathcal{H}_\kappa} + C \sum_{i=1}^{n} \sum_{k,l=1}^{m} \varepsilon^i_{k,l}, \tag{6.2}$$

where $\kappa(\mathbf{x}, \mathbf{x}') : \mathbb{R}^d \times \mathbb{R} \mapsto \mathbb{R}$ is a kernel function, $\mathcal{H}_\kappa(\boldsymbol{\beta})$ is a Hilbert space endowed with a kernel function $\kappa(\cdot, \cdot; \boldsymbol{\beta}) = \sum_{s=1}^{K} \beta_s \kappa_s(\cdot, \cdot)$. and C is a regularization parameter. The domain $\mathbb{B}_1$ is defined in Eq. (6.3).

$$\mathbb{B}_1 = \left\{ \boldsymbol{\beta} \in \mathbb{R}^s_+ : \|\boldsymbol{\beta}\|_1 = \sum_{j=1}^{s} |\beta_j| \leq 1 \right\}. \tag{6.3}$$

By using the following definition for $\Delta^i_{k,l}$,

$$\Delta^i_{k,l} = \frac{y^i_k - y^i_l}{2} \langle f_k - f_l, \kappa(\mathbf{x^i}, \cdot) \rangle_{H_\kappa}.$$

We can rewrite the objective function in Eq. (6.2) as follows

$$h(f; \boldsymbol{\beta}) = \frac{1}{2} \sum_{l=1}^{m} \langle f_l, f_l \rangle_{H_K(\boldsymbol{\beta})} + C \sum_{i=1}^{n} \sum_{l,k=1}^{m} I(y^i_l \neq y^i_k) \ell\left(\Delta^i_{k,l}\right).$$

We then rewrite $\ell(z)$ as

$$\ell(z) = \max_{x \in [0,1]} (x - xz).$$

Using the above expression for $\ell(z)$, the second term in $h(f; \boldsymbol{\beta})$ can be rewritten as,

$$\sum_{i=1}^{n} \sum_{l,k=1}^{m} I(y^i_l \neq y^i_k) \max_{\gamma^i_{k,l} \in [0,C]} \left(\gamma^i_{k,l} - \gamma^i_{k,l} \Delta^i_{k,l}\right).$$

Then, the problem in Eq. (6.2) can be rewritten as follows,

$$\max_{\boldsymbol{\beta}\in\mathbb{B}_1} \min_{f_l\in\mathcal{H}(\boldsymbol{\beta})_m} \max_{\gamma^i_{l,k}\in[0,C]} g(f,\gamma,\boldsymbol{\beta}),$$

where

$$\begin{aligned} g(f,\gamma,\boldsymbol{\beta}) &= \sum_{i=1}^{n}\sum_{l,k=1}^{m} I(y^i_l \neq y^i_k)\gamma^i_{l,k} + \frac{1}{2}\sum_{l=1}^{m}\langle f_l, f_l\rangle_{H(\boldsymbol{\beta})_K} \\ &\quad -\sum_{i=1}^{n}\sum_{l,k=1}^{m} I(y^l_i \neq y^i_k)\gamma^i_{l,k}\Delta^i_{k,l}. \end{aligned}$$

Next, we switch the order of minimization over f and maximization over γ. By taking the minimization over f_l first, we have

$$f_l(x;\boldsymbol{\beta}) = \sum_{i=1}^{n} y^i_l \left(\sum_{k=1}^{m} I(y^i_l \neq y^i_k)\gamma^i_{l,k}\right)\kappa(\mathbf{x^i},\mathbf{x};\boldsymbol{\beta}).$$

In the above derivation, we use the relation $I(y^i_l \neq y^i_k)(y^i_l - y^i_k) = 2y^i_l$. To simplify our notation, we introduce $\Gamma_i \in [0,C]^{m\times m}$ where $\Gamma^i_{l,k} = \gamma^i_{l,k}$ if $y^i_l \neq y^i_k$ and zero otherwise. Note that since $\gamma^i_{l,k} = \gamma^i_{k,l}$, we have $\Gamma^i = [\Gamma^i]^\top$. We furthermore introduce the notation $[\Gamma^i]_l$ as the sum of the elements in the lth row, i.e., $[\Gamma^i]_l = \sum_{k=1}^{m}\Gamma^i_{l,k}$. Using these notations, we have $f_l(\mathbf{x};\boldsymbol{\beta})$ expressed as

$$f_l(\mathbf{x}) = \sum_{i=1}^{n} y^i_l [\Gamma^i]_l \kappa(\mathbf{x^i},\mathbf{x};\boldsymbol{\beta}).$$

Finally, the remaining maximization problem becomes

$$\begin{aligned} \min_{\boldsymbol{\beta}\in\mathbb{B}_1}\max_{\Gamma}\quad & \sum_{i=1}^{n}\sum_{k=1}^{m}[\Gamma^i]_k - \frac{1}{2}\sum_{k=1}^{m}\sum_{i,j=1}^{n}\kappa(\mathbf{x^i},\mathbf{x};\boldsymbol{\beta})y^i_k y^j_k [\Gamma^i]_k[\Gamma^j]_k \\ \text{s. t.}\quad & \Gamma^i_{k,l} = \begin{cases} 0 \le \Gamma^i_{k,l} \le C & y^i_k \neq y^i_l \\ 0 & \text{otherwise} \end{cases} \\ & \Gamma^i = [\Gamma^i]^\top, \quad i = 1,\ldots,n; k,l = 1,\ldots,m. \end{aligned}$$

Note that Eq. (6.4) is a generalized version of Eq. (4.4) and also might be expensive to solve, as the number

of constraints is $O(m^2)$, where m is the number of classes. Therefore, we propose a similar approximation.

6.3.2 Proposed Approximation

Without a loss of generality, consider a training example $\mathbf{x^i}$ that is assigned to the first a classes, and is not assigned to the remaining $b = m - a$ classes. According to the definition of Γ^i in (6.4), we can rewrite Γ as

$$\Gamma = \begin{pmatrix} 0 & Z \\ Z^\top & 0 \end{pmatrix}, \tag{6.4}$$

where $Z \in [0, C]^{a \times b}$. Using this notation, variable $\tau_k = [\Gamma^i]_k$ is computed as

$$\tau_k = \begin{cases} \sum_{l=1}^{b} Z_{k,l} & 1 \leq k \leq a \\ \sum_{l=1}^{a} Z_{l,k} & a+1 \leq k \leq m, \end{cases}$$

where $Z_{k,l}$ is an element in Z that is bounded by 0 and C. According to the above definition, for each instance, τ_k is the sum of either the k^{th} column or the k^{th} row of Z depending on whether the label k is relevant to that instance or not. As discussed in Chapter 4, formulating τ_k by using Z enables us to exploit label relationships during the optimization process.

Using Theorem 4 and Corollary 5 from Chapter 4, we introduce the variable α_k^i for $[\Gamma^i]_k$. We furthermore restrict $\boldsymbol{\alpha}^i = (\alpha_1^i, \ldots, \alpha_k^i)$ to be in the domain $\mathcal{G} = \left\{\tau \in [0, C]^m : \sum_{k=1}^{a} \tau_k = \sum_{k=a+1}^{m} \tau_k\right\}$ to ensure that feasible Γ^i can be recovered from a solution of α_k^i. Then, using the vector notation, we can rewrite the new optimization problem for multiple kernel multi-label ranking (MK-MLR) as in Eq. (6.5).

$$\begin{aligned} \min_{\boldsymbol{\beta} \in \mathbb{B}_1} \max_{\boldsymbol{\alpha} \in \mathcal{Q}_1} \quad & \widehat{\mathcal{L}}(\boldsymbol{\alpha}, \boldsymbol{\beta}) = \sum_{k=1}^{m} \left\{ \mathbf{1}^\top \boldsymbol{\alpha}_k - \frac{1}{2} (\boldsymbol{\alpha}_k \circ \mathbf{y}_k)^\top \mathbf{K}(\boldsymbol{\beta}) (\boldsymbol{\alpha}_k \circ \mathbf{y}^k) \right\}, \\ \text{s. t.} \quad & \sum_{k=1}^{m} I(y_k^i = 1) \alpha_k^i = \sum_{k=1}^{m} I(y_k^i = -1) \alpha_k^i, \\ & \alpha_k^i \in [0, C], \quad i = 1, \ldots, n, k = 1, \ldots, m, \end{aligned} \tag{6.5}$$

where $\kappa(\mathbf{x}, \mathbf{x}'; \boldsymbol{\beta}) = \sum_{j=1}^{s} \beta_j \kappa_j(\mathbf{x}, \mathbf{x}')$ and $\mathbb{B}_1 = \left\{ \boldsymbol{\beta} \in \mathbb{R}_+^s : \|\boldsymbol{\beta}\|_1 = \sum_{j=1}^{s} |\beta_j| \leq 1 \right\}$. It is important to

note that the only difference between Eq. (6.5) and the optimization problem of ML-MKL-Sum (Eq. (3.2) in Chapter 3) is the domain defined for $\boldsymbol{\alpha}$.

6.3.3 Optimization via Semi-infinite Linear Programming

One of the conclusions in Chapter 2 was that MKL-SILP (Semi-Infinite Linear Programming) [71] is the most efficient method among the MKL-L_1 baselines. Therefore we will use SILP to optimize Eq. (6.5).

Let's define $S_s(\boldsymbol{\alpha}) = -\sum_{k=1}^{m} \left\{ \mathbf{1}^\top \boldsymbol{\alpha}_k - \frac{1}{2}(\boldsymbol{\alpha}_k \circ \mathbf{y}_k)^\top \mathbf{K}_s (\boldsymbol{\alpha}_k \circ \mathbf{y}_k) \right\}$. Then, we can rewrite Eq. (6.5) as the following min-max problem,

$$\max_{\boldsymbol{\beta} \in \mathbb{B}_1} \min_{\boldsymbol{\alpha} \in \mathcal{Q}_1} \quad \sum_{s=1}^{K} \beta_s S_s(\boldsymbol{\alpha}), \tag{6.6}$$

For the optimal solution $\boldsymbol{\alpha}^*$, $\theta^* = S(\boldsymbol{\alpha}^*, \boldsymbol{\beta})$ would be minimal, meaning that $S(\boldsymbol{\alpha}, \boldsymbol{\beta}) \geq \theta$ for any $\boldsymbol{\alpha} \in \mathcal{Q}_1$. Therefore, as proposed in [71], we need to solve the following SILP problem in order to find a saddle-point of Eq. (6.6).

$$\begin{aligned} \min_{\theta \in \mathbb{R}, \boldsymbol{\beta} \in \mathbb{B}_1} \quad & \theta \\ \text{s. t.} \quad & \sum_{j=1}^{s} \beta_j \{ -\alpha^\top \mathbf{1} + \frac{1}{2}(\boldsymbol{\alpha}_k \circ \mathbf{y}_k)^\top \mathbf{K}_j (\boldsymbol{\alpha}_k \circ \mathbf{y}_k) \} \geq \theta, \\ & \sum_{k=1}^{m} I(y_k^i = 1) \alpha_k^i = \sum_{k=1}^{m} I(y_k^i = -1) \alpha_k^i, \\ & \alpha_k^i \in [0, C], \quad i = 1, \ldots, n, k = 1, \ldots, m. \end{aligned} \tag{6.7}$$

MKL-SILP is a wrapper method, meaning that learning the kernel weights and classification functions can be separated in each iteration of the optimization process. In this chapter, we use the MKL-SILP method with two modifications. Note that, unlike the binary MKL-SILP or ML-MKL-Sum formulations, we cannot use an off-the-shelf SVM solver to maximize Eq. (6.5) with respect to $\boldsymbol{\alpha}$ because of the domain definition. Instead, we need to replace the SVM solver with the MLR-L_1 method that we proposed in Chapter 4. In

addition, compared to binary MKL-SILP, the number of constraints in each step increases since each class generates its own constraints.

In order to optimize Eq. (6.7), we use the column generation method that is used in [71] and [116] to solve the MKL-SILP problem: In an alternating optimization process, the optimal $(\boldsymbol{\beta}, \theta)$ are calculated for a restricted set of constraints. Then, for fixed a $\boldsymbol{\beta}$, new constraints that are determined by $\boldsymbol{\alpha}_k$, $k = 1, \ldots, m$ are generated. This step corresponds to solving for the optimal $\boldsymbol{\alpha}$ for fixed a $\boldsymbol{\beta}$. Therefore, Eq. (6.7) can be solved by simply replacing the SVM solver within the off-the-shelf MKL-SIP solvers (Shogun, ML-MKL-Sum) with the MLR-L_1 algorithm, which is presented in Chapter 4.

6.4 Experimental Results

In this section, we empirically evaluate the proposed multiple kernel multi-label ranking algorithm by comparing it to other MKL baselines for the image categorization task.

6.4.1 Data Sets

In order to compare our proposed multi-label learning method to state-of-the-art MKL methods, we use two benchmark multi-label data sets that we have discussed in Section A.1.6.

The MIR Flickr25000 data set [154] is a subset of the MIR Flickr-1M data set that contains 25,000 images and 457 image tags. We followed [101] and created 15 sets of low level-features: (i) GIST features [102]; (ii) six sets of color features generated by two different spatial pooling layouts [103] (1×1 and 3×1) and three types of color histograms (i.e., RGB, LAB, and HSV). (iii) eight sets of local features generated by two key-point detection methods (i.e., dense sampling and Harris-Laplacian [104]), two spatial layouts (1×1 and 3×1), and two local descriptors (SIFT and robust hue descriptor [105]). A RBF kernel function with χ^2 distance was applied to each of the 15 feature sets. In addition to these 15 low-level features, we extracted 177 different kinds of object banks [173], which encode semantic and spatial information regarding an image. Each object bank is a 256-dimensional vector, which is a collection of response-maps of pre-trained generic object detectors.

In order to test how different baselines perform with respect to the numbers of training images, we created training subsets with different sizes (%2, %5, %25, and %50 of the whole data set). Also, after ranking the categories (image tags) in terms of their frequency (number of images annotated with them), we picked the top 200 categories for multi-label learning evaluation. The number of test samples is $12,500$.

ESP Game data set The second data set we use in this chapter is a subset of the ESP Game data set. We computed nine base kernels by using low level features. The first kernel is based on dense-SIFT descriptors and a Bag-of-Words model with $1,000$ visual words. In addition to dense sampling, we also used the Harris-Laplacian (HarLap) [104] method for key-point detection. For HarLap based Bag-of-Words model, we created two visual dictionaries with sizes 250 and $1,000$ and used two types of spatial pyramid kernels (i.e., 1×1 and 2×2 spatial partitioning), leading to 4 different base kernels. We also created color histograms, each with $4,096$ bins, by using three different color spaces, namely RGB, LAB and HSV. Finally, we constructed a base kernel by using GIST features [102]. In addition to these low level features, we extracted 177 different kinds of object banks for ESP Game data set. In total, we have 186 base kernels for the ESP Game data set.

To study the influence of the number of training samples and labels on multi-label learning performance, we varied the number of training samples and number of labels for the ESP Game data set as well. We created four subsets of the training data (with $\{1,000, 2,500, 5,000, 10,000\}$ images). Also, after ranking the categories in terms of their frequency (number of images annotated with them) in the data set, we picked the top $\{20, 50, 100, 200, 500\}$ categories to create five different test settings in terms of the number of classes. The number of test images is set to 5,000.

6.4.2 Baseline Methods

Following the experiments in Chapter 3, we compare the proposed MK-MLR with four MKL methods, two single kernel baselines, and two average kernel baselines (AVG-SVM and AVG-MLR). The single kernel baselines are the single kernel one-vs-all SVM scheme (SK-SVM) and the single-kernel multi-label ranking method (SK-MLR) that we presented in Chapter 4 (as MLR-L_1). We ran these two methods for each base kernel separately and reported the results for the kernel with the highest score.

The MKL baselines can be categorized into two groups. The first group is the one-vs-all MKL framework, which requires solving one MKL problem separately for each class. For this group, we use two base MKL solvers that are shown to be the most efficient wrapper MKL methods in Chapter 2 : (i) SILP (semi-infinite linear programming) solver for MKL-L_1 [71], and (ii) SIP (semi-infinite programming) solver for MKL-L_2. The second group of methods requires learning a single kernel combination simultaneously for all classes. The two baseline methods that fall into this group are: (i) ML-MKL-Sum, which learns a kernel combination shared by all classes using the optimization method in [116], and (ii) ML-MKL-SA method: A stochastic sampling based algorithm we presented in Chapter 3. Note that all the baselines except MK-MLR, AVG-MLR, and SK-MLR are based on the one-vs-all framework.

6.4.3 Implementation

The experiments were run on a cluster where each node has two four-core Intel Xeon E5620s at 2.4 GHz with 24 GB of RAM. Since the number of kernels is not small (192 for MIR Flickr25000 and 186 for ESP Game), we did not store and use pre-computed the kernel matrices. Instead, all MKL baselines worked with on the fly kernel computation.

All the baseline methods were coded in MATLAB. For the SVM based MKL wrapper methods, we used LIBSVM [107] as the off-the-shelf SVM solver. MOSEK [89] is used for solving the related optimization problem for MKL-SIP, as suggested in [52].

For kernel based methods, we used the RBF kernel in our experiments. The regularization parameter C is chosen with a grid search over $\{10^{-4}, 10^{-1}, \ldots, 10^{3}\}$. The bandwidth of the RBF kernel is set to the average pair-wise Euclidean distance between the training image pairs.

6.4.4 Evaluation Measures

To evaluate the effectiveness of different algorithms for multiple kernel multi-label learning, we first vary the number of selected categories and report the Area under ROC curve (AUC) over the selected classes. This procedure is named as category based evaluation (see appendix, Section A.1.5 for details), in which we rank test images for each class and the evaluation is performed on each label independently, before their

Table 6.1: The change of category based AUC score (%) withe respect to the number of selected classes for a subset of the ESP Game data set with 2,500 training images.

	number of classes			
	50	100	200	500
SK-SVM	70.40	70.00	69.85	69.01
SK-MLR	71.84	71.32	70.55	70.04
AVG	75.86	75.61	75.43	73.66
MKL-L_1	77.07	76.12	75.60	73.10
MKL-L_2	76.43	76.05	75.78	73.19
ML-MKL-Sum	76.86	76.22	76.05	73.62
ML-MKL-SA	77.26	76.53	76.33	73.89
AVG-MLR	76.06	76.02	76.11	73.66
MK-MLR	**78.39**	**77.69**	**77.58**	**74.87**

average is taken over all classes. We also use image based evaluation, particularly for comparing multi-label ranking performance. Image based MLR-AUC measures show how accurate is the ranking of outcomes. In addition, we evaluate the training efficiency of algorithms by the level of sparsity, training and prediction times (seconds).

6.4.5 Multi-label Learning Performance

We list the category and image based AUC results results for the ESP Game data set in Tables 6.1 and 6.2, respectively. The results in these two tables are obtained by varying the number of classes for the setting in which $2,500$ images are used for training. For instance, in the setting where the number of classes is 200, we calculate the AUC score for the top 200 classes (column 3) after ranking them based on the number of positively labeled images they have. We draw the following conclusions from Table 6.1:

- Multiple kernel algorithms consistently outperform single kernel algorithms.
- Learning a sparse combination of base kernels via MKL-L_1 gives better results compared to the average kernel and MKL-L_2 methods.
- Learning one shared kernel combination for all classes does not cause a significant performance drop.

Table 6.2: The change of image based AUC score (%) withe respect to the number of selected classes for a subset of the ESP Game data set with 2,500 training images.

	number of classes			
	50	100	200	500
SK-SVM	75.95	76.32	76.68	76.14
SK-MLR	77.73	78.97	80.41	79.90
AVG	80.81	81.44	82.06	81.67
MKL-L_1	81.67	81.85	82.06	81.50
MKL-L_2	79.09	80.14	81.24	80.82
ML-MKL-Sum	81.51	81.84	82.22	81.78
ML-MKL-SA	81.67	81.99	82.40	81.93
AVG-MLR	81.86	82.97	84.10	83.47
MK-MLR	**83.28**	**84.04**	**84.93**	**84.68**

Table 6.3: The change of category based AUC score (%) withe respect to the number of selected classes for a subset of the MIR Flickr data set with 6,250 training images.

	number of classes			
	50	100	200	500
SK-SVM	65.14	64.83	63.75	62.16
SK-MLR	65.67	65.36	64.52	63.20
AVG	70.31	68.45	66.93	64.88
MKL-L_1	70.98	69.03	66.96	64.98
MKL-L_2	70.83	68.86	67.24	65.31
ML-MKL-Sum	71.00	69.53	67.93	65.97
ML-MKL-SA	71.28	69.83	68.21	66.05
AVG-MLR	**72.10**	**70.16**	68.35	66.30
MK-MLR	**72.28**	**70.34**	68.25	66.44

Table 6.4: The change of image based AUC score (%) withe respect to the number of selected classes for a subset of the MIR Flickr data set with 6,250 training images.

	number of classes			
	50	100	200	500
SK-SVM	63.82	62.94	62.28	62.01
SK-MLR	64.67	63.96	63.35	62.88
AVG	72.89	71.99	71.10	70.69
MKL-L_1	73.57	72.70	71.53	71.08
MKL-L_2	73.13	72.35	71.64	70.71
ML-MKL-Sum	73.37	72.58	71.62	70.60
ML-MKL-SA	73.60	72.91	71.95	70.88
AVG-MLR	**75.26**	**74.23**	**73.71**	**72.75**
MK-MLR	**75.26**	**74.40**	**73.70**	**72.91**

- Although the proposed multi-label ranking method is not designed to optimize category based evaluation measures, it still gives comparable results to MKL-L_1 and outperforms the remaining baselines.
- The proposed MK-MLR method clearly outperforms SK-MLR and AVG-MLR baselines, demonstrating the effectiveness of multiple kernel learning for multi-label ranking.

The results on Table 6.1 are calculated by performing category based evaluation. A better way to evaluate multi-label raking performance is using image based evaluation: ranking each label given a test image. By increasing the number of retrieved labels per image, we can obtain a sequence of true positive and false positive rates and calculate AUC values. Since the proposed MK-MLR method optimizes ranking loss, as expected, it outperforms the other baselines (see Table 6.2). Also note that, compared to the other baselines, the relative performance of all the multi-label ranking methods (MK-MLR, SK-MLR, and AVG-MLR) increases, showing that multi-label ranking methods benefit from a larger number of labels. Another conclusion we draw from Table 6.2 is that multiple kernel methods outperform their single kernel counterparts.

Although the proposed method outperforms the other baselines in terms of the AUC score, it might not be clear how much impact this difference in the AUC score would make in a retrieval system. In order to get a better understanding of the classification accuracies (recall), we plot the classification accuracies of different baselines vs. the number of retrieved labels (rank) in Figure 6.1. To generate this plot we increase the number of retrieved images from 5 to 30 (the maximum number of labels per image is 30 in the subset

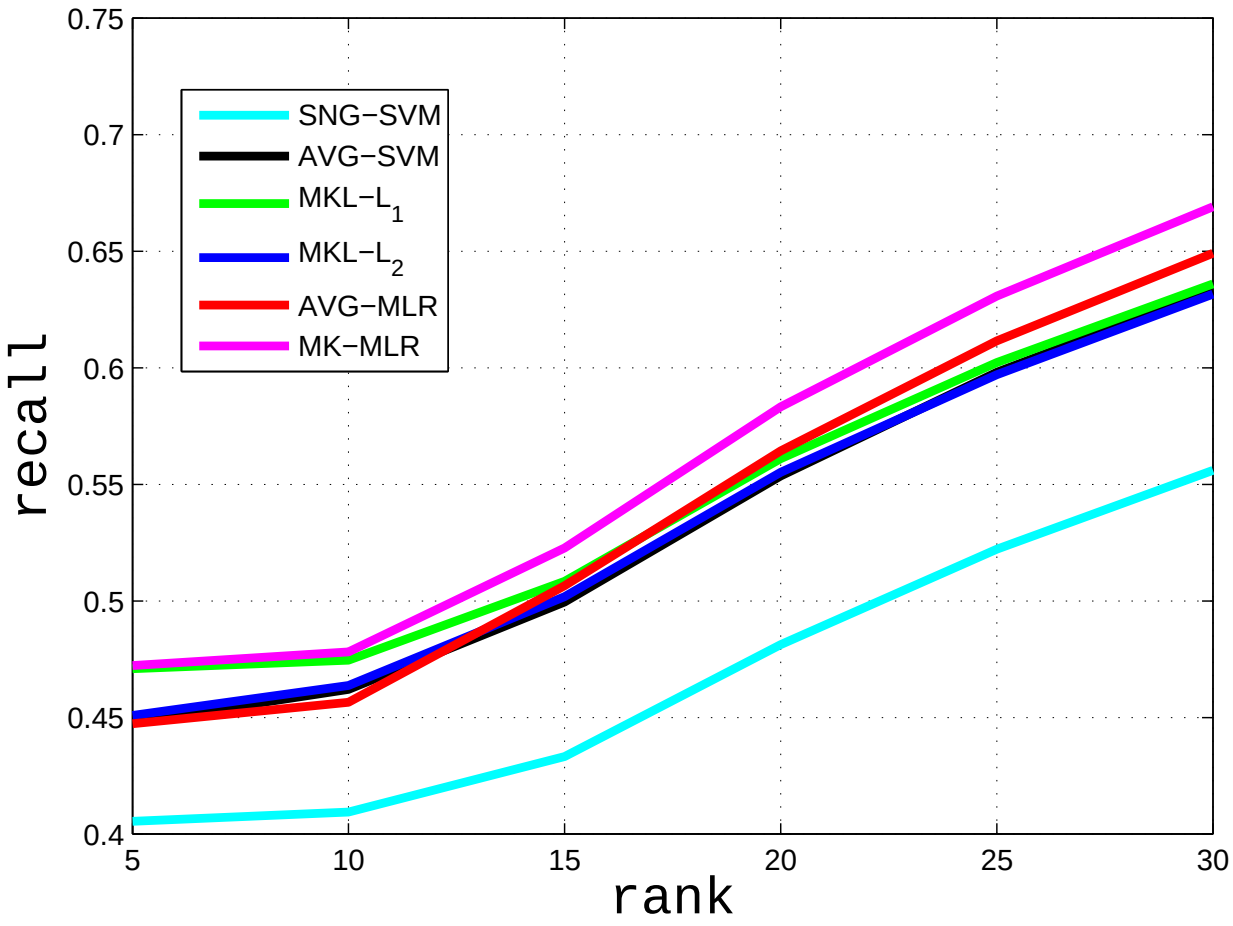

Figure 6.1: The plot of recall vs. number of retrieved labels per image. The number of training images is $2,500$.

we are using).

We see from Figure 6.1 that MLR methods, both AVG-MLR and MK-MLR, yield superior performance compared to the other baselines. In fact, the accuracy of MK-MLR is 2-3% better than that of AVG-MLR and at least 4-5% better than the remaining baselines.

In order to see how the image based AUC score changes with respect to number of samples, in Tables 6.5 and 6.6 , we report AUC and MAP scores for the top 200 classes in four settings, with different subsets of the training data with $\{1,000, 2,500, 5,000, 10,000\}$ images.

The following conclusions can be made from Tables 6.5 and 6.6 :

- MK-MLR method is not outperformed by any other baseline in any setting. In fact, the proposed

Table 6.5: The change of category based AUC score (%) with respect to the number of training samples for a subset of the ESP Game data set. The AUC score is calculated using the top 200 classes.

	number of training samples			
	1,000	2,500	5,000	10,000
SK-SVM	67.45	69.85	70.13	70.01
SK-MLR	68.14	70.71	71.02	70.85
AVG-SVM	72.09	75.43	77.71	79.11
MKL-L_1	72.27	75.60	77.57	79.36
MKL-L_2	72.40	75.78	77.92	80.23
ML-MKL-Sum	72.69	76.05	78.03	80.56
ML-MKL-SA	72.85	76.33	78.87	81.02
AVG-MLR	72.62	76.11	78.27	80.90
MK-MLR	**74.12**	**77.58**	**79.48**	**81.61**

Table 6.6: The change of image based AUC score (%) with respect to the number of training samples for a subset of the ESP Game data set. The AUC score is calculated using the top 200 classes.

	number of training samples			
	1,000	2,500	5,000	10,000
SK-SVM	75.97	76.68	78.31	78.69
SK-MLR	79.95	80.41	82.49	83.27
AVG-SVM	80.41	82.05	84.21	84.82
MKL-L_1	80.52	82.06	84.01	84.99
MKL-L_2	80.78	81.23	84.36	85.01
ML-MKL-Sum	80.79	82.21	83.07	83.86
ML-MKL-SA	80.93	82.79	83.82	84.80
AVG-MLR	82.25	84.10	85.35	86.05
MK-MLR	**83.08**	**84.93**	**85.87**	**86.45**

Table 6.7: The change of category based AUC score (%) with respect to the number of training samples for a subset of the MIR Flickr data set. The AUC score is calculated using the top 200 classes.

	number of training samples			
	500	1,250	6,200	12,500
SK-SVM	59.72	62.23	63.75	64.51
SK-MLR	60.31	62.41	64.19	64.97
AVG-SVM	60.46	64.02	66.93	67.85
MKL-L_1	61.14	64.93	66.96	68.34
MKL-L_2	60.76	64.32	67.24	68.59
ML-MKL-Sum	62.20	65.71	67.93	69.23
ML-MKL-SA	62.21	65.78	68.21	69.61
AVG-MLR	61.11	65.02	68.35	69.97
MK-MLR	**63.06**	**66.89**	**68.85**	**70.33**

MK-MLR algorithm significantly outperforms the competing algorithms in the majority of the experimental settings.

- Using multiple kernels improves the performance.
- All baselines experience an increase in their performance when the number of training instances increases.

We provide the category based and image based AUC scores for the MIR Flickr25000 data set in Tables 6.7 and 6.8. We vary the number of training samples to see how the increase in the training data set size affects the performance. One thing to observe from these two tables is that the performance of the baselines is overall worse compared to the ESP Game data set experiments, particularly when the number of training images is small. Because of this reason, the performance gap between the baselines is not as high as it is for the ESP Game experiments. Further, we can make the following statements based on the results in Tables 6.7 and 6.8.

- MKL methods that learn a single kernel combination for all classes (ML-MKL-Sum and ML-MKL-SA) give slightly better results than training MKL for each class separately (MKL-L_1 and MKL-L_2) for the MIR Flickr25000 data set.

Table 6.8: The change of image based AUC score (%) with respect to the number of training samples for a subset of the MIR Flickr data set. The AUC score is calculated using the top 200 classes.

	number of samples			
	500	1,250	6,250	12,500
SK-SVM	63.89	64.99	65.57	67.81
SK-MLR	64.76	67.83	68.21	69.12
AVG-SVM	65.06	68.26	71.10	71.80
MKL-L_1	66.11	69.29	71.53	71.99
MKL-L_2	65.40	68.59	70.94	71.86
ML-MKL-Sum	67.13	70.12	71.62	72.13
ML-MKL-SA	67.16	70.18	71.95	72.54
AVG-MLR	66.40	68.84	**73.71**	75.26
MK-MLR	**68.12**	**70.93**	**73.70**	**75.91**

- The performance difference between MK-MLR and AVG-MLR decreases as the number of training samples increases. As we have previously discussed in Chapter 2, this is because the quality of all the base kernels increases with an increased number of training samples, and the advantage that a sparse combination would bring, i.e., eliminating weak kernels, vanishes.

- MLR algorithms always perform better than their OvA counterparts, i.e, SK-MLR performs better than SK-SVM; AVG-MLR outperforms AVG-SVM.

6.4.6 Training Efficiency

In this section, we compare the computational efficiency of the MK-MLR algorithm to the other MKL baselines in terms of training times. We group the MKL algorithms into two categories: (i) ML-MKL for learning individual kernel combination for each class, (ii) ML-MKL for learning shared kernel combination. We report the training times for each method under various experimental setting with different number of training samples and classes.

Figure 6.2 and 6.3 compare the training times of the MKL baselines for a fixed number of training set size, $5,000$, under four settings with increasing class numbers: $\{50, 100, 200, 500\}$. It is clear from Figure 6.2 that the proposed method is significantly faster than MKL methods that require learning a separate

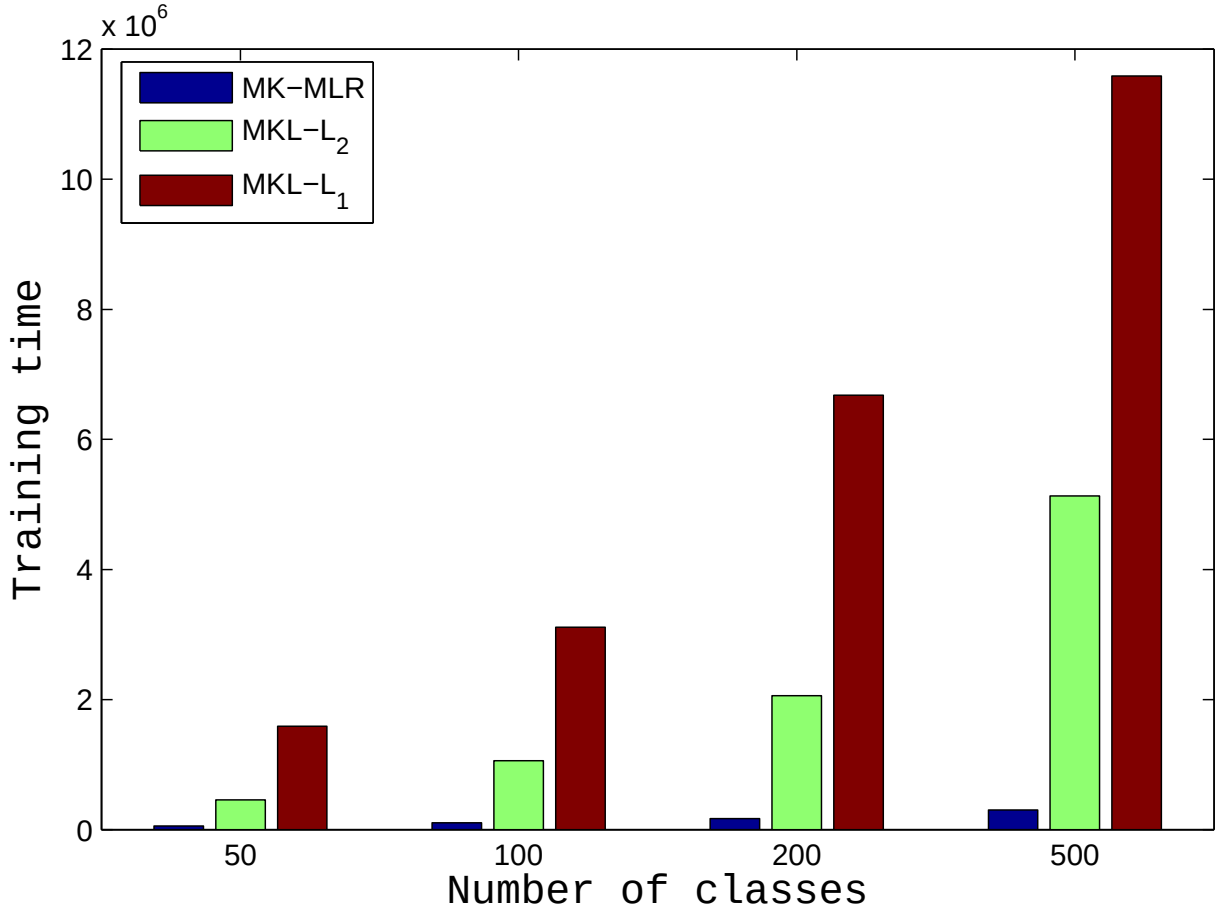

Figure 6.2: Comparing MK-MLR to ML-MKL methods that learn optimal kernel combination separately for each class in terms of training time. We use $5,000$ training images and create four different settings by changing the number of classes $\{50, 100, 200, 500\}$

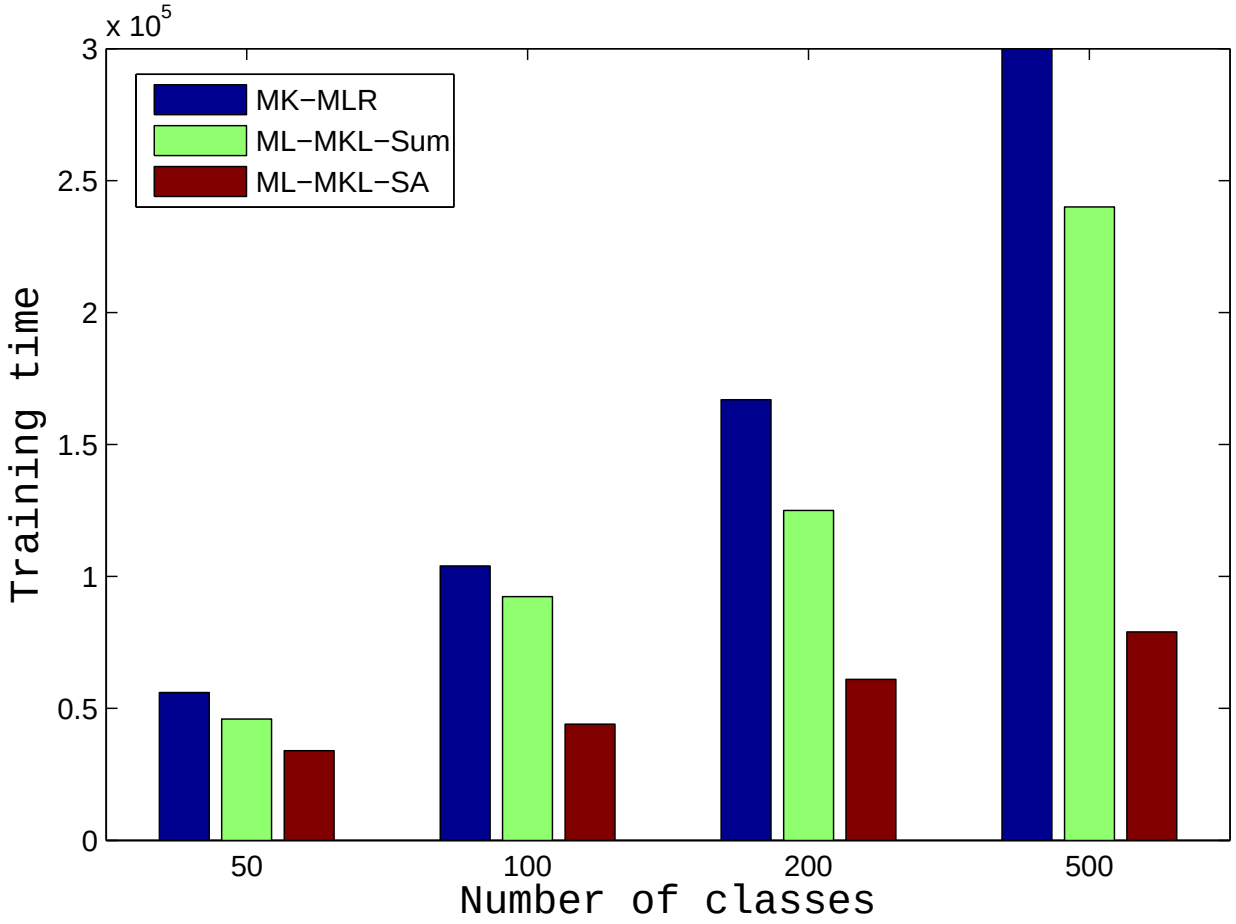

Figure 6.3: Comparing MK-MLR to ML-MKL methods that learn one optimal kernel combination for all classes in terms of training time. We use $5,000$ training images and create four different settings by changing the number of classes $\{50, 100, 200, 500\}$

kernel combination for each class (MKL-L_1 and MKL-L_2). The main advantage of the proposed method is that it avoids repetitiously performing expensive kernel construction and combining operations for each class. The computational complexity of kernel construction is $O(dn^2)$, where d is the dimension of feature vectors and n is the number of samples. When the number of classes and base kernels is larger (order of hundreds), MK-MLR has a significant advantage over these methods.

We also see from Figure 6.3 that MK-MLR is slower than the two MKL baselines which learn one shared kernel combination for all classes. In Chapter 3, we proved that the computational complexity of ML-MKL-SA is sublinear, $O(m^{1/3}\sqrt{lnm})$, in terms of the number of classes, m. Therefore, it is not surprising to see that ML-MKL-SA is the fastest method. Moreover, we can expect the gap between the training times to increase as the number of classes increases. The reason for the performance gap between MK-MKL and ML-MKL-Sum, which use the same SILP solver for kernel weights, is the implementation difference of the dual variable optimizers. Recall from Chapter 4 that our multi-label ranking method and kernel SVM show very close performance in terms of computational complexity and yield almost equivalent training times when implemented in the same environment. On the other hand, since we use a MATLAB implementation for the MLR algorithm, MK-MLR algorithm gives higher training times compared to ML-MKL-Sum, which uses a very efficient SVM solver that is coded with C. However, note that the performance difference does not increase as the number of training sample increases, since these two methods have the same complexity.

Figures 6.4 and 6.5, which compare the training times of the baselines over different data set sizes, $\{1,000, 2,500, 5,000\}$, confirm the conclusions we drew from Figures 6.2 and 6.3. MKL-L_1 and MKL-L_2 methods are significantly slower, since they require expensive kernel computation and combination operations for each class. In addition, both ML-MKL-SA and ML-MKL-Sum methods are faster than MK-MLR. However, ML-MKL-SA does not have a computational advantage as it did when the comparison was made in terms of the change in the number of classes. All the baselines have similar dependency to the number of samples. Therefore, we see a similar growth in training times for them.

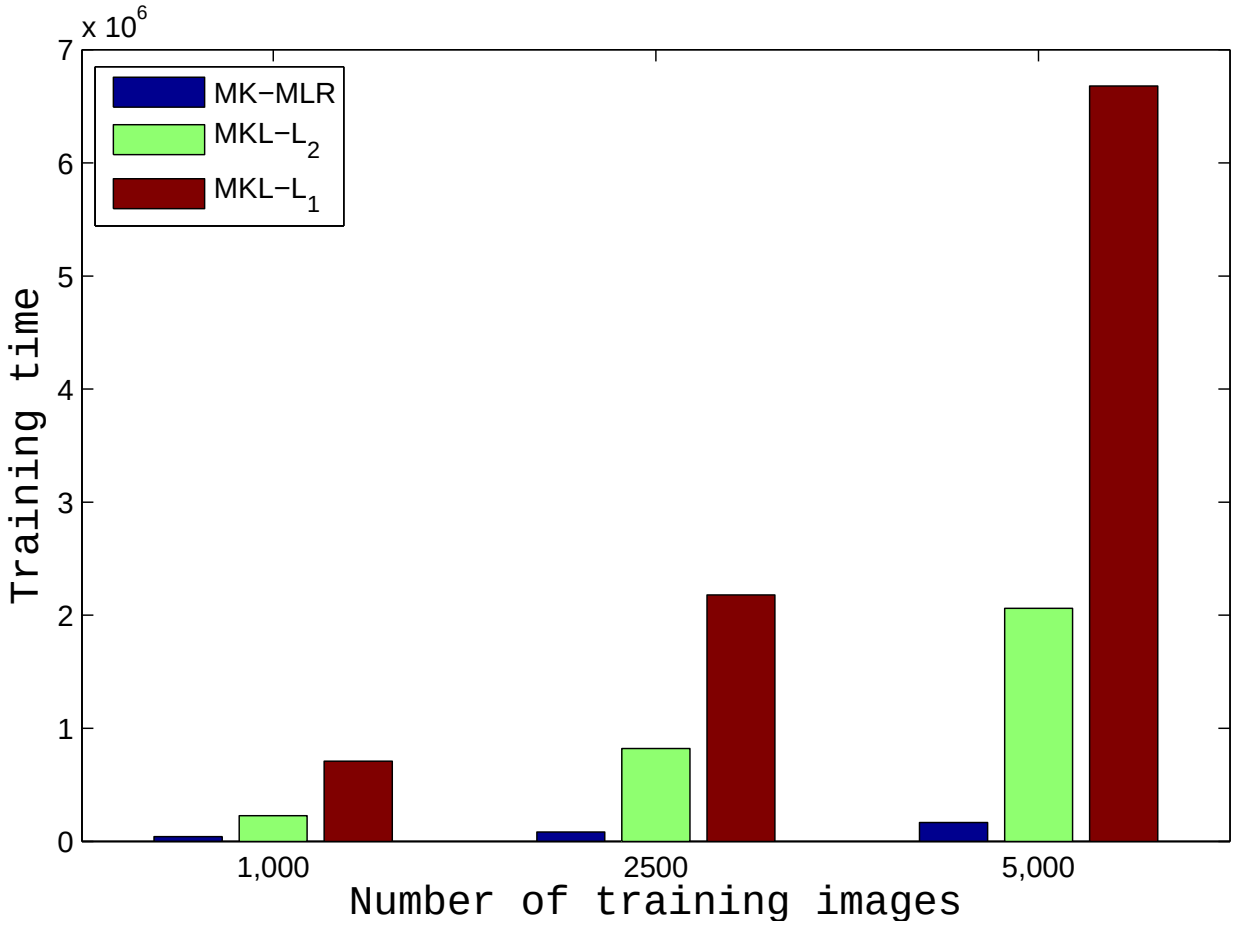

Figure 6.4: Comparing MK-MLR to ML-MKL methods that learn one optimal kernel combination separately for each class in terms of training time. We use images from 200 classes and create three settings by changing the data set size $\{1,000, 2,500, 5,000\}$

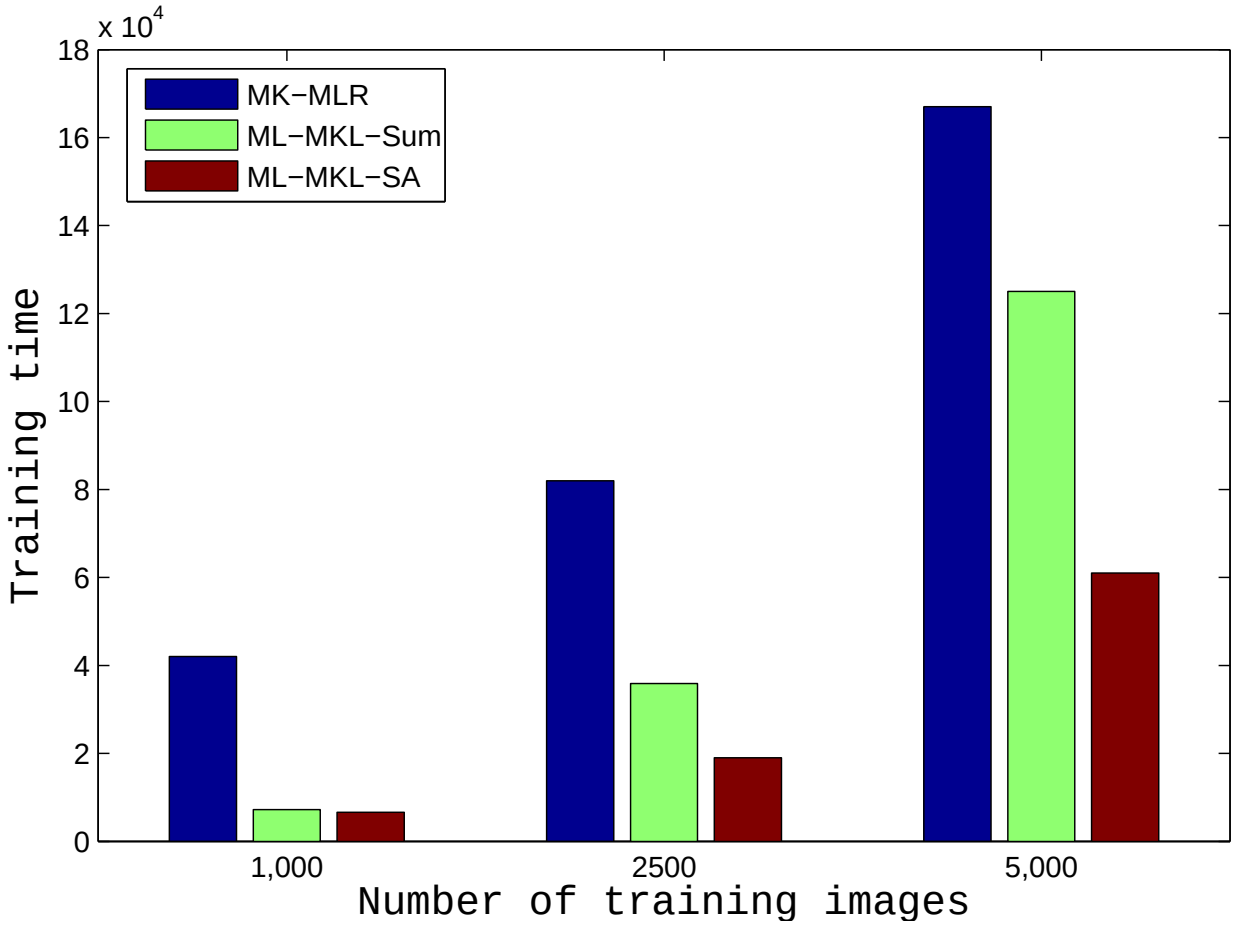

Figure 6.5: Comparing MK-MLR to ML-MKL methods that learn one optimal kernel combination for all classes in terms of training time. We use images from 200 classes and create three settings by changing the data set size $\{1,000, 2,500, 5,000\}$

6.4.7 Prediction Efficiency

Prediction speed is in general more crucial than training speed in real word systems. Given a query image, a multi-label prediction system requires calculating an output score for each class. For multiple kernel setting, an output score for class k can be computed as,

$$f_k(\mathbf{x}) = \sum_{i=1}^{n} \alpha_k^i . \kappa(\mathbf{x}_i, \mathbf{x}; \boldsymbol{\beta}_k),$$

where $\kappa(.,.;\boldsymbol{\beta}_k)$ is the optimal kernel function (linear combination of the base kernels) for class k. Since the computation of output function score is standard for all baselines that use multiple kernels, only the following three factors affect the prediction speed:

- Multi-label kernel combination: Do output functions for each class require a different kernel combination, or do they share a single kernel combination function?
- Sparsity of kernel combination weights.
- Sparsity of output functions.

Therefore, in addition to reporting the actual prediction times, we also discuss these factors to get a better understanding of the prediction efficiency. For a fixed number of training samples ($5,000$) and classes (200), we report the sparsity of kernel weights and dual variables in Table 6.9 for the multiple kernel baselines. We also compute two types of prediction times, both reported in seconds: (i) Average prediction time per single class, (ii) Total prediction time. Note that the average prediction time per class is not calculated simply by dividing the total prediction time by the number of classes, but it is the time to make a prediction if there was only one class needed (binary prediction). When the prediction scores for all classes need to be computed, the prediction time does not increase linearly, since feature extraction, which is the most time-consuming step, can be done once for all classes. An analysis on the sparsity values leads to the following conclusions:

- The main bottleneck for prediction time is the feature extraction step. The time it takes to extract all 186 features that are being used for the ESP Game data set is 10.29 seconds per image. Therefore,

the level of sparsity in kernel coefficients vector is the major factor determining the prediction time efficiency.

- The feature extraction time is not uniform among the features we use. Dense-SIFT based BoW representation is the one that takes the most time with 1.08 seconds. On the other hand, the average time to compute an object bank feature vector is 0.04 seconds. Therefore, sparseness by itself is not the only factor that affects the feature extraction time. For instance a less sparse solution that excludes dense-SIFT based BoW representation from the final feature combination might be more efficient than a sparser solution that requires using dense-SIFT.

- AVG-SVM and MKL-L_2 methods employ all base kernels, meaning that they require extracting all feature types. Since feature extraction is one of the most expensive steps of prediction, having a non-sparse kernel coefficient vector makes AVG-SVM and MKL-L_2 slower in prediction, compared to the methods that learn a sparse kernel combination vector.

- The average sparsity of kernel combination weights over all classes is 87.77% for MKL-L_1, making it the method with the fastest prediction step for a single class. However, when the kernel weights for all classes are considered together, we see that only 21 of the base kernels are not used for any class prediction function. Therefore, although individual binary classifiers have sparse kernel combinations, the overall multi-label prediction sparsity is 11.29% for MKL-L_1, making it significantly slower than methods that use a single kernel combination, namely MK-MLR and ML-MKL-Sum, when all the classes need to be evaluated.

- The average sparsity of kernel combination weights that are learned throughout the ML-MKL-SA is 51.58% per iteration. However, since the final kernel combination is the mean of all previous kernel combination weights, the final sparsity becomes 11.29%, which is significantly lower compared to the ML-MKL-Sum and MK-MLR methods.

- MK-MLR outputs a very sparse kernel combination. Because of this, MK-MLR enables a fast prediction by avoiding unnecessary feature extraction and kernel construction steps.

Table 6.9: Sparsity (%) of kernel weights and dual variables for the multiple kernel baselines and the resulting prediction times. These results are obtained from a subset of the ESP Game data set with $5,000$ training images and 200 classes.

	AVG-SVM	MKL L_1	MKL L_2	ML-MKLSum	ML-MKLSA	MK-MLR
Sparsity($\boldsymbol{\beta}$)	0	87.77	0	77.42	11.29	76.88
Sparsity($\boldsymbol{\alpha}$)	60.55	61.72	59.53	57.88	60.81	47.11
Avg. pred. time per class	10.71	3.74	5.31	5.31	10.69	4.94
Total pred. time	11.38	10.76	11.30	5.58	11.33	5.11

- All OvA based MKL methods produce similar sparsity percentages for dual variables. Although the sparsity of the proposed MK-MLR method is around 10% lower than others, MK-MLR also yields a sparse support vector set. Sparsity of the support set is crucial for reducing storage requirements, kernel construction, and output function calculation costs. However, its impact is much smaller compared to the sparsity of the kernel combination weights in our experimental settings.

6.5 Conclusions and Future Work

In this chapter, we presented an efficient multiple kernel multi-label ranking method by putting together different ideas from the previous chapters. Our experiments in Chapter 4 showed that formulating image categorization as a multi-label ranking problem leads to superior performance compared to more widely-used formulations such as binary decomposition (e.g., OvO and OvA). Therefore, we extended multi-label ranking to multiple kernel setting and proposed the MK-MLR algorithm. Following the conclusions of Chapter 3, we proposed to learn a shared kernel combination for all classes. This approach improves the computational efficiency of both the training and prediction steps significantly. MK-MLR algorithm learns kernel weights and class output functions simultaneously using the semi-infinite linear programming (SILP) method, which is shown to be the most computationally efficient wrapper MKL solver.

Our experimental results on two multi-label data sets, ESP Game and MIR Flickr25000 demonstrated

the superiority of the proposed MK-MLR algorithm. MK-MLR efficiently combines heterogeneous data sources and exploit label correlations to maximize image categorization performance. In addition to yielding strong prediction performance, MK-MLR is also faster than OvA MKL formulations, which require solving MKL for each class. The sparsity of kernel combination weights and dual variables also leads to a much faster prediction step. However, there is still room for improvement of the prediction speed. One of the drawbacks of MK-MLR is that the computational complexity of the prediction step is linear in the number of classes. A future direction would be employing label set projection methods, such as compressed sensing, to make the prediction complexity sublinear in the number of classes.

Chapter 7

Contributions and Future Work

The main contributions of this thesis are efficient multiple kernel learning (MKL) and multi-label ranking algorithms that advance the state of the art in kernel learning for image categorization by combining different image representations and exploiting image label correlations for improved multi-label predictions.

7.1 Contributions

In Chapter 3 we proposed a stochastic approximation based multi-label multiple kernel learning algorithm that makes the following contributions:

- Developed a multi-label multiple kernel learning method that enables information sharing between class labels to improve the performance on the classes with a small number of training samples.
- Demonstrated that learning a shared combination of kernels for all classes improves the computational efficiency significantly without adversely affecting the classification performance.
- Proposed an stochastic optimization algorithm with a computational cost that is sublinear in the number of classes, $O(m^{1/3}\sqrt{lnm})$, making it suitable for handling a large number of classes, m.

The multi-label ranking method in Chapters 4 offers the following contributions:

- Formulated multi-label learning as a multi-label ranking task, which is more flexible than classification based on binary decisions because of the ability to provide an ordered list of image labels.

- Developed an approximation that reduces the number of constraints in the optimization problem and makes it linear in terms of the number of classes as compared to the quadratic dependency in the original ranking formulation. The approximation also enables class correlations to be implicitly included into the optimization process for an improved multi-label learning performance.

- Proposed an efficient optimization problem that is based on block coordinate descent and a simple line search algorithm for which the search boundaries are provided. Experimental results demonstrate that the computational load of the multi-label ranking algorithm is in the same order as one-vs-all SVM.

- Showed superior performance as compared to state-of-the-art multi-label learning methods on a data set in which full label information is available.

Studies on multi-label learning with incomplete class assignments in Chapter 5 offer the following contributions:

- Formally defined the problem of learning from multi-label data with incomplete class assignments.

- Developed a multi-label ranking method (MLR-GL), which explicitly addresses the challenge of learning from incompletely labeled data by exploiting the group lasso technique to combine the ranking errors.

- Proposed a computationally efficient optimization algorithm that has a closed-form solution. Experimental results demonstrate that the complexity of the multi-label ranking algorithm is in the same order as one-vs-all SVM.

- Empirically demonstrated the robustness of MLR-GL for incomplete class assignment problem.

We proposed a multiple kernel multi-label ranking method (MK-MLR) in Chapter 6, which is an extension of the MLR-L_1 algorithm in Chapter 4 to the multiple kernel setting and makes the following contributions:

- Proposed a method (MK-MLR) that combines multiple kernel learning and multi-label ranking in a single framework.
- Developed an efficient semi-infinite linear programming (SILP) algorithm that learns a single kernel combination for all classes.
- Showed empirically that the MK-MLR algorithm finds the optimal shared sparse kernel combinations of the base kernels for all classes. Sparse solutions improve the computational efficiency and robustness by eliminating weak or noisy kernels/features.
- Sparseness is particularly important for the prediction step, in which feature extraction is the main bottleneck. The experimental results showed that sparsity of the kernel combination coefficient vector reduces the prediction time. Because of its sparse solutions, MK-MLR algorithm reduces the prediction time significantly (order of seconds) compared to other methods which fail to yield sparse solutions.

Based on the extensive empirical evaluations made in this dissertation, we make the following recommendations:

- Despite the high computational cost in the training step, multiple kernel learning is useful for image categorization. It not only optimizes the classification performance by choosing the best kernel combination, but the sparse MKL also decreases the prediction time significantly by minimizing the time spent for feature extraction.
- MKL is particularly useful when the number of kernels/features is high and there are potentially weak/noisy kernels, which necessitates kernel selection for an improved classification performance. In the settings where there is a small number of strong base kernels, using the average of the base kernels would give comparable results to MKL.
- Learning a shared kernel combination for all classes is a good strategy to follow in multiple kernel learning for image categorization. Although the assumption of all classes sharing the same kernel might not work for other application domains, it not only yields good classification performance, but also reduces the training and prediction times significantly.

- Casting multi-label learning as a ranking problem is an effective way to boost the classification performance, particularly when the number of classes is high. The multi-label ranking methods presented in this dissertation are able to exploit the label correlations without making strong assumptions on the data, proving their effectiveness in classification and in their generalizability.

7.2 Future Work

Despite significant progress in the literature and this dissertation, there are some shortcomings of the current multiple kernel and multi-label learning methods for image categorization. We point out the following research directions:

- Improving the scalability of multiple kernel learning methods: Although MKL methods have been shown to be very useful in learning an optimal combination of different image representations and corresponding kernel functions, they do not scale well to training sets with millions of images and thousands of classes. In Chapter 3, we addressed the problem of a large number of classes. However, handling a large number of training samples is still the biggest challenge in using MKL. One of the priorities for MKL research should be making MKL methods scalable to data containing millions of samples.

- Computational efficiency in the prediction phase: In general, computational efficiency in the prediction step is more important than the training efficiency for practical systems, since the training phase can be done off-line. On the other hand, a server, for instance, might need to make a decision in a short time, making a fast prediction algorithm necessary. Therefore, it is important to develop efficient multiple kernel multi-label prediction algorithms. However, there are only a few studies in the machine learning literature that target improving the prediction speed.

APPENDIX

Appendix A

Supplementary materials

In this chapter, we first discuss the image categorization problem by briefly explaining the image representations, data sets, and evaluation measures we use in our experiments. Then, we provide the proofs of some theorems that were not included in the corresponding chapters.

A.1 Image Representation

We start with a brief background on image representations, and then briefly explain the bag-of-words (BoW) model, which is the most widely used low-level image representation technique. We also discuss the use of high level (semantic) image representations for image categorization.

A.1.1 A Brief History

The history of the published work on image categorization can be traced back to the 1960s [174]. The majority of the studies in the 1960s aimed to model and recognize simple geometric objects in an image. Such techniques are called "model-based recognition methods" [175, 176]. The goal in model-based recognition is to define or describe models for object categories and find matches between models and the detected objects in an image.

In the 1990s, we saw a rapid growth in the object recognition literature, probably due to the improve-

ments in imaging and processing technologies. Although there were still methods using local shape-based features, i.e., modeling via small shape parts [177] and polygon approximation of object boundaries [178], researchers started to use color [179, 180] and texture based representations [181, 182] as well. The early works on automatic image annotation, which can be considered as a subset of the image categorization problem, used image segmentation to extract blobs/regions from the image. Once the features are extracted for each of these regions, the corresponding image labels would be extracted for these regions [183–186]. However, this approach requires a successful segmentation step, which is a very difficult task.

Interest in extracting key points from an image and describing the local patches around these key points evolved in the 1990s [187,188]. The popularity of local features/descriptors has increased even more rapidly with the success of the SIFT algorithm, the seminal work by Lowe [189]. The SIFT approach for local descriptor extraction enabled high accuracy for the image matching problem. The bag-of-words (BoW) model enabled using key-point descriptors beyond the simple image matching problem by efficiently constructing a global representation for an entire image, which is necessary for image categorization, based on local key-point descriptors like SIFT features [190]. Among various approaches developed for image representation, the bag-of-words (BoW) model is the most popular due to its simplicity and success in practice. Most state-of-the-art methods use the bag-of-words model. Therefore, we also use the BoW model in our experiments.

A.1.2 The Bag-of-Words (BoW) Model

The first step in the BoW model is to detect key points or key-regions from images. Many algorithms have been developed for key-point/region detection [104,189,191], each having its own strengths and weaknesses. For instance, although dense sampling is shown to be superior to other techniques for image categorization, it usually yields a large number of key points and might lead to high computational costs. To have a richer variety of representations, in our experiments we used Harris-Laplacian [104] and Canny-edge detector based key-point methods in addition to dense sampling.

The second step is to generate local descriptors for the detected key points/regions. There is a rich literature on local descriptors, among which scale invariant feature transform (SIFT) [189] is, without doubt,

Table A.1: A list of techniques that can be used in each module of the Bag-of-Words (BOW) model

Region Detector	Dense sampling, random sampling, Harris points, Harris-Laplace regions, Hessian-Laplace, Harris-Affine regions, Hessian-Affine regions
Descriptor	SIFT, GLOH, Shape context, PCS-SIFT, spin images, steerable filters, LBP, cross-correlation, color histograms, HOG
Visual Dictionary	k-means, hierarchical k-means, GMM
Encoding/quantization	Vector quantization, Salient coding, LLC, LCC, Fisher vector, Sparse coding
Pooling technique	max-pooling, average pooling
Spatial arrangement	1×1, 2×2, 4×4, 1×3, 3×1
Kernel function	Linear, RBF, polynomial, χ^2

the most popular. Other techniques that we use in our experiments to improve the recognition performance are local binary patterns (LBP) [95] and histogram of oriented gradients (HOG) [192].

Given the descriptors, the third step of the BoW model is to construct a visual vocabulary. Both the dictionary size and the technique used to create the dictionary can have a significant impact on the final recognition accuracy. In our experiments, we use k-means clustering technique to generate the dictionary. Given the dictionary, the next step is to map each key-point to a visual word in the dictionary, a step that is often referred to as the encoding module. Recent studies express a vast amount of interest in the encoding step, resulting in many alternatives to vector quantization (e.g., Fisher kernel representation [193]).

The last step in the BoW model is the pooling step that pools encoded local descriptors into a global histogram representation. Various pooling strategies have been proposed for the BoW model such as mean and max-pooling, two techniques that we employ in our experiments. Studies [103] have shown that it is important to take into account the spatial layout of key points in the pooling step. One common approach is to divide an image into multiple regions and construct a histogram for each region separately. A well known example of this approach is spatial pyramid pooling [103] that divides an image into 1×1, 2×2, and 4×4 grids.

Table A.1 lists different techniques for each module of the BoW model. Besides the BoW model, many alternative low-level image features have been proposed for object recognition, including GIST [102], color

Table A.2: Data set statistics

	# samples	# classes	avg. no. of labels/img	avg no. of img/label
Caltech 101	8,677	101	1	85.9
ImageNet subset	81,738	101	1	85.9
VOC 2007	9,963	20	1.5	729.9
MIR Flickr subset	10,199	457	2.7	145.4
ESP Game subset	100,000	500	8.5	1691.3

histograms, V1S+ [97], and geometric blur [99].

A.1.3 High-level Image Representations

Although most of the image categorization is based on low-level features, particularly the BoW model, the use of high-level features is growing. One of the popular high-level image representations tools is the object banks method [173]. Li et al. defined a total of 177 different pre-computed object detectors using large object recognition data sets like ImageNet and LabelMe [194]. Each object detector is based on multi-scale, spatial pyramid representation and linear classifiers. Then, an image can be represented as a set of responses to these object detectors (classifiers). The object bank method is closely related to the image attributes method [195]. Attributes are human-designed names, such as {*"striped"*, *"has a tail"*} and by using a separate classifier for each attribute, an image can be described based on the attributes it has. In our multiple kernel learning experiments, we employ object bank representations in addition to several low level features to increase the number of base kernels and richness of the representations.

A.1.4 Data Sets

The majority of the data sets we use are multi-labeled data sets. However, in order to compare different multiple kernel learning solvers, we also use multi-class single-label benchmarks. Table A.2 provides statistics of the data sets we used in our experiments.

A.1.4.0.1 The Caltech 101 data set has been used in many MKL studies; therefore, we also use it in our MKL experiments. It is comprised of 9,146 images from 101 object classes and an additional class of

cougar

strawberry

snoopy

crocodile

Figure A.1: Four example images from the Caltech 101 data set with their labels.

"background" images. Caltech 101 is a multi-class single-label data set in which each image is assigned to one object class. As it can be seen from the sample images in Figure A.1, the objects are generally center aligned, scaled, and are not occluded. Because of these reasons, Caltech 101 is considered as a relatively easy data set for classification.

A.1.4.0.2 The Pascal VOC 2007 data set is comprised of 9,963 images from 20 object classes. Unlike Caltech 101, more than half of the images in VOC 2007 are assigned to multiple classes. Overall, it is a more challenging data set than Caltech 101 because of the large variations in object size, orientation, and shape, as well as the occlusion problem.

A.1.4.0.3 A subset of ImageNet data set is used in [106] for evaluating multiple kernel learning methods for image categorization. While the ImageNet data set contains 14,197,122 images from 21,841 categories, the data set that is used in the ImageNet Large Scale Visual Recognition Challenges contain 1.2 million training images from $1,000$ categories [196]. However, following the protocol in [106], we use $81,738$ images from ImageNet that belong to 18 out of 20 categories specified in VOC 2007; only 18 of the VOC 2007 categories are available within the ImageNet data set. This is significantly larger than Caltech

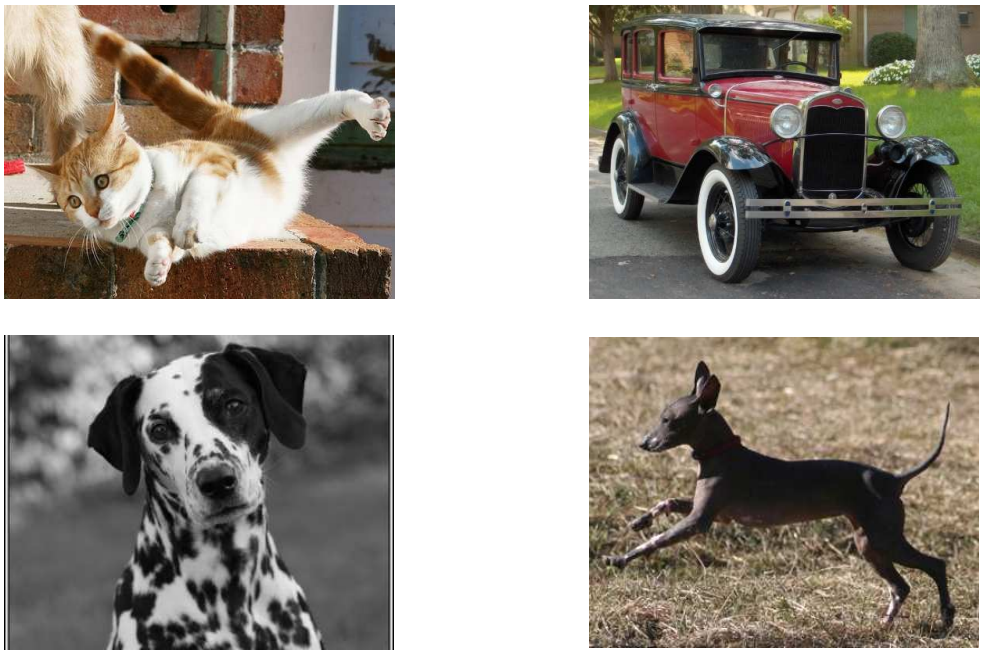

Figure A.2: Four example images from the ImageNet data set. A *cat* and a *car* image are shown in the top row. The second row has two dog images, one from the *dalmatian* synset, and one from the *Mexican hairless* synset

101 and VOC 2007, making it possible to examine the scalability of MKL methods for image categorization. Like Caltech 101, ImageNet is also a multi-class single-label data set, and we use this data set exclusively for the MKL experiments. Although the objects in the images are not always well-aligned and scaled, this data set is not considered challenging for classification because objects are roughly aligned, and there is only mild object occlusion, as seen in Figure A.2. Therefore, we can still consider the subset of ImageNet that we are using as a relatively easy data set. Note that, although the ImageNet data set has a hierarchical label structure, we will not be considering this structure in our experiments. For instance, we label the two images in the bottom row of Figure A.2 as two instances of dog images, although their synsets, which are *dalmatian* and *mexican hairless*, are different.

A.1.4.0.4 MIR Flickr25000 is a subset of the MIR Flickr-1M data set [154] that is used for classification challenges. It was created to be used for the visual concept detection and annotation tasks in the IMAGECLEF Challenge [197]. The data set contains 25,000 images with 457 types of tags. MIR

Figure A.3: Two example images from the MIR Flickr data set. Left image (reflection effect) is by Szymczak [1] and the right image (fish eye effect) is by Wild. [2]

Flickr25000 can be considered as a more difficult data set for classification compared to VOC 2007 and Caltech 101 because it is multi-labeled and it has a larger number of classes. In addition to all the challenges we have listed for the VOC 2007 data set, the MIR Flickr25000 data set poses extra difficulties because of the camera effects used by the photographers who took the photos, such as tilt shifting, post-processing, cinematic effects, etc. Figure A.3 shows two images with such effects.

A.1.4.0.5 ESP Game is an online game that involves comparing the annotations of multiple users (competitors) for an image to retrieve the relevant labels [198]. The labels that are agreed on by multiple annotators are treated as true labels, and the annotators who provide these true labels acquire points for each correct annotation they provide. The ESP Game data set, which contains $100,000$ images with $26,449$ annotations, is also one of the more difficult data sets for multi-label learning. As it can be seen from Figure A.4, the types of images (e.g., cartoon, video games, portrait, etc.) show an immense variety, and images are not always of high quality (low resolution, occlusion). We pick 500 of the most frequent labels and use the images that contain at least one of these 500 labels. Although most of the labels describe concrete objects, there are also abstract image labels such as *fight*, *sale*, *view*, and *symbol*.

A.1.5 Evaluation Measures

We use two approaches to evaluate an algorithm for image categorization. Given an image, the first approach is to rank the labels and measure the ability of an algorithm to rank the relevant labels higher than

Figure A.4: Four example images from the ESP Game data set.

irrelevant ones. In the second approach, given a category (label), the goal is to measure the performance of an algorithm in separating positive-labeled images from negative-labeled ones. The first approach is image based evaluation, whereas the second one is category based evaluation.

A.1.5.1 Image Based Evaluation:

Since we focus on multi-label ranking, we rank the classes in the descending order of their scores for a given image. The true label assignments (provided by human annotators) of an image are called relevant labels and the remaining labels are called irrelevant labels. For each image, we predict its categories by retrieving the first k labels with the largest scores. We vary k, i.e., the number of retrieved labels, from 1 to the total number of categories, and compute the following scores for an image indexed with i:

- True Positive (TP_i): The number of correctly retrieved relevant labels
- False Positive (FP_i): The number of retrieved labels which are not relevant
- False Negative (FN_i): The number of relevant labels which are not retrieved
- True Negative (TN_i): The number of rejected irrelevant labels per image

- True Positive Rate: $TPR_i = \frac{TP_i}{TP_i+FN_i}$
- False Positive Rate: $FPR_i = \frac{FP_i}{FP_i+TN_i}$
- Recall $= \frac{TP_i}{TP_i+FN_i}$
- Precision $= \frac{TP_i}{TP_i+FP_i}$

Once the above scores are calculated for an image, we can obtain the AUC-ROC (area under the curve for Receiver operating characteristic graph) and AP (average precision) measures. ROC curve plots TPR (y-axis) against FPR (x-axis), and the area under the curve ($\%$), which is a value between 0 and 100, measures the ranking performance of an algorithm: higher scores are better. Following PASCAL Visual Object Classes (VOC) challenge, we calculate the precision values corresponding to a set of evenly spaced recall levels $\{$0, 0.1, ... ,1.0$\}$, and calculate the mean of these precision values to get the AP score. Once AUC-ROC and AP scores are calculated for each image, we take the mean of these scores over all test images (micro-averaging).

A.1.5.2 Category Based Evaluation:

We use category based evaluation for the multiple kernel learning experiments, which involves comparing binary MKL algorithms. Note that, unlike the previous case, we rank images for each label. Let us redefine the measures we use for the classification performance:

- Category-based True Positive (TP_c): The number of images that are correctly assigned a positive label for a category
- Category-based False Positive (TP_c): The number of images that are falsely assigned a positive label for a category
- Category-based False Negative (FN_c): The number of images that are falsely assigned a negative label for a category
- Category-based Recall: $= \frac{TP_c}{TP_c+FN_c}$

- Category-based Precision: $= \frac{TP_c}{TP_c+FP_c}$

By using the category based precision and recall values, we can calculate the average precision (AP) score for each category.

As suggested in the PASCAL Visual Object Classes challenge, we only use the MAP score. The reason for this is that the three data sets we use for the MKL experiments, namely Caltech 101, VOC 2007, and a subset of ImageNet, give fairly high classification performance in terms of AUC-ROC, making it difficult to distinguish the performance difference between the baselines. Therefore, we will be using only the MAP score for the MKL experiments.

A.1.6 State-of-the-art Performance in Image Categorization

The winner of the ILSVRC (ImageNet) 2012 and 2013 Challenges used deep convolutional networks on raw pixel data [199]. For example, the winner of ILSVRC (ImageNet) 2012 uses a trained neural network that has 60 million parameters and 650,000 neurons, consisting of five convolutional layers. The deep convolutional neural network algorithm yielded an error rate of 0.15 for rank-5 predictions, improving over the second best method in the competition by 10%. The performance was further improved in ILSVRC by combining several CNNs and an error rate of 11.74% was achieved. Deep convolutional networks produce very promising results both for classification and detection when the number of images is high (in the order of millions). In this dissertation, we are interested in developing classification algorithms that would work on any image representation. In contrast, convolutional neural networks learn their own features.

The method ranked second in the ILSVRC 2012 Challenge used a set of different BoW representations, including SIFT, LBP, and GIST based Fisher vector features. This approach, which produces an error rate of 0.26 for rank-5 predictions, learns a separate classifier for each feature, which are 262,144 dimensional vectors. It then calculates a weighted sum of these individual classifiers for the final predictions.

Similar to ILSVRC 2012 Challenge, we see that the top performing methods in the Pascal VOC categorization challenge combine different representations (mostly Fisher vector representation based) and build features that are over 300,000 dimensional. The winner group includes additional modules such as object detection/localization and subclass modeling. While we see that the winner method in the VOC 2012 chal-

lenge, which utilizes object detection, yields a MAP score of 82%, the reported result on the VOC 2007 data set using a single feature is 61.7% (only classification). It is important to note that we are interested in developing methods that perform only categorization, meaning that our algorithms only require a single global descriptor for each image and do not need localization (i.e., bounding boxes) information in the training process.

When very high dimensional feature vectors are used, linear SVMs yield results similar to kernel SVMs. Although linear SVMs are more efficient than kernel SVMs, the main bottleneck for them in the prediction step is feature extraction. On the other hand, our goal in this dissertation is to optimize the classification performance for features that are relatively low dimensional (1,000 to 10,000) by using kernel classifiers.

A.2 Proofs for Chapter 2

In this section we prove the equivalance between Eqs. (A.1) and (A.2) (originally Eqs. (2.2) and (2.7) in Chapter 2).

$$\min_{\boldsymbol{\beta}\in\Delta, f\in\mathcal{H}_\beta} \frac{1}{2}\,||f||^2_{\mathcal{H}_\beta} + C\sum_{i=1}^{n}\ell(y^i f(\mathbf{x^i})) \tag{A.1}$$

$$\min_{\lambda\in\mathbb{R}^s_+,\sum_j\lambda_j=1}\ \min_{\{f_j\in\mathcal{H}_j\}_{j=1}^s} \frac{1}{2}\sum_{j=1}^{s}\lambda_j||f_j||^2_{\mathcal{H}_j} + C\sum_{i=1}^{n}\ell\left(\sum_{j=1}^{s}y^i\lambda_j f_j(\mathbf{x^i})\right) \tag{A.2}$$

We first rewrite $C\ell(z)$ as $\max_{\alpha\in[0,C]}\alpha(1-z)$ and place it into Eq. (A.2) to get Eq. (A.3),

$$\min_{\lambda\in\mathbb{R}^s_+,\sum_j\lambda_j=1}\ \min_{\{f_j\in\mathcal{H}_j\}_{j=1}^s}\ \max_{\alpha\in[0,C]^n} \frac{1}{2}\sum_{j=1}^{s}\lambda_j||f_j||^2_{\mathcal{H}_j} + \sum_{i=1}^{n}\alpha^i\left(1-\sum_{j=1}^{s}y^i\lambda_j f_j(\mathbf{x^i})\right). \tag{A.3}$$

The problem in Eq. (A.3) becomes a convex-concave optimization problem and, according to von Newman's lemma, we can switch minimization with respect to f_j and maximization with respect to $\boldsymbol{\alpha}$. It is straightforward to show that $f_j(\mathbf{x}) = \sum_{i=1}^{n}\alpha_i y^i \kappa_j(\mathbf{x}, \mathbf{x^i})$ is the minimizer. Using this expression, the optimization problem can be rewritten as in Eq. (A.4), which is exactly the same as the dual form of Eq. (2.2).

$$\min_{\boldsymbol{\beta}\in\Delta}\max_{\alpha\in\mathcal{Q}}\widehat{\mathcal{L}}(\boldsymbol{\alpha},\boldsymbol{\beta}) = \mathbf{1}^\top\boldsymbol{\alpha} - \frac{1}{2}(\boldsymbol{\alpha}\circ\mathbf{y})^\top\mathbf{K}(\boldsymbol{\beta})(\boldsymbol{\alpha}\circ\mathbf{y}). \tag{A.4}$$

This is an evidence that Eqs. (A.2) and (A.1) are equivalent and concludes the proof.

A.3 Proofs for Chapter 3

Proposition 4. *Eq. (A.6) is the dual problem of Eq. (A.5).*

$$\min_{\boldsymbol{\beta}\in\Delta}\min_{\{f_k\in\mathcal{H}(\boldsymbol{\beta})\}_{k=1}^m}\left\{\sum_{k=1}^m H_k = \sum_{k=1}^m\left\{\frac{1}{2}|f_k|^2_{\mathcal{H}(\boldsymbol{\beta})} + \sum_{i=1}^n \ell\left(y_k^i f_k(\mathbf{x^i})\right)\right\}\right\}, \tag{A.5}$$

where $\ell(z) = \max(0, 1-z)$ *and* $\mathcal{H}(\boldsymbol{\beta})$ *is a Reproducing Kernel Hilbert Space endowed with kernel* $\kappa(\mathbf{x},\mathbf{x}';\boldsymbol{\beta}) = \sum_{j=1}^s \beta_j\kappa_j(\mathbf{x},\mathbf{x}')$.

$$\min_{\boldsymbol{\beta}\in\Delta}\max_{\boldsymbol{\alpha}\in\mathcal{Q}_1}\left\{\mathcal{L}(\boldsymbol{\beta},\boldsymbol{\alpha}) = \sum_{k=1}^m\left\{[\boldsymbol{\alpha}_k]^\top\mathbf{1} - \frac{1}{2}(\boldsymbol{\alpha}_k\circ\mathbf{y}_k)^\top\mathbf{K}(\boldsymbol{\beta})(\boldsymbol{\alpha}_k\circ\mathbf{y}_k)\right\}\right\}, \tag{A.6}$$

where $\mathcal{Q}_1 = \{\boldsymbol{\alpha} = (\boldsymbol{\alpha}_1,\ldots,\boldsymbol{\alpha}_m) : \boldsymbol{\alpha}_k \in [0,C]^n, k = 1,\ldots,m\}$.

Proof. We first rewrite $\ell(z)$ as

$$\ell(z) = \max_{x\in[0,1]}(x - xz),$$

Using the above expression for $\ell(z)$, the second term of H_k can be rewritten as,

$$\sum_{i=1}^n \max_{\alpha_k^i\in[0,C]}\left(\alpha_k^i - \alpha_k^i y_k^i f_k(\mathbf{x^i})\right),$$

According to von Newman's lemma, we can switch minimization (over f_k) with maximization (over $\boldsymbol{\alpha}$). By taking the minimization over f_k first, we have

$$f_k(x) = \sum_{i=1}^n y_k^i\alpha_k^i\kappa(\mathbf{x^i},\mathbf{x}).$$

Finally the problem becomes

$$\min_{\boldsymbol{\beta}\in\Delta}\max_{\boldsymbol{\alpha}\in[0,C]}\left\{\mathcal{L}(\boldsymbol{\beta},\boldsymbol{\alpha})=\sum_{k=1}^{m}\left\{[\boldsymbol{\alpha}_k]^\top\mathbf{1}-\frac{1}{2}(\boldsymbol{\alpha}_k\circ\mathbf{y}_k)^\top\mathbf{K}(\boldsymbol{\beta})(\boldsymbol{\alpha}_k\circ\mathbf{y}_k)\right\}\right\}.$$

□

Proposition 5. *Eq. (A.8) is the dual problem of Eq. (A.7).*

$$\min_{\boldsymbol{\beta}\in\Delta}\min_{\{f_k\in\mathcal{H}(\boldsymbol{\beta})\}_{k=1}^m}\max_{1\le k\le m}H_k, \tag{A.7}$$

$$\min_{\boldsymbol{\beta}\in\Delta}\max_{\boldsymbol{\rho}\in B}\left\{\mathcal{L}(\boldsymbol{\beta},\boldsymbol{\rho})=\left\{\sum_{k=1}^{m}\left\{[\boldsymbol{\rho}_\mathbf{k}]^\top\mathbf{1}-\frac{1}{2}(\boldsymbol{\rho}_\mathbf{k}\circ\mathbf{y}_\mathbf{k})^\top\mathbf{K}(\boldsymbol{\beta})(\boldsymbol{\rho}_\mathbf{k}\circ\mathbf{y}_\mathbf{k})\right\}^{\frac{1}{2}}\right\}^2\right\}. \tag{A.8}$$

where

$$B=\left\{(\boldsymbol{\rho}_1,\ldots,\boldsymbol{\rho}_\mathbf{m}):\boldsymbol{\rho}_\mathbf{k}\in\mathbb{R}_+^n,k=1,\ldots,m,\boldsymbol{\rho}_\mathbf{k}\in[0,C\lambda_k]^n\ \textit{s.t.}\ \sum_{k=1}^{m}\lambda_k=1\right\}.$$

Proof. We start by formulating Eq. (A.7) as,

$$\min_{\boldsymbol{\beta}\in\Delta}\min_{\{f_k\in\mathcal{H}(\boldsymbol{\beta})\}_{k=1}^m}\min t \tag{A.9}$$

$$\text{subject to } H_k\le t,\ k=1,\ldots,m, \tag{A.10}$$

with extra variable $t\in\mathbb{R}$. Introducing the multiplier λ_k for $H_k\le t$, and using Proposition 1, the Lagrangian is

$$\begin{aligned}&t+\sum_{k=1}^{m}\lambda_k\left\{[\boldsymbol{\alpha}^k]^\top\mathbf{1}-\frac{1}{2}(\boldsymbol{\alpha}_\mathbf{k}\circ\mathbf{y}_\mathbf{k})^\top\mathbf{K}(\boldsymbol{\beta})(\boldsymbol{\alpha}_\mathbf{k}\circ\mathbf{y}_\mathbf{k})-t\right\}\\ &=(1-\mathbf{1}^\mathbf{T}\boldsymbol{\lambda})t+\sum_{k=1}^{m}\lambda_k\left\{[\boldsymbol{\alpha}_\mathbf{k}]^\top\mathbf{1}-\frac{1}{2}(\boldsymbol{\alpha}_\mathbf{k}\circ\mathbf{y}_\mathbf{k})^\top\mathbf{K}(\mathbf{p})(\boldsymbol{\alpha}_\mathbf{k}\circ\mathbf{y}_\mathbf{k})\right\},\end{aligned} \tag{A.11}$$

where $\boldsymbol{\alpha} \in [0, C]^n$. So, the dual function is

$$g(\boldsymbol{\beta}, \boldsymbol{\rho}, \boldsymbol{\lambda}) = \begin{cases} \sum_{k=1}^{m} \left\{ [\boldsymbol{\rho}_\mathbf{k}]^\top \mathbf{1} - \frac{1}{2} (\boldsymbol{\rho}_\mathbf{k} \circ \mathbf{y}_\mathbf{k})^\top \frac{\mathbf{K}(\boldsymbol{\beta})}{\lambda_k} (\boldsymbol{\rho}_\mathbf{k} \circ \mathbf{y}_\mathbf{k}) - t \right\} & \mathbf{1}^\top \boldsymbol{\lambda} = 1 \\ -\infty & \text{otherwise} \end{cases},$$

where $\boldsymbol{\rho}_\mathbf{k} = \boldsymbol{\alpha}_\mathbf{k} \lambda_k$. Then the dual problem is

$$\min_{\boldsymbol{\beta} \in \Delta} \max_{\boldsymbol{\rho} \in B} \max_{\boldsymbol{\lambda} \in \Lambda} \left\{ \mathcal{L}(\boldsymbol{\beta}, \boldsymbol{\rho}, \boldsymbol{\lambda}) = \sum_{k=1}^{m} \left\{ [\boldsymbol{\rho}_\mathbf{k}]^\top \mathbf{1} - \frac{1}{2} (\boldsymbol{\rho}_\mathbf{k} \circ \mathbf{y}_\mathbf{k})^\top \frac{\mathbf{K}(\boldsymbol{\beta})}{\lambda_k} (\boldsymbol{\rho}_\mathbf{k} \circ \mathbf{y}_\mathbf{k}) \right\} \right\},$$

where

$$B = \left\{ (\boldsymbol{\rho}_\mathbf{1}, \ldots, \boldsymbol{\rho}_\mathbf{m}) : \boldsymbol{\rho}_\mathbf{k} \in \mathbb{R}_+^n, k = 1, \ldots, m, \boldsymbol{\rho}_\mathbf{k} \in [0, C\lambda_k]^n \right\}.$$

Let $\min_{\boldsymbol{\beta} \in \Delta_1} \max_{\boldsymbol{\rho} \in B} (\boldsymbol{\rho}_\mathbf{k} \circ \mathbf{y}_\mathbf{k})^\top \mathbf{K}(\boldsymbol{\beta}) (\boldsymbol{\rho}_\mathbf{k} \circ \mathbf{y}_\mathbf{k}) = \psi_k$. To eliminate $\boldsymbol{\lambda}$, we rewrite the dual problem as maximization over $\boldsymbol{\lambda}$ for optimal ψ_k. Then, the Lagrangian becomes

$$\max_{\boldsymbol{\lambda} \in \Lambda} -\frac{1}{2} \sum_{k=1}^{m} \frac{\psi_k}{\lambda_k} + \upsilon \left(\sum_{k=1}^{m} \lambda_k - 1 \right).$$

Maximizing over $\boldsymbol{\lambda}$, we get

$$\upsilon = \frac{1}{2} \left\{ \sum_{k=1}^{m} \sqrt{\psi}_k \right\}^2$$

$$\lambda_k = \frac{\sqrt{\psi_k}}{\sum_{j=1}^{m} \sqrt{\psi_j}}$$

By eliminating $\boldsymbol{\lambda}$, we obtain the following dual of (A.7):

$$\min_{\boldsymbol{\beta} \in \Delta_1} \max_{\boldsymbol{\rho} \in B} \left\{ \mathcal{L}(\boldsymbol{\beta}, \boldsymbol{\rho}) = \left\{ \sum_{k=1}^{m} \left\{ [\boldsymbol{\rho}_\mathbf{k}]^\top \mathbf{1} - \frac{1}{2} (\boldsymbol{\rho}_\mathbf{k} \circ \mathbf{y}_\mathbf{k})^\top \mathbf{K}(\boldsymbol{\beta}) (\boldsymbol{\rho}_\mathbf{k} \circ \mathbf{y}_\mathbf{k}) \right\}^{\frac{1}{2}} \right\}^2 \right\}.$$

□

Proposition 6. *We define potential functions* $\Phi_\beta = \frac{\eta_\beta}{\eta_\gamma} \sum_{j=1}^{s} \beta_j \ln \beta_j$ *for* $\boldsymbol{\beta}$ *and* $\Phi_\gamma = \sum_{i=1}^{m} \gamma^i \ln \gamma^i$ *for* $\boldsymbol{\gamma}$*,*

and have the following equations for updating $\boldsymbol{\beta}^t$ and $\boldsymbol{\gamma}^t$ as

$$\beta_j^{t+1} = \frac{\beta_j^t}{Z_\beta^t} \exp(-\eta_\beta \nabla_{\beta_j} \mathcal{L}(\boldsymbol{\beta}^t, \boldsymbol{\gamma}^t)),\ \gamma_k^{t+1} = \frac{\gamma_k^t}{Z_\gamma^t} \exp(-\eta_\gamma \nabla_{\gamma_k} \mathcal{L}(\boldsymbol{\beta}^t, \boldsymbol{\gamma}^t)), \tag{A.12}$$

where Z_β^t and Z_γ^t are normalization factors that ensure ${\boldsymbol{\beta}^t}^\top \mathbf{1} = {\boldsymbol{\gamma}^t}^\top \mathbf{1} = 1$.

Proof. We denote by $D_{\Phi_{\boldsymbol{\beta}}}(\boldsymbol{\beta}, \boldsymbol{\beta}') : \Delta \times \Delta \mapsto \mathbb{R}_+$ and $D_{\Phi_\gamma}(\boldsymbol{\gamma}, \boldsymbol{\gamma}') : \Gamma \times \Gamma \mapsto \mathbb{R}_+$ the Bregman distance functions for $\boldsymbol{\beta}$ and $\boldsymbol{\gamma}$ that are induced by Φ_β and Φ_γ, respectively. Note that the Bregman distance between $\mathbf{z}$ and $\mathbf{z}'$ induced by the strictly convex function Φ, denoted by $D_\Phi(z, z')$, is defined as

$$D_\Phi(\mathbf{z}, \mathbf{z}') = \Phi(\mathbf{z}) - \Phi(\mathbf{z}') - \nabla\Phi(\mathbf{z}')^\top (\mathbf{z} - \mathbf{z}')$$

Using the Bregman distance function, we introduce two projection operators: $A_{\boldsymbol{\beta}}(\mathbf{g}_\beta; \Delta)$ that projects solution $\boldsymbol{\beta}$ into domain Δ along the direction $\mathbf{g}_\beta \in \mathbb{R}^s$ and $B_\gamma(\mathbf{g}_\gamma; \Gamma)$ that projects solution $\boldsymbol{\gamma}$ into domain Γ along the direction $\mathbf{g}_\gamma \in \mathbb{R}^m$. These two operators are defined as follows:

$$A_\beta(\mathbf{g}_{\boldsymbol{\beta}}) = \min_{\boldsymbol{\beta}' \in \Delta} \mathbf{g}_\beta^\top \boldsymbol{\beta}' + D_{\Phi_\beta}(\boldsymbol{\beta}', \boldsymbol{\beta}),\ B_{\boldsymbol{\gamma}}(\mathbf{g}_\gamma) = \min_{\gamma' \in \Gamma} \mathbf{g}_\gamma^\top \boldsymbol{\gamma}' + D_{\Phi_\gamma}(\boldsymbol{\gamma}', \boldsymbol{\gamma})$$

Based on the mirror prox method, we can solve the optimization problem in Eq. (3.3) iteratively. Given the solution $\boldsymbol{\beta}^t$ and $\boldsymbol{\gamma}^t$ of the current iteration, the new solution, denoted by $\boldsymbol{\beta}^{t+1}$ and $\boldsymbol{\gamma}^{t+1}$, is computed as

$$\boldsymbol{\beta}^{t+1} = A_{\boldsymbol{\beta}^t}\left(\eta_\beta \nabla_\beta \mathcal{L}(\boldsymbol{\beta}^t, \boldsymbol{\gamma}^t, \boldsymbol{\alpha}^t)\right), \quad \boldsymbol{\gamma}^{t+1} = C_{\gamma^t}\left(-\eta_\gamma \nabla_\gamma \mathcal{L}(\boldsymbol{\beta}^t, \boldsymbol{\gamma}^t, \boldsymbol{\alpha}^t)\right), \tag{A.13}$$

where $\eta_\beta > 0$ and $\eta_\gamma > 0$ are the step sizes. The two gradients are computed as

$$g_j(\boldsymbol{\beta}) = \frac{\partial \mathcal{L}(\boldsymbol{\beta}, \boldsymbol{\gamma}, \boldsymbol{\alpha})}{\partial \beta_j} = -\frac{1}{2} \sum_{k=1}^{m} \gamma_k (\boldsymbol{\alpha}_\mathbf{k} \circ \mathbf{y}_\mathbf{k})^\top \mathbf{K_j} (\boldsymbol{\alpha}_\mathbf{k} \circ \mathbf{y}_\mathbf{k}), j = 1, \ldots, s \tag{A.14}$$

$$g_k(\boldsymbol{\gamma}) = \frac{\partial \mathcal{L}(\boldsymbol{\beta}, \boldsymbol{\gamma}, \boldsymbol{\alpha})}{\partial \gamma_k} = [\boldsymbol{\alpha}_\mathbf{k}]^\top \mathbf{1} - \frac{1}{2} (\boldsymbol{\alpha}_\mathbf{k} \circ \mathbf{y}_\mathbf{k})^\top \mathbf{K}(\boldsymbol{\beta}) (\boldsymbol{\alpha}_\mathbf{k} \circ \mathbf{y}_\mathbf{k}), k = 1, \ldots, m \tag{A.15}$$

$$\tag{A.16}$$

By choosing the potential functions as

$$\Phi_{\boldsymbol{\beta}} = \frac{\eta_\beta}{\eta_\gamma} \sum_{j=1}^{s} \beta_j \ln \beta_j, \quad \Phi_\gamma = \sum_{k=1}^{m} \gamma_k \ln \gamma_k, \tag{A.17}$$

we have the following updating rules for $\boldsymbol{\beta}^{t+1} = (\beta_1^{t+1}, \ldots, \beta_s^{t+1})$ and $\boldsymbol{\gamma}^{t+1} = (\gamma_1^{t+1}, \ldots, \gamma_m^{t+1})$

$$\beta_j^{t+1} = \frac{\beta_j^t}{Z_\beta^t} \exp\left(-\eta_\gamma g_j(\boldsymbol{\beta}^t)\right), j = 1, \ldots, s \tag{A.18}$$

$$\gamma_k^{t+1} = \frac{\gamma_t^k}{Z_\gamma^t} \exp\left(\eta_\gamma g_k(\gamma_t)\right), k = 1, \ldots, m \tag{A.19}$$

where Z_β^t and Z_γ^t are defined as

$$Z_\beta^t = \sum_{j=1}^{s} \beta_j^t \exp\left(-\eta_\gamma g_j(\boldsymbol{\beta}^t)\right) \quad Z_\gamma^t = \sum_{k=1}^{m} \gamma_k^t \exp\left(\eta_\gamma g_k(\gamma^t)\right)$$

□

Theorem 10. *After running Algorithm 3 over T iterations, we have the following inequality for the solution $\widehat{\mathbf{p}}$ and $\widehat{\boldsymbol{\gamma}}$ obtained by Algorithm 3*

$$\mathrm{E}\left[\Delta\left(\hat{\boldsymbol{\beta}}, \hat{\boldsymbol{\gamma}}\right)\right] \leq \frac{1}{\eta_\gamma T}\left(\ln m + \ln s\right) + \eta_\gamma \left(d\frac{m^2}{2\delta^2}\lambda_0^2 n^2 C^4 + n^2 C^2\right),$$

where d is a constant term and $\mathrm{E}[\cdot]$ stands for the expectation over the sampled task indices of all iterations.

Proof. Define

$$\widehat{\mathbf{g}}^\beta(\boldsymbol{\beta}^t, \boldsymbol{\gamma}^t) = (\widehat{g}_1^\beta(\boldsymbol{\beta}^t, \boldsymbol{\gamma}^t), \ldots, \widehat{g}_s^\beta(\boldsymbol{\beta}^t, \boldsymbol{\gamma}^t)),\ \widehat{\mathbf{g}}^\gamma(\boldsymbol{\beta}^t, \boldsymbol{\gamma}^t) = (\widehat{g}_1^\gamma(\boldsymbol{\beta}^t, \boldsymbol{\gamma}^t), \ldots, \widehat{g}_m^\gamma(\boldsymbol{\beta}^t, \boldsymbol{\gamma}^t)).$$

Using the result of variation inequality [119], we have the following inequality for any $\boldsymbol{\beta} \in \Delta$ and

$\boldsymbol{\gamma} \in \Gamma$

$$\Delta\left(\boldsymbol{\beta}^t, \boldsymbol{\gamma}^t\right) \leq (\boldsymbol{\beta}^t - \boldsymbol{\beta})^\top \nabla_\beta \mathcal{L}(\boldsymbol{\beta}^t, \boldsymbol{\gamma}^t) - (\boldsymbol{\gamma}^t - \boldsymbol{\gamma})^\top \nabla_\gamma \mathcal{L}(\boldsymbol{\beta}^t, \boldsymbol{\gamma}^t). \tag{A.20}$$

According to Proposition 1, we have

$$\mathrm{E}_t\left[\widehat{\mathbf{g}}^\beta(\boldsymbol{\beta}^t, \boldsymbol{\gamma}^t)\right] = \nabla_\beta \mathcal{L}(\boldsymbol{\beta}^t, \boldsymbol{\gamma}^t),\ \mathrm{E}_t\left[\widehat{\mathbf{g}}^\gamma(\boldsymbol{\beta}^t, \boldsymbol{\gamma}^t)\right] = \nabla_\gamma \mathcal{L}(\boldsymbol{\beta}^t, \boldsymbol{\gamma}^t).$$

We therefore can rewrite Eq. (A.20) as

$$\mathrm{E}_t\left[\Delta\left(\boldsymbol{\beta}^t, \boldsymbol{\gamma}^t\right)\right] \leq \mathrm{E}_t\left[(\boldsymbol{\beta}^t - \boldsymbol{\beta})^\top \widehat{\mathbf{g}}^\beta(\boldsymbol{\beta}^t, \boldsymbol{\gamma}^t) - (\boldsymbol{\gamma}^t - \boldsymbol{\gamma})^\top \widehat{\mathbf{g}}^\gamma(\boldsymbol{\beta}^t, \boldsymbol{\gamma}^t)\right].$$

From [200] (chapter 11), we know that

$$\eta_\gamma(\boldsymbol{\beta}^t - \boldsymbol{\beta})^\top \widehat{\mathbf{g}}^\beta(\boldsymbol{\beta}^t, \boldsymbol{\gamma}^t) \leq \mathrm{KL}(\boldsymbol{\beta}\|\boldsymbol{\beta}^t) - \mathrm{KL}(\boldsymbol{\beta}\|\boldsymbol{\beta}^{t+1}) + \mathrm{KL}(\boldsymbol{\beta}^t\|\boldsymbol{\beta}^{t+1}),$$

and

$$-\eta_\gamma(\boldsymbol{\gamma}^t - \boldsymbol{\gamma})^\top \widehat{\mathbf{g}}^\gamma(\boldsymbol{\beta}^t, \boldsymbol{\gamma}^t) \leq \mathrm{KL}(\boldsymbol{\gamma}\|\boldsymbol{\gamma}^t) - \mathrm{KL}(\boldsymbol{\gamma}\|\boldsymbol{\gamma}^{t+1}) + \mathrm{KL}(\boldsymbol{\gamma}^t\|\boldsymbol{\gamma}^{t+1}).$$

Therefore, we have

$$\eta_\gamma \sum_{t=1}^{T} \Delta\left(\boldsymbol{\beta}^t, \boldsymbol{\gamma}^t\right) \leq \mathrm{KL}(\boldsymbol{\beta}\|\boldsymbol{\beta}^1) + \mathrm{KL}(\boldsymbol{\gamma}\|\boldsymbol{\gamma}^1) + \sum_{t=1}^{T} \left\{\mathrm{KL}(\boldsymbol{\beta}^t\|\boldsymbol{\beta}^{t+1}) + \mathrm{KL}(\boldsymbol{\gamma}^t\|\boldsymbol{\gamma}^{t+1})\right\}.$$

We are going to bound each of the three terms on the right hand side of the inequality. First, it is obvious that $\mathrm{KL}(\boldsymbol{\beta}\|\boldsymbol{\beta}^1) \leq \ln s$ and $\mathrm{KL}(\boldsymbol{\gamma}\|\boldsymbol{\gamma}^1) \leq \ln m$ given both $\boldsymbol{\gamma}^1$ and $\boldsymbol{\beta}^1$ are uniform distributions. Second, we bound $\mathrm{KL}(\boldsymbol{\beta}^t\|\boldsymbol{\beta}^{t+1})$ as follows

$$
\begin{aligned}
\mathrm{KL}(\boldsymbol{\beta}^t|\boldsymbol{\beta}^{t+1}) &= \frac{\eta_\beta}{\eta_\gamma}\left\{\sum_{j=1}^{s}\beta_j^t \ln\left(\frac{\beta_j^t}{\beta_j^{t+1}}\right)\right\} = \frac{\eta_\beta}{\eta_\gamma}\left\{\sum_{j=1}^{s}\beta_j^t \ln\left(Z_\beta^t \exp\{\eta_\gamma \widehat{g}_j^\beta\}\right)\right\} \\
&= \frac{\eta_\beta}{\eta_\gamma}\left\{\sum_{j=1}^{s}\beta_j^t \eta_\gamma \widehat{g}_j^\beta(\boldsymbol{\beta}^t,\boldsymbol{\gamma}^t) + \sum_{j=1}^{s}\beta_j^t \ln(Z_p^t)\right\} \\
&= \frac{\eta_\beta}{\eta_\gamma}\left\{\sum_{j=1}^{s}\beta_j^t \eta_\gamma \widehat{g}_j^\beta(\boldsymbol{\beta}^t,\boldsymbol{\gamma}^t) + \sum_{j=1}^{s}\beta_j^t \ln\left(\sum_{j=1}^{s}\beta_j^t \exp\left[-\eta_\gamma \widehat{g}_j^\beta(\boldsymbol{\beta}^t,\boldsymbol{\gamma}^t)\right]\right)\right\} \\
&= \frac{\eta_\beta}{\eta_\gamma}\left\{-\left(-\eta_\gamma E\left[\widehat{g}_j^\beta\right]\right) + \ln\left(E\left[\exp\left(-\eta_\gamma \widehat{g}_j^\beta(\boldsymbol{\beta}^t,\boldsymbol{\gamma}^t)\right)\right]\right)\right\} \\
&\leq \frac{\eta_\beta}{\eta_\gamma}\left\{\frac{\eta_\gamma^2}{2}\max_{1\leq j\leq s}[\widehat{g}_j^\beta(\boldsymbol{\beta}^t,\boldsymbol{\gamma}^t)]^2\right\} = \frac{c\eta_\gamma^2}{2}|\widehat{g}^\beta(\boldsymbol{\beta}^t,\boldsymbol{\gamma}^t)|_\infty^2,
\end{aligned}
$$

where the inequality follows directly from the Hoeffiding inequality, and c is a constant such that $\eta_p = c\eta_\gamma$. Similarly, we have $\mathrm{KL}(\boldsymbol{\gamma}^t\|\boldsymbol{\gamma}^{t+1}) \leq \frac{\eta_\gamma^2}{2}|\widehat{\mathbf{g}}^\gamma(\boldsymbol{\beta}^t,\boldsymbol{\gamma}^t)|_\infty^2$.

By combining the above results together, we have

$$
\eta_\gamma \mathrm{E}\left[\sum_{t=1}^{T}\Delta\left(\boldsymbol{\beta}^t,\boldsymbol{\gamma}^t\right)\right] \leq \ln m + \ln s + \eta_\gamma^2 \sum_{t=1}^{T}\mathrm{E}\left[c|\widehat{\mathbf{g}}^\beta(\boldsymbol{\beta}^t,\boldsymbol{\gamma}^t)|_\infty^2 + |\widehat{\mathbf{g}}^\gamma(\boldsymbol{\beta}^t,\boldsymbol{\gamma}^t)|_\infty^2\right]
$$

Using Eq. (A.14), we can bound $|\widehat{\mathbf{g}}^\beta(\boldsymbol{\beta}^t,\boldsymbol{\gamma}^t)|_\infty$ as follows

$$
\begin{aligned}
|\widehat{\mathbf{g}}^\beta(\boldsymbol{\beta}^t,\boldsymbol{\gamma}^t)|_\infty &= \max_{1\leq j\leq s}\widehat{g}_j^\beta(\boldsymbol{\beta}^t,\boldsymbol{\gamma}^t) \\
&= \max_{1\leq j\leq s}\left|-\frac{1}{2}(\boldsymbol{\alpha}^{a_t}\circ\mathbf{y}^{a_t})^\top \mathbf{K}^a(\boldsymbol{\alpha}^{a_t}\circ\mathbf{y}^{a_t})\right| \\
&\leq \frac{1}{2}(C\mathbf{1})^\top \mathbf{V}\mathbf{D}\mathbf{V}^{-1}(C\mathbf{1})] \leq \frac{\lambda_0}{2}(C\mathbf{1})^\top \mathbf{V}\mathbf{I}\mathbf{V}^{-1}(C\mathbf{1}) = \frac{\lambda_0}{2}(C\mathbf{1})^\top \mathbf{I}(C\mathbf{1}) \\
&\leq \frac{1}{2}nC^2\lambda_0,
\end{aligned}
$$

where $\mathbf{K} = \mathbf{V}\mathbf{D}\mathbf{V}^{-1}$ is the eigendecomposition of the PSD matrix $\mathbf{K}$, $\lambda_0 = \max_{1\leq j\leq s}\lambda_{\max}(\mathbf{K_j})$, and $\lambda_{\max}(\mathbf{Z})$ stands for the maximum eigenvalue of matrix $\mathbf{Z}$. Similarly, by using Eq. (A.15) we can bound

$|\widehat{\mathbf{g}}^{\gamma}(\boldsymbol{\beta}^t, \boldsymbol{\gamma}^t)|_\infty$ as

$$|\widehat{\mathbf{g}}^{\gamma}(\boldsymbol{\beta}^t, \boldsymbol{\gamma}^t)|_\infty = \max_{1 \le k \le m} \{\widehat{g}_k^{\gamma}(\boldsymbol{\beta}^t, \boldsymbol{\gamma}^t) \le \frac{m}{\delta} \max\left(nC, \frac{\lambda_0}{2} nC^2\right).$$

Next, we have the bound simplified as

$$\mathrm{E}\left[\sum_{t=1}^{T} \Delta\left(\boldsymbol{\beta}^t, \boldsymbol{\gamma}^t\right)\right] \le \frac{1}{\eta_\gamma}\left(\ln m + \ln s\right) + \eta_\gamma T\left(d\frac{m^2}{2\delta^2}\lambda_0^2 n^2 C^4 + n^2 C^2\right),$$

where d is a constant. We complete the proof by using the fact $\Delta(\boldsymbol{\beta}, \boldsymbol{\gamma})$ is jointly convex in both $\boldsymbol{\beta}$ and $\boldsymbol{\gamma}$; therefore, $\sum_{t=1}^{T} \Delta\left(\boldsymbol{\beta}^t, \boldsymbol{\gamma}^t\right) \ge T\Delta\left(\hat{\boldsymbol{\beta}}, \hat{\boldsymbol{\gamma}}\right)$. □

Corollary 11. *With $\delta = m^{\frac{2}{3}}$ and $\eta_\gamma = \frac{1}{n} m^{-\frac{1}{3}}\sqrt{(\ln m)/T}$, after running Algorithm 3 over T iterations, we have $\mathrm{E}[\Delta(\hat{\boldsymbol{\beta}}, \hat{\boldsymbol{\gamma}})] \le O(m^{1/3}\sqrt{(\ln m)/T})$ in terms of m and T.*

A.4 Proofs for Chapter 4

A.4.1 Proof of Theorem 3

For notational convenience, let us define

$$\Delta_{k,l}^i = \frac{y_k^i - y_l^i}{2}\langle f_k - f_l, \kappa(\mathbf{x}^{\mathbf{i}}, \cdot)\rangle_{H_\kappa}$$

Using this, the objective function in Eq. (4.2) can be rewritten as follows

$$h(f) = \frac{1}{2}\sum_{l=1}^{m}\langle f_l, f_l\rangle_{H_K} + C\sum_{i=1}^{n}\sum_{l,k=1}^{m} I(y_l^i \ne y_k^i)\ell\left(\Delta_{k,l}^i\right)$$

We then rewrite $\ell(z)$ as

$$\ell(z) = \max_{x \in [0,1]}(x - xz)$$

Using the above expression for $\ell(z)$, the second term in $h(f)$ can be rewritten as,

$$\sum_{i=1}^{n}\sum_{l,k=1}^{m} I(y_l^i \neq y_k^i) \max_{\gamma_{k,l}^i \in [0,C]} \left(\gamma_{k,l}^i - \gamma_{k,l}^i \Delta_{k,l}^i\right)$$

The problem in Eq. (4.2) now becomes a convex-concave optimization problem as

$$\min_{f_l \in \mathcal{H}_m} \max_{\gamma_{l,k}^i \in [0,C]} g(f,\gamma)$$

where

$$\begin{aligned} g(f,\gamma) &= \sum_{i=1}^{n}\sum_{l,k=1}^{m} I(y_l^i \neq y_k^i)\gamma_{l,k}^i + \frac{1}{2}\sum_{l=1}^{m}\langle f_l, f_l\rangle_{H_K} \\ &\quad - \sum_{i=1}^{n}\sum_{l,k=1}^{m} I(y_i^l \neq y_k^i)\gamma_{l,k}^i \Delta_{k,l}^i \end{aligned}$$

According to von Newman's lemma, we can switch minimization with maximization. By taking the minimization over f_l first, we have

$$f_l(x) = \sum_{i=1}^{n} y_l^i \left(\sum_{k=1}^{m} I(y_l^i \neq y_k^i)\gamma_{l,k}^i\right) \kappa(\mathbf{x^i}, \mathbf{x})$$

In the above derivation, we use the relation $I(y_l^i \neq y_k^i)(y_l^i - y_k^i) = 2y_l^i$. To simplify our notation, we introduce $\Gamma_i \in [0,C]^{m\times m}$ where $\Gamma_{l,k}^i = \gamma_{l,k}^i$ if $y_l^i \neq y_k^i$ and zero otherwise. Note that since $\gamma_{l,k}^i = \gamma_{k,l}^i$, we have $\Gamma^i = [\Gamma^i]^\top$. We furthermore introduce the notation $[\Gamma^i]_l$ as the sum of the elements in the lth row, i.e., $[\Gamma^i]_l = \sum_{k=1}^m \Gamma_{l,k}^i$. Using these notations, we have $f_l(\mathbf{x})$ expressed as

$$f_l(\mathbf{x}) = \sum_{i=1}^{n} y_l^i [\Gamma^i]_l \kappa(\mathbf{x^i}, \mathbf{x})$$

Finally, the remaining maximization problem becomes

$$\begin{aligned}
\max \quad & \sum_{i=1}^{n}\sum_{k=1}^{m}[\Gamma^i]_k - \frac{1}{2}\sum_{k=1}^{m}\sum_{i,j=1}^{n}\kappa(\mathbf{x^i},\mathbf{x})y_k^i y_k^j[\Gamma^i]_k[\Gamma^j]_k \\
\text{s. t.} \quad & \Gamma_{k,l}^i = \begin{cases} 0 \le \Gamma_{k,l}^i \le C & y_k^i \neq y_l^i \\ 0 & \text{otherwise} \end{cases} \\
& \Gamma^i = [\Gamma^i]^\top, \quad i = 1,\ldots,n; k,l = 1,\ldots,m
\end{aligned}$$

A.4.2 Proof of Theorem 4 .

It is straightforward to shown $\tau \in Q_1 \rightarrow \tau \in Q_2$. The main challenge is to show the other direction, i.e., $\tau \in Q_2 \rightarrow \tau \in Q_1$. For a given τ, in order to check if there exists $Z \in [0,C]^{a\times b}$ such that $\tau 1 : a = Z\mathbf{1}_b$ and $\tau_{a+1:m} = Z^\top \mathbf{1}_a$, we need show that the following optimization problem is feasible

$$\begin{aligned}
\min \quad & 0 \\
\text{s. t.} \quad & Z \in \mathbb{R}_+^{a\times b}, \tau 1 : a = Z\mathbf{1}_b,\ \tau_{a+1:m} = Z^\top \mathbf{1}_a
\end{aligned} \tag{A.21}$$

For the convenience of presentation, we denote by $\mu_a = \tau_{1:a} \in \mathbb{R}^a$, and by $\mu_b = \tau_{a+1:K} \in \mathbb{R}^b$, and rewrite the above feasibility problem as

$$\begin{aligned}
\min \quad & 0 \\
\text{s. t.} \quad & Z \in [0,C]^{a\times b}, \mu_a = Z\mathbf{1}_b,\ \mu_b = Z^\top \mathbf{1}_a
\end{aligned} \tag{A.22}$$

It is important to note that, for the above optimization problem, its optimal value is 0 when the solution is feasible, and $+\infty$ when no feasible solution satisfies the condition. By introducing the Lagrangian multipliers $\lambda_a \in \mathbb{R}^a$ for $\mu_a = Z\mathbf{1}_b$ and $\lambda_b \in \mathbb{R}^b$ for $\mu_b = Z^\top \mathbf{1}_b$, we have

$$\min_{Z\succeq 0}\ \max_{\lambda_a,\lambda_b}\ \lambda_a^\top(\mu_a - Z\mathbf{1}_b) + \lambda_b^\top(\mu_b - Z^\top \mathbf{1}_a) \tag{A.23}$$

By taking the minimization over Z, we have

$$\begin{aligned} \max_{\lambda_a,\lambda_b} \quad & \lambda_a^\top \mu_a + \lambda_b^\top \mu_b \\ \text{s. t.} \quad & \lambda_a \mathbf{1}_b^\top + \mathbf{1}_a \lambda_b^\top \preceq \mathbf{0} \end{aligned} \tag{A.24}$$

To decide if there is a feasible solution to Eq. (A.22), the necessary and sufficient condition is that the optimal value for Eq. (A.24) is zero. First, we show that the objective function of Eq. (A.24) is upper bounded by zero under the constraint $\lambda_a \mathbf{1}_b^\top + \mathbf{1}_a \lambda_b^\top \preceq \mathbf{0}$. We denote by λ_a^+ and λ_b^+ the maximum elements in vector λ_a and λ_b, respectively, i.e, $\lambda_a^+ = \max_{1 \le i \le a} [\lambda_a]^i$ and $\lambda_b^+ = \max_{1 \le i \le b} [\lambda_b]^i$. Evidently, according to the constraint $\lambda_a \mathbf{1}_b^\top + \mathbf{1}_a \lambda_b^\top \preceq \mathbf{0}$, we have $\lambda_a^+ + \lambda_b^+ \le 0$. We then have the objective function bounded as

$$\lambda_a^\top \mu_a + \lambda_b^\top \mu_b \le \lambda_a^+ \mathbf{1}_a^\top \mu_a + \lambda_b^+ \mathbf{1}_b^\top \mu_b = (\lambda_a^+ + \lambda_b^+) \mathbf{1}_a^\top \mu_a \le 0$$

Second, it is straightforward to verify that zero optimal value is obtainable by setting $\lambda_a = \mathbf{0}_a$ and $\lambda_b = \mathbf{0}_b$. Combining the above two arguments, we have the optimal value for Eq. (A.24) is zero, which therefore indicates that there is a feasible solution to Eq. (A.22). By this, we prove that $\tau \in Q_2 \rightarrow \tau \in Q_1$.

A.4.3 Proof of Theorem 6

We first turn the problem in Eq. (4.15) into the following min-max problem

$$\begin{aligned} \max_{\boldsymbol{\alpha}^{\mathbf{i}} \in [0,C]^m} \min_{\lambda} \quad & \sum_{l=1}^{m} \alpha_l^i - \frac{1}{2} \sum_{k=1}^{m} y_k^i f_k^{-i}(\mathbf{x}^{\mathbf{i}}) \alpha_k^i - \\ & \frac{\kappa(\mathbf{x}^{\mathbf{i}}, \mathbf{x}^{\mathbf{i}})}{2} \sum_{k=1}^{m} [\alpha_k^i]^2 + \lambda \mathbf{y}^{\mathbf{i}\top} \boldsymbol{\alpha}^{\mathbf{i}} \end{aligned} \tag{A.25}$$

Since the objective function in Eq. (A.25) is convex in λ and concave in $\boldsymbol{\alpha}^{\mathbf{i}}$, therefore according von Neuman's lemma, switching minimization with maximization will not affect the final solution. Thus, we could

obtain the solution by maximizing over $\boldsymbol{\alpha}$, i.e.,

$$\alpha_k^i = \pi_{[0,C]}\left(\frac{1+\lambda y_k^i - \frac{1}{2}y_k^i f_k^{-i}(\mathbf{x^i})}{\kappa(\mathbf{x^i},\mathbf{x^i})}\right)$$

where $\pi_{[0,C]}(x)$ projects x onto the region $[0, C]$. To compute λ, we aim to solve the following equation

$$\sum_{k=1}^{m} y_k^i \pi_{[0,C]}\left(\frac{1+\lambda y_k^i - \frac{1}{2}y_k^i f_k^{-i}(\mathbf{x^i})}{\kappa(\mathbf{x^i},\mathbf{x^i})}\right) = 0 \tag{A.26}$$

Since when $y_k^i = 1$, the projection in Eq. (A.26) is $\pi_{[0,C]}$ and when $y_k^i = -1$, it is $\pi_{[-C,0]}$, we could represent $y_k^i \pi_{[0,C]}\left(\frac{1+\lambda y_k^i - \frac{1}{2}y_k^i f_k^{-i}(\mathbf{x^i})}{\kappa(\mathbf{x^i},\mathbf{x^i})}\right)$ by $h(\frac{y_k^i+\lambda-\frac{1}{2}f_k^{-i}(\mathbf{x^i})}{\kappa(\mathbf{x^i},\mathbf{x^i})}, y_k^i C)$ where $h(\mathbf{x},\mathbf{y})$ is already defined in the theorem. Since ${\mathbf{y^i}}^\top \boldsymbol{\alpha^i} = 0$, we have the following equation for λ

$$g(\lambda) = \sum_{k=1}^{m} h\left(\frac{y_k^i + \lambda - \frac{1}{2}f_k^{-i}(\mathbf{x^i})}{\kappa(\mathbf{x^i},\mathbf{x^i})}, y_k^i C\right) = 0 \tag{A.27}$$

A.4.4 Proof of Proposition 3

To estimate $\lambda_{\min}$, we rewrite $g(\lambda)$ as

$$g(\lambda) = \sum_{k=1}^{m} I(y_k^i = 1)\pi_{[0,C]}\left(\frac{1+\lambda-\frac{1}{2}f_k^{-i}(\mathbf{x^i})}{\kappa(\mathbf{x^i},\mathbf{x^i})}\right) - \sum_{k=1}^{m} I(y_k^i = -1)\pi_{[0,C]}\left(\frac{1-\lambda+\frac{1}{2}f_k^{-i}(\mathbf{x^i})}{\kappa(\mathbf{x^i},\mathbf{x^i})}\right)$$

To estimate $\lambda_{\min}$, we search for $\lambda_{\min}$ such that $g(\lambda_{\min}) \leq 0$. To this end, we define the following quantity

$$\Delta = \sum_{k=1}^{m} I(y_k^i = 1)\pi_{[0,C]}\left(\frac{1-\frac{1}{2}f_k^{-i}(\mathbf{x^i})}{\kappa(\mathbf{x^i},\mathbf{x^i})}\right) - \sum_{k=1}^{m} I(y_k^i = -1)\pi_{[0,C]}\left(\frac{1+\frac{1}{2}f_k^{-i}(\mathbf{x^i})}{\kappa(\mathbf{x^i},\mathbf{x^i})}\right)$$

If $\Delta \leq 0$, we have $\lambda_{\min} = 0$. Otherwise, we set $\lambda_{\min}$ as the maximum of the following two quantities

$$a_{\min} = -C\kappa(\mathbf{x^i},\mathbf{x^i}) + \min_{y_k^i=-1}\left(1+\frac{1}{2}f_k^{-1}(\mathbf{x^i})\right),\ b_{\min} = -\max_{y_k^i=1}\left(1-\frac{1}{2}f_k^{-i}(\mathbf{x^i})\right)$$

It is evidently that one of the solutions will result into the negative value for $g(\lambda)$ since (a) by setting $\lambda_{\min} = b_{\min}$, we ensure that every $\pi_{[0,C]}(1+\lambda-\frac{1}{2}f_k^{-i}(\mathbf{x^i}))$ is zero, (b) by setting $\lambda_{\min} = a_{\min}$, we have

that every $\pi_{[0,C]}(1-\lambda+\frac{1}{2}f_k^{-i}(\mathbf{x^i}))$ being C.

To obtain $\lambda_{\max}$, we again check $\Delta \geq 0$. If so, $\lambda_{\max}=0$. Otherwise, the solution for $\lambda_{\max}$ should be the minimum of the following two quantities

$$a_{\max}=C\kappa(\mathbf{x}^i,\mathbf{x^i})-\min_{y_k^i=1}\left(1-\frac{1}{2}f_k^{-i}(\mathbf{x^i})\right),\ b_{\max}=\max_{y_k^i=-1}\left(1+\frac{1}{2}f_k^{-i}(\mathbf{x^i})\right)$$

A.5 Proofs for Chapter 5

A.5.1 Proof of Lemma 1

We start proving Lemma 1 by writing the dual function of Eq. (5.5), which is as follows:

$$g(\lambda)=\sup_{\gamma^i}L(\gamma,\lambda)=\sup_{\gamma^i}\sum_{l\notin Y_i}\sum_{k\in Y_i}\gamma_{k,l}^i\ell(f_k(\mathbf{x^i})-f_l(\mathbf{x^i}))+\sum_{l\notin Y_i}\lambda_l(1-\sum_{k\in Y_i}\gamma_{k,l}^2)$$

Since $L(\gamma,\lambda)$ is a concave function, the upper bound is found by setting $\frac{\partial L(\gamma,\lambda)}{\partial\gamma}=0$.

$$g(\lambda)=\sum_{l\notin Y_i}\sum_{k\in Y_i}\frac{\ell^2(f_k(\mathbf{x^i})-f_l(\mathbf{x^i}))}{4\lambda_l}+\sum_{l\notin Y_i}\lambda_l$$

The Lagrange dual is to minimize g over all $\lambda \geq 0$. The optimal λ_l can easily be found as $\sqrt{\sum_{k\in Y_i}\ell^2(f_k(\mathbf{x^i})-f_l(\mathbf{x^i}))}/2$. Therefore, the Lagrange dual form becomes

$$\sum_{l\notin Y_i}\sqrt{\sum_{k\in Y_i}\ell^2(f_k(\mathbf{x^i})-f_l(\mathbf{x^i}))}.$$

This concludes the proof.

A.5.2 Proof of Theorem 9

We can rewrite $\ell(z)$ as

$$\ell(z)=\max_{x\in[0,1]}(x-xz)$$

Using the above expression for $\ell(z)$, the objection function can be rewritten as

$$\min_{f_k \in \mathcal{H}_K} \max_{\gamma^i_{k,l} \in \Delta_i} \max_{\beta^i_{k,l} \in [0,1]} \frac{1}{2} \sum_{k=1}^{m} |f_k|^2_{\mathcal{H}_K} \tag{A.28}$$
$$+C \sum_{i=1}^{n} \sum_{k \in Y^i} \sum_{l \notin Y^i} \gamma^i_{k,l} \beta^i_{k,l} \left(1 - f_k(x_i) + f_l(x_i)\right)$$

The problem now becomes a convex-concave optimization. By defining new variable $\Gamma^i_{k,l}$ as

$$\Gamma^i_{k,l} = \gamma^i_{k,l} \beta^i_{k,l} + \gamma^i_{l,k} \beta^i_{l,k},$$

we rewrite Eq. (A.29) as

$$\min_{f_k \in \mathcal{H}_K} \max_{\Gamma^i_{k,l} \in \Delta_i} \frac{1}{2} \sum_{k=1}^{m} |f_k|^2_{\mathcal{H}_K} \tag{A.29}$$
$$+ \sum_{i=1}^{n} \sum_{k,l=1}^{m} \Gamma^i_{k,l} \left(1 - f_k(x_i) + f_l(x_i)\right)$$

Since Eq. (A.30) is a convex-concave optimization problem, according to von Newman's lemma, we can switch minimization with maximization. By taking the minimization with respect to f_k, we have

$$f_k(x) = C \sum_{i=1}^{n} \left(\sum_{l=1}^{m} \Gamma^i_{k,l} - \sum_{l=1}^{m} \Gamma^i_{l,k} \right) \kappa(x, x_i) \tag{A.30}$$

According to the definition of Δ_i, $\Gamma^i_{k,l}$ is nonzero only when $k \in Y^i$ (i.e., $y^i_k = 1$) and $l \notin Y^i$ (i.e., $y^i_k = -1$). We thus can rewrite $f_k(x)$ in Eq. (A.30) as

$$f_k(x) = C \sum_{i=1}^{n} \left(\sum_{l=1}^{m} \Gamma^i_{k,l} + \sum_{l=1}^{m} \Gamma^i_{l,k} \right) y^k_i \kappa(\mathbf{x}, \mathbf{x^i})$$

By defining $\alpha^i_k = \sum_{l=1}^{m} \Gamma^i_{k,l} + \sum_{l=1}^{m} \Gamma^i_{l,k}$, we have the result in the theorem.

A.5.3 Proof of Lemma 2

First, using the notation of $\mathbf{h}_k$, we rewrite the objective function in Eq. (5.15) as

$$\max_{\gamma \in \Delta} -CK_{i,i}\eta \sum_{s=1}^{b} |\boldsymbol{\gamma}_{\cdot,\mathbf{s}}|_2^2 + 2\sum_{s=1}^{b} \mathbf{h}_s^\top \boldsymbol{\gamma}_{\cdot,\mathbf{s}}$$

Since all $\gamma_{\cdot,s}, s = 1, \ldots, b$ are decoupled in both the domain Δ and the objective function, we can decompose the above problem into b independent optimization problems,

$$\max_{\boldsymbol{\gamma}_{\cdot,\mathbf{s}} \in \mathbb{R}_+^a} \left\{ -CK_{i,i}\eta |\boldsymbol{\gamma}_{\cdot,\mathbf{s}}|_2^2 + 2\mathbf{h}_s^\top \boldsymbol{\gamma}_{\cdot,\mathbf{s}} : |\boldsymbol{\gamma}_{\cdot,\mathbf{s}}|_2 \leq 1 \right\} \quad , \tag{A.31}$$

where $s = 1, \ldots, b$. For each independent optimization problem, we introduce a Lagrangian multiplier $\lambda_s \geq 0$ for constraint $|\boldsymbol{\gamma}_{\cdot,\mathbf{s}}|_2 \leq 1$, and have

$$\min_{\lambda_s \geq 0} \max_{\boldsymbol{\gamma}_{\cdot,\mathbf{s}} \in \mathbb{R}_+^a} -(CK_{i,i}\eta + \lambda_s)|\gamma_{\cdot,s}|_2^2 + 2\mathbf{h}_s^\top \boldsymbol{\gamma}_{\cdot,\mathbf{s}} + \lambda_s$$

The optimal solution to the maximization of $\boldsymbol{\gamma}$ is

$$\boldsymbol{\gamma}_{\cdot,\mathbf{s}} = \pi_{\mathcal{G}} \left(\frac{\mathbf{h}_s}{\lambda_s + CK_{i,i}\eta} \right)$$

In order to decide the value for λ_s, we use the complementary slackness condition, i.e., $\lambda_s(|\boldsymbol{\gamma}_{\cdot,\mathbf{s}}|_2^2 - 1) = 0$. There are two cases: $\lambda = 0$ implies $|\boldsymbol{\gamma}_{\cdot,\mathbf{s}}|_2^2 \leq 1$, and $\lambda > 0$ implies $|\boldsymbol{\gamma}_{\cdot,\mathbf{s}}|_2^2 = 1$. This leads to the result stated in the Lemma.

BIBLIOGRAPHY

BIBLIOGRAPHY

[1] M. P. Szymczak, Flickr photo, http://www.flickr.com/photos/marooned/.

[2] D. Wild, Flickr photo, http://www.flickr.com/people/publicenergy/.

[3] L. Fei-fei, R. Fergus, S. Member, and P. Perona, "One-shot learning of object categories," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 28, no. 4, pp. 594 – 611, 2006.

[4] Q. Chen, Z. Song, S. Liu, X. Chen, X. Yuan, T.-S. Chua, S. Yan, Y. Hua, Z. Huang, and S. Shen, "Boosting classification with exclusive context," in *In PASCAL Visual Object Classes Challenge Workshop*, 2010.

[5] J. Yang, Y. Li, Y. Tian, L. Duan, and W. Gao, "Group-sensitive multiple kernel learning for object categorization," in *IEEE Int. Conference on Computer Vision*, 2009, pp. 436–443.

[6] A. Vedaldi, V. Gulshan, M. Varma, and A. Zisserman, "Multiple kernels for object detection," in *IEEE Int. Conference on Computer Vision*, 2009, pp. 606–613.

[7] A. Jiang, C. Wang, and Y. Zhu, "Calibrated rank-svm for multi-label image categorization," in *Proc. of IEEE Int. Joint Conference on Neural Networks*, 2008, pp. 1450–1455.

[8] A. Znaidia, H. Le Borgne, and C. Hudelot, "Belief theory for large-scale multi-label image classification," in *Belief Functions: Theory and Applications*. Springer, 2012, vol. 164, pp. 205–212.

[9] S. S. Bucak, R. Jin, and A. Jain, "Multi-label multiple kernel learning by stochastic approximation: Application to visual object recognition," in *Proc. of Neural Information Processing Systems*, 2010, pp. 325–333.

[10] N. Ueda and K. Saito, "Parametric mixture models for multi-labeled text," in *Proc. of Neural Information Processing Systems*, 2002, pp. 721–728.

[11] S. Shalev-Shwartz and Y. Singer, "Efficient learning of label ranking by soft projections onto polyhedra," *Journal of Machine Learning Research*, vol. 7, pp. 1567–1599, 2006.

[12] N. Ghamrawi and A. McCallum, "Collective multi-label classification," in *Proc. of ACM Int. Conference on Information and Knowledge Management*, 2005, pp. 195–200.

[13] Y. Liu, R. Jin, and L. Yang, "Semi-supervised multi-label learning by constrained non-negative matrix factorization," in *Proc. of Conference on Artificial Intelligence*, 2006, pp. 421–426.

[14] F. Sun, J. Tang, H. Li, G.-J. Qi, and T. Huang, "Multi-label image categorization with sparse factor representation," *IEEE Transactions on Image Processing*, vol. 23, no. 3, pp. 1028–1037, 2014.

[15] S. S. Bucak, P. K. Mallapragada, R. Jin, and A. K. Jain, "Efficient multi-label ranking for multi-class learning: Application to object recognition," in *Proc. of IEEE Int. Conference on Computer Vision*, 2009, pp. 2098–2105.

[16] Y.-Y. Lin, J.-F. Tsai, and T.-L. Liu, "Efficient discriminative local learning for object recognition," in *Proc. of IEEE Int. Conference on Computer Vision*, 2009, pp. 598–605.

[17] D. Hall, "A system for object class detection," in *Cognitive Vision Systems*. Springer, 2006, pp. 73–85.

[18] B. M. Sadler and G. B. Giannakis, "Shift-and rotation-invariant object reconstruction using the bispectrum," *Journal of the Optical Society of America A*, vol. 9, no. 1, pp. 57–69, 1992.

[19] R. Fergus, P. Perona, and A. Zisserman, "Weakly supervised scale-invariant learning of models for visual recognition," *International Journal of Computer Vision*, vol. 71, no. 3, pp. 273–303, 2007.

[20] L. Spirkovska and M. B. Reid, "Robust position, scale, and rotation invariant object recognition using higher-order neural networks," *Pattern Recognition*, vol. 25, no. 9, pp. 975–985, 1992.

[21] T. Kadir, A. Zisserman, and M. Brady, "An affine invariant salient region detector," in *Computer Vision–ECCV*. Springer, 2004, pp. 228–241.

[22] A. Diplaros, T. Gevers, and I. Patras, "Combining color and shape information for illumination-viewpoint invariant object recognition," *IEEE Transactions on Image Processing*, vol. 15, no. 1, pp. 1–11, 2006.

[23] E. Hsiao and M. Hebert, "Occlusion reasoning for object detection under arbitrary viewpoint," in *Proc. of IEEE Conference on Computer Vision and Pattern Recognition*, 2012, pp. 3146–3153.

[24] A. Selinger and R. C. Nelson, "Improving appearance-based object recognition in cluttered backgrounds," in *Proc. of Int. Conference on Pattern Recognition*, 2000, pp. 46–50.

[25] S. Dickinson, "The evolution of object categorization and the challenge of image abstraction," in *Object Categorization: Computer and Human Vision Perspectives*. Cambridge University Press, 2009, pp. 1–37.

[26] S. Maji, A. C. Berg, and J. Malik, "Efficient classification for additive kernel SVMs," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 35, no. 1, pp. 66–77, 2013.

[27] S. Har-Peled, D. Roth, and D. Zimak, "Constraint classification for multiclass classification and ranking," in *Proc. of Neural Information Processing Systems*, 2002, pp. 809–816.

[28] S. S. Bucak, R. Jin, and A. K. Jain, "Multi-label learning with incomplete class assignments," in *Proc. of IEEE Conference on Computer Vision and Pattern Recognition*, 2011, pp. 2801–2808.

[29] K. Yu and W. Chu, "Gaussian process models for link analysis and transfer learning," in *Proc. of European Symp. on Artificial Neural Networks*, 2008, pp. 1657–1664.

[30] N. Loeff and A. Farhadi, "Scene discovery by matrix factorization," in *Computer Vision–ECCV*. Springer, 2008, pp. 451–464.

[31] Amazon Mechanical Turk, https://www.mturk.com/mturk.

[32] B. Scholkopf and A. J. Smola, *Learning with Kernels: Support Vector Machines, Regularization, Optimization, and Beyond*. Cambridge, MA, USA: MIT Press, 2001.

[33] J. Zhang, S. Lazebnik, and C. Schmid, "Local features and kernels for classification of texture and object categories: a comprehensive study," *International Journal of Computer Vision*, vol. 73, no. 2, pp. 213–238, 2007.

[34] M. A. Tahir, K. van de Sande, J. Uijlings, F. Yan, X. Li, K. Mikolajczyk, J. Kittler, T. Gevers, and A. Smeulders, "SurreyUVA_SRKDA method," in *PASCAL Visual Object Classes Challenge Workshop*, 2008.

[35] F. R. Bach, G. R. G. Lanckriet, and M. I. Jordan, "Multiple kernel learning, conic duality, and the SMO algorithm," in *Proc. of Int. Conference on Machine Learning*, 2004, pp. 6–13.

[36] Z. Wang, S. Chen, and T. Sun, "MultiK-MHKS: A novel multiple kernel learning algorithm," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 30, no. 2, pp. 348–353, 2008.

[37] D. P. Lewis, T. Jebara, and W. S. Noble, "Nonstationary kernel combination," in *Proc. of Int. Conference on Machine Learning*, 2006, pp. 553–560.

[38] L. Jie, F. Orabona, M. Fornoni, B. Caputo, and N. Cesa-bianchi, "OM-2: An online multi-class multi-kernel learning algorithm," in *Proc. of IEEE Online Learning for Computer Vision Workshop*, 2010, pp. 43–50.

[39] J. Saketha Nath, G. Dinesh, S. Raman, C. Bhattacharyya, A. Ben-Tal, and K. Ramakrishan, "On the algorithmics and applications of a mixed-norm based kernel learning formulation," in *Proc. of Neural Information Processing Systems*, 2009, pp. 844–852.

[40] P. V. Gehler and S. Nowozin, "Let the kernel figure it out: Principled learning of pre-processing for kernel classifiers," in *Proc. of IEEE Int. Conference on Computer Vision*, 2009, pp. 2836 – 2843.

[41] F. Yan, K. Mikolajczyk, M. Barnard, H. Cai, and J. Kittler, "Lp norm multiple kernel fisher discriminant analysis for object and image categorisation," in *Proc. of IEEE Conference on Computer Vision and Pattern Recognition*, 2010, pp. 3626–3632.

[42] S. Nakajima, A. Binder, C. Mller, W. Wojcikiewicz, M. Kloft, U. Brefeld, K.-R. Mller, and M. Kawanabe, "Multiple kernel learning for object classification," in *Workshop on Information-based Induction Sciences*, 2009.

[43] P. V. Gehler and S.Nowozin, "On feature combination for multiclass object classification," in *Proc. of Int. Conference on Machine Learning*, 2009, pp. 221–228.

[44] J. Ren, Z. Liang, and S. Hu, "Multiple kernel learning improved by MMD," in *Proc. of Int. Conference on Advanced Data Mining and Applications*, 2010, pp. 63–74.

[45] C. Cortes, M. Mohri, and A. Rostamizadeh, "Learning non-linear combinations of kernels," in *Proc. of Neural Information Processing Systems*, 2009, pp. 396–404.

[46] Z. Xu, R. Jin, H. Yang, I. King, and M. R. Lyu, "Simple and efficient multiple kernel learning by group lasso," in *Proc. of Int. Conference on Machine Learning*, 2010, pp. 1175–1182.

[47] C. Cortes, M. Mohri, and A. Rostamizadeh, "L_2 regularization for learning kernels," in *Proc. of Conference on Uncertainty in Artificial Intelligence*, 2009, pp. 109–116.

[48] Z. Xu, R. Jin, S. Zhu, M. Lyu, and I. King, "Smooth optimization for effective multiple kernel learning," in *Proc. of Conference on Artificial Intelligence*, 2010, pp. 637–642.

[49] R. Tomioka and T. Suzuki, "Sparsity-accuracy trade-off in MKL," in *NIPS Workshop on Understanding Multiple Kernel Learning Methods*, 2009.

[50] F. Yan, K. Mikolajczyk, J. Kittler, and A. Tahir, "A comparison of L1 norm and L2 norm multiple kernel SVMs in image and video classification," in *Int. Workshop on Content-Based Multimedia Indexing*, 2009.

[51] A. Rakotomamonjy, F. Bach, S. Canu, and Y. Grandvalet, "More efficiency in multiple kernel learning," in *Proc. of Int. Conference on Machine Learning*, 2007, pp. 775–782.

[52] Z. Xu, R. Jin, I. King, and M. R. Lyu, "An extended level method for efficient multiple kernel learning," in *Proc. of Neural Information Processing Systems*, 2009, pp. 1825–1832.

[53] A. Rakotomamonjy, F. Bach, Y. Grandvalet, and S. Canu, "SimpleMKL," *Journal of Machine Learning Research*, vol. 9, no. 11, pp. 2491–2521, 2008.

[54] G. Lanckriet, N. Cristianini, P. Bartlett, and L. E. Ghaoui, "Learning the kernel matrix with semidefinite programming," *Journal of Machine Learning Research*, vol. 5, pp. 27–72, 2004.

[55] S. Sonnenburg, G. Rätsch, and C. Schäfer, "A general and efficient multiple kernel learning algorithm," in *Proc. of Neural Information Processing Systems*, 2006, pp. 1273–1280.

[56] M. Kloft, U. Brefeld, S. Sonnenburg, and A. Zien, "Lp-norm multiple kernel learning," *Journal of Machine Learning Research*, vol. 12, pp. 953–997, 2011.

[57] M. Kowalski, M. Szafranski, and L. Ralaivola, "Multiple indefinite kernel learning with mixed norm regularization," in *Proc. of Int. Conference on Machine Learning*, 2009, pp. 545–552.

[58] C. Cortes, M. Mohri, and A. Rostamizadeh, "Generalization bounds for learning kernels," in *Proc. of Int. Conference on Machine Learning*, 2010, pp. 247–254.

[59] Z. Hussain and J. Shawe-Taylor, "A note on improved loss bounds for multiple kernel learning," *arXiv preprint arXiv:1106.6258*, 2011.

[60] M. Kloft, U. Rückert, and P. L. Bartlett, "A unifying view of multiple kernel learning," in *Proc. of European Conference on Machine Learning and Knowledge Discovery in Databases*, 2010, pp. 66–81.

[61] M. Kloft, U. Brefeld, S. Sonnenburg, P. Laskov, K.-R. Müller, and A. Zien, "Efficient and accurate lp-norm multiple kernel learning," in *Proc. of Neural Information Processing Systems*, 2009, pp. 997–1005.

[62] K. Gai, G. Chen, and C. Zhang, "Learning kernels with radiuses of minimum enclosing balls," in *Proc. of Neural Information Processing Systems*, 2010, pp. 649–657.

[63] M. Varma and B. R. Babu, "More generality in efficient multiple kernel learning," in *Proc. of Int. Conference on Machine Learning*, 2009, pp. 1065–1072.

[64] J. Aflalo, A. Ben-Tal, C. Bhattacharyya, J. S. Nath, and S. Raman, "Variable sparsity kernel learning," *Journal of Machine Learning Research*, vol. 12, pp. 565–592, 2011.

[65] F. Bach, "Exploring large feature spaces with hierarchical multiple kernel learning," in *Proc. of Neural Information Processing Systems*, 2009, pp. 105–112.

[66] J. Yang, Y. Li, Y. Tian, L.-Y. Duan, and W. Gao, "Per-sample multiple kernel approach for visual concept learning," *EURASIP Journal on Image and Video Processing*, vol. 2010, no. 2, pp. 1–13, January 2010.

[67] M. Gnen and E. Alpaydin, "Localized multiple kernel learning," in *Proc. of Int. Conference on Machine Learning*, 2008, pp. 352–359.

[68] S. Ji, L. Sun, R. Jin, and J. Ye, "Multi-label multiple kernel learning," in *Proc. of Neural Information Processing Systems*, 2009, pp. 777–784.

[69] M. Varma and D. Ray, "Learning the discriminative power-invariance trade-off," in *Proc. of IEEE Int. Conference on Computer Vision*, 2007, pp. 1–8.

[70] S. Vishwanathan, Z. Sun, N. Ampornpunt, and M. Varma, "Multiple kernel learning and the SMO algorithm," in *Proc. of Neural Information Processing Systems*, 2010, pp. 2361–2369.

[71] S. Sonnenburg, G. Rätsch, C. Schäfer, and B. Schölkopf, "Large scale multiple kernel learning," *Journal of Machine Learning Research*, vol. 7, pp. 1531–1565, 2006.

[72] H. Liu and L. Yu, "Toward integrating feature selection algorithms for classification and clustering," *IEEE Transactions on Knowledge and Data Engineering*, vol. 17, no. 4, pp. 491–502, 2005.

[73] F. Bach, "Consistency of the group lasso and multiple kernel learning," *Journal of Machine Learning Research*, vol. 9, pp. 1179–1225, 2008.

[74] A. Bosch, A. Zisserman, and X. Munoz, "Representing shape with a spatial pyramid kernel," in *Proc. of ACM Int. Conference on Image and Video Retrieval*, 2007, pp. 401–408.

[75] T. Hertz, "Learning distance functions: Algorithms and applications," Ph.D. dissertation, The Hebrew University of Jerusalem, 2006.

[76] N. Cristianini, J. Kandola, A. Elisseeff, and J. Shawe-Taylor, "On kernel-target alignment," in *Proc. of Neural Information Processing Systems*, 2002, pp. 367–373.

[77] O. Chapelle, J. Weston, and B. Schlkopf, "Cluster kernels for semi-supervised learning," in *Proc. of Neural Information Processing Systems*, 2003, pp. 585–592.

[78] R. I. Kondor and J. Lafferty, "Diffusion kernels on graphs and other discrete structures," in *Proc. of Int. Conference on Machine Learning*, 2002, pp. 315–322.

[79] J. Zhuang, I. W. Tsang, and S. C. H. Hoi, "SimpleNPKL: Simple non-parametric kernel learning," in *Proc. of Int. Conference on Machine Learning*, 2009, pp. 1273–1280.

[80] B. Kulis, M. Sustik, and I. Dhillon, "Learning low-rank kernel matrices," in *Proc. of Int. Conference on Machine Learning*, 2006, pp. 505–512.

[81] S. C. H. Hoi and R. Jin, "Active kernel learning," in *Proc. of Int. Conference on Machine Learning*, 2008, pp. 400–407.

[82] R. Tibshirani, "Regression shrinkage and selection via the lasso," *J. Royal. Statist. Soc B.*, vol. 58, no. 1, pp. 267–288, 1996.

[83] C. Longworth and M. J. Gales, "Multiple kernel learning for speaker verification," in *Proc. of IEEE Int. Conference on Acoustics, Speech and Signal Processing*, 2008, pp. 1581–1584.

[84] V. Sindhwani and A. C. Lozano, "Non-parametric group orthogonal matching pursuit for sparse learning with multiple kernels," in *Proc. of Neural Information Processing Systems*, 2011, pp. 414–431.

[85] Y. Nesterov, *Introductory Lectures on Convex Optimization: A Basic Course*. Springer, 1998.

[86] A. Martins, N. Smith, E. Xing, P. Aguiar, and M. Figueiredo, "Online multiple kernel learning for structured prediction," in *NIPS Workshop on New Directions in Multiple Kernel Learning*, 2010.

[87] A. Zien and S. Cheng, "Multiclass multiple kernel learning," in *Proc. of Int. Conference on Machine Learning*, 2007, pp. 1191–1198.

[88] J. F. Sturm, "Using sedumi 1. 02, a matlab toolbox for optimization over symmetric cones," *Optimization Methods and Software*, vol. 11-12, pp. 625–653, 1999.

[89] "The MOSEK optimization software." [Online]. Available: http://www.mosek.com/

[90] J. C. Platt, *Fast Training of Support Vector Machines Using Sequential Minimal Optimization*. Cambridge, MA, USA: MIT Press, 1999, pp. 185–208.

[91] R. Jin, S. C. H. Hoi, and T. Yang, "Online multiple kernel learning: Algorithms and mistake bounds," in *Proc. of Int. Conference on Algorithmic Learning Theory*, 2010, pp. 390–404.

[92] F. Rosenblatt, "The perceptron: A probabilistic model for information storage and organization in the brain," *Psychological Review*, vol. 65, no. 6, pp. 386–408, 1958.

[93] Y. Freund and R. E. Schapire, "A decision-theoretic generalization of on-line learning and an application to boosting," *Journal of Computer and System Sciences*, vol. 55, no. 1, pp. 119–139, 1997.

[94] M. Everingham, L. Van Gool, C. K. I. Williams, J. Winn, and A. Zisserman, "The PASCAL Visual Object Classes Challenge 2007 (VOC2007) Results," http://www.pascal-network.org/challenges/VOC/voc2007/workshop/index.html.

[95] T. Ojala, M. Pietikainen, and T. Maenpaa, "Multiresolution grayscale and rotation invariant texture classification with local binary patterns," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 24, no. 7, pp. 971–987, 2002.

[96] S. Belongie, J. Malik, and J. Puzicha, "Shape matching and object recognition using shape contexts," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 24, no. 4, pp. 509–522, 2002.

[97] N. Pinto, D. D. Cox, and J. J. Dicarlo, "Why is real-world visual object recognition hard?" *PLoS Computational Biology*, vol. 4, no. 1, 2008.

[98] O. Tuzel, F. Porikli, and P. Meer, "Region covariance: A fast descriptor for detection and classification," in *Computer Vision–ECCV*. Springer, 2006, pp. 589–600.

[99] A. Berg and J. Malik, "Geometric blur for template matching," in *Proc. of IEEE Conference on Computer Vision and Pattern Recognition*, 2001, pp. 607–614.

[100] E. Shechtman and M. Irani, "Matching local self-similarities across images and videos," in *Proc. of IEEE conference on Computer Vision and Pattern Recognition*, 2007, pp. 607–614.

[101] M. Guillaumin, T. Mensink, J. Verbeek, and C. Schmid, "TagProp: Discriminative metric learning in nearest neighbor models for image auto-annotation," in *Proc. of IEEE Int. Conference on Computer Vision*, 2009, pp. 309–316.

[102] A. Oliva and A. Torralba, "Modeling the shape of the scene: A holistic representation of the spatial envelope," *International Journal of Computer Vision*, vol. 42, no. 3, pp. 145–175, 2001.

[103] S. Lazebnik, C. Schmid, and J. Ponce, "Beyond bags of features: Spatial pyramid matching for recognizing natural scene categories," in *Proc. of IEEE Conference on Computer Vision and Pattern Recognition*, 2006, pp. 2169 – 2178.

[104] K. Mikolajczyk and C. Schmid, "Indexing based on scale invariant interest points," in *Proc. of IEEE Int. Conference on Computer Vision*, 2001, pp. 525–531.

[105] J. van de Weijer and C. Schmid, "Coloring local feature extraction," in *Computer Vision–ECCV*. Springer, 2006, pp. 334–348.

[106] F. Perronnin, J. Sanchez, and Y. Liu, "Large-scale image categorization with explicit data embedding," in *Proc. of IEEE Conference on Computer Vision and Pattern Recognition*, 2010, pp. 2297 –2304.

[107] C.-C. Chang and C.-J. Lin, "LIBSVM: A library for support vector machines," *ACM Transactions on Intelligent Systems and Technology*, vol. 2, pp. 21–27, 2011.

[108] M. Grant and S. Boyd., "CVX: Matlab software for disciplined convex programming, version 1.21," http://cvxr.com/cvx, april 2011.

[109] F. Bach, R. Thibaux, and M. I. Jordan, "Computing regularization paths for learning multiple kernels," in *Proc. of Neural Information Processing Systems*, 2005, pp. 73–80.

[110] F. Li, J. Carreira, and C. Sminchisescu, "Object recognition as ranking holistic figure-ground hypotheses," in *Proc. of IEEE Conference on Computer Vision and Pattern Recognition*, 2010, pp. 1712 – 1719.

[111] G. L. Oliveira, E. R. Nascimento, A. W. Vieira, and M. F. M. Campos, "Sparse spatial coding: A novel approach for efficient and accurate object recognition," in *Proc. of IEEE Int. Conference on Robotics and Automation*, 2012, pp. 2592–2598.

[112] Z. Song, Q. Chen, Z. Huang, Y. Hua, and S. Yan, "Contextualizing object detection and classification," in *Proc of IEEE Int. Conference on Computer Vision and Pattern Recognition*, 2011, pp. 1585 – 1592.

[113] H. Harzallah, F. Jurie, and C. Schmid, "Combining efficient object localization and image classification," in *Proc. of Int. Conference on Computer Vision*, 2009, pp. 237–244.

[114] K. Chatfield, V. Lemtexpitsky, A. Vedaldi, and A. Zisserman, "The devil is in the details: an evaluation of recent feature encoding methods," in *Proc. of the British Machine Vision Conference*, 2011, pp. 1–12.

[115] S. Sonnenburg, G. Ratsch, S. Henschel, C. Widmer, J. Behr, A. Zien, F. d. Bona, A. Binder, C. Gehl, and V. Franc, "The shogun machine learning toolbox," *The Journal of Machine Learning Research*, vol. 99, pp. 1799–1802, 2010.

[116] L. Tang, J. Chen, and J. Ye, "On multiple kernel learning with multiple labels," in *Proc. of Int. Joint Conference on Artifical Intelligence*, 2009, pp. 1255–1260.

[117] F. Orabona, L. Jie, and B. Caputo, "Online-batch strongly convex multi kernel learning," in *Proc. of IEEE Conference on Computer Vision and Pattern Recognition*, 2010, pp. 787–794.

[118] S. Mei, "Multi-kernel transfer learning based on chou's pseaac formulation for protein submitochondria localization," *Journal of Theoretical Biology*, vol. 293, pp. 121–130, 2012.

[119] A. Nemirovski, "Prox-method with rate of convergence o(1/t) for variational inequalities with lipschitz continuous monotone operators and smooth convex-concave saddle point problems," *SIAM Journal on Optimization*, vol. 15, no. 1, pp. 229–251, 2004.

[120] T. G. Dietterich and G. Bakiri, "Solving multiclass learning problems via error-correcting output codes," *Journal of Artificial Intelligence Research*, vol. 2, no. 1, pp. 263–286, 1995.

[121] C.-W. Hsu and C.-J. Lin, "A comparison of methods for multi-class support vector machines," *IEEE Transactions on Neural Networks*, vol. 13, no. 2, pp. 415–425, 2002.

[122] D. Hsu, S. M. Kakade, J. Langford, and T. Zhang, "Multi-label prediction via compressed sensing," in *Proc. of Neural Information Processing Systems*, 2009, pp. 772–780.

[123] T. Zhou, D. Tao, and X. Wu, "Compressed labeling on distilled labelsets for multi-label learning," *Machine Learning*, vol. 88, no. 1-2, pp. 69–126, 2012.

[124] F. Tai and H.-T. Lin, "Multilabel classification with principal label space transformation," *Neural Computation*, vol. 24, no. 9, pp. 2508–2542, 2012.

[125] M.-L. Zhang and Z.-H. Zhou, "ML-KNN: A lazy learning approach to multi-label learning," *Pattern Recognition*, vol. 40, no. 7, pp. 2038–2048, 2007.

[126] R. E. Schapire and Y. Singer, "Improved boosting algorithms using confidence-rated predictions," *Machine Learning*, vol. 37, no. 3, pp. 297–336, 1999.

[127] J. R. Quinlan, *C4.5: programs for machine learning*. San Francisco, CA, USA: Morgan Kaufmann Publishers Inc., 1993.

[128] A. Clare and R. D. King, "Knowledge discovery in multi-label phenotype data," in *Principles of Data Mining and Knowledge Discovery*. Springer, 2001, pp. 42–53.

[129] Y. Freund and L. Mason, "The alternating decision tree learning algorithm," in *Proc. of Int. Conference on Machine Learning*, 1999, pp. 124–133.

[130] H. Blockeel, M. Bruynooghe, S. Dzeroski, J. Ramon, and J. Struyf, "Hierarchical multi-classification," in *ACM SIGKDD Workshop on Multi-Relational Data Mining*, 2002.

[131] F. De Comité, R. Gilleron, and M. Tommasi, "Learning multi-label alternating decision trees from texts and data," in *Machine Learning and Data Mining in Pattern Recognition*. Springer, 2003, pp. 35–49.

[132] J. Struyf, S. Dzeroski, H. Blockeel, and A. Clare, "Hierarchical multi-classification with predictive clustering trees in functional genomics," ser. EPIA. Springer-Verlag, 2005, pp. 272–285.

[133] Y. Han, F. Wu, Y. Zhuang, and X. He, "Multi-label transfer learning with sparse representation," *IEEE Transactions on Circuits and Systems for Video Technology*, vol. 20, no. 8, pp. 1110–1121, 2010.

[134] G.-J. Qi, C. Aggarwal, Y. Rui, Q. Tian, S. Chang, and T. Huang, "Towards cross-category knowledge propagation for learning visual concepts," in *Proc. of IEEE Conference on Computer Vision and Pattern Recognition*, 2011, pp. 897–904.

[135] K. Crammer, O. Dekel, J. Keshet, S. Shalev-Shwartz, and Y. Singer, "Online passive-aggressive algorithms," *Journal of Machine Learning Research*, vol. 7, pp. 551–585, 2006.

[136] A. Elisseeff and J. Weston, "A kernel method for multi-labelled classification," in *Proc. of Neural Information Processing Systems*, 2001, pp. 681–687.

[137] O. Dekel, C. D. Manning, and Y. Singer, "Log-linear models for label ranking," in *Proc. of Neural Information Processing Systems*, 2003.

[138] A. McCallum, "Multi-label text classification with a mixture model trained by EM," in *AAAI Workshop on Text Learning*, 1999.

[139] S. Ji, L. Tang, S. Yu, and J. Ye, "Extracting shared subspace for multi-label classification," in *Proc. of ACM SIGKDD Int. Conference on Knowledge Discovery and Data Mining*, 2008, pp. 381–389.

[140] K. Yu, S. Yu, and V. Tresp, "Multi-label informed latent semantic indexing," in *Proc. of Annual ACM SIGIR Conference on Research and Development in Information Retrieval*, 2005, pp. 258–265.

[141] A. Quattoni, M. Collins, and T. Darrell, "Transfer learning for image classification with sparse prototype representations," in *Proc. of IEEE Conference on Computer Vision and Pattern Recognition*, 2008, pp. 1–8.

[142] S.-J. Huang, Y. Yu, and Z.-H. Zhou, "Multi-label hypothesis reuse," in *Proc. of ACM SIGKDD Int. Conference on Knowledge Discovery and Data Mining*, 2012, pp. 525–533.

[143] Y. Guo and S. Gu, "Multi-label classification using conditional dependency networks," in *Proc. of Int. Joint Conference on Artificial Intelligence*, 2011, pp. 1300–1305.

[144] G. Chen, J. Zhang, F. Wang, C. Zhang, and Y. Gao, "Efficient multi-label classification with hypergraph regularization," in *Proc. of IEEE conference on Computer Vision and Pattern Recognition*, 2009, pp. 1658–1665.

[145] L. Sun, S. Ji, and J. Ye, "Hypergraph spectral learning for multi-label classification," in *Proc. of ACM SIGKDD Int. Conference on Knowledge Discovery and Data Mining*. ACM, 2008, pp. 668–676.

[146] S. Godbole and S. Sarawagi, "Discriminative methods for multi-labeled classification," in *Advances in Knowledge Discovery and Data Mining*. Springer, 2004, pp. 22–30.

[147] N. Alaydie, C. K. Reddy, and F. Fotouhi, "Exploiting label dependency for hierarchical multi-label classification," in *Advances in Knowledge Discovery and Data Mining*. Springer, 2012, pp. 294–305.

[148] A. Veloso, W. Meira Jr, M. Gonçalves, and M. Zaki, "Multi-label lazy associative classification," in *Knowledge Discovery in Databases*. Springer, 2007, pp. 605–612.

[149] J. Read, L. Martino, and D. Luengo, "Efficient Monte Carlo optimization for multi-dimensional classifier chains," *arXiv preprint arXiv:1211.2190*, 2012.

[150] K. Weinberger, A. Dasgupta, J. Langford, A. Smola, and J. Attenberg, "Feature hashing for large scale multitask learning," in *Proc. of Int. Conference on Machine Learning*, 2009, pp. 1113–1120.

[151] R. A. Amar, D. R. Dooly, S. A. Goldman, and Q. Zhang, "Multiple-instance learning of real-valued data," in *Proc. of Int. Conference on Machine Learning*, 2001, pp. 3–10.

[152] M. Palatucci, D. Pomerleau, G. Hinton, and T. Mitchell, "Zero-shot learning with semantic output codes," in *Proc. of Neural Information Processing Systems*, 2009, pp. 1410–1418.

[153] C.-J. Hsieh, K.-W. Chang, C.-J. Lin, S. Keerthi, and S. Sundararajan, "A dual coordinate descent method for large-scale linear svm," in *Proc. of Int. Conference on Machine Learning*, 2008, pp. 408–415.

[154] M. J. Huiskes and M. S. Lew, "The MIR Flickr retrieval evaluation," in *Proc. of ACM Int. Conf. on Multimedia Information Retrieval*, 2008, pp. 39–43.

[155] M. Guillaumin, J. Verbeek, and C. Schmid, "Multimodal semi-supervised learning for image classification," in *Proc. of IEEE Conference on Computer Vision and Pattern Recognition*, 2010, pp. 902–909.

[156] R.-E. Fan, P.-H. Chen, and C.-J. Lin, "Working set selection using second order information for training svm," *Journal of Machine Learning Research*, vol. 6, pp. 1889–1918, 2005.

[157] J. C. Platt, "Probabilistic outputs for support vector machines and comparisons to regularized likelihood methods," *Advances in Large Margin Classifiers*, vol. 10, no. 3, pp. 61–74, 1999.

[158] M. Marszalek and C. Schmid, "Semantic hierarchies for visual object recognition," in *Proc. of IEEE Conference on Computer Vision and Pattern Recognition*. IEEE, 2007, pp. 1–7.

[159] N. Nguyen and R. Caruana, "Classification with partial labels," in *Proc. of ACM SIGKDD Int. Conference on Knowledge Discovery and Data Mining*, 2008, pp. 551–559.

[160] R. Jin and Z. Ghahramani, "Learning with multiple labels," in *Proc. of Neural Information Processing Systems*, 2002, pp. 897–904.

[161] A. Pentland, "Expectation maximization for weakly labeled data," in *Proc. of Int. Conference on Machine Learning*, 2001, pp. 218–225.

[162] K. Crammer and Y. Singer, "On the learnability and design of output codes for multiclass problems," *Machine Learning*, vol. 47, no. 2, pp. 201–233, 2002.

[163] M. Yuan and Y. Lin, "Model selection and estimation in regression with grouped variables," *J. Royal. Statist. Soc B.*, vol. 68, no. 1, pp. 49–67, 2006.

[164] J. Fan, Y. Shen, N. Zhou, and Y. Gao, "Harvesting large-scale weakly-tagged image databases from the web," in *Proc. of IEEE Conference on Computer Vision and Pattern Recognition*, 2010, pp. 802–809.

[165] M. Szummer, "Learning from partially labeled data," Ph.D. dissertation, Massachusetts Institute of Technology, 2002.

[166] S. M. Kakade, S. Shalev-Shwartz, and A. Tewari, "Efficient bandit algorithms for online multiclass prediction," in *Proc. of Int. Conference on Machine Learning*, 2008, pp. 440–447.

[167] S. Wang, R. Jin, and H. Valizadegan, "A potential-based framework for online multi-class learning with partial feedback," in *Proc. of Int. Conference on Artificial Intelligence and Statistics*, 2010, pp. 900–907.

[168] F. Alizadeh and D. Goldfarb, "Second-order cone programming," *Mathematical Programming*, vol. 95, no. 1, pp. 3–51, 2003.

[169] M. Petrovskiy, "Paired comparisons method for solving multi-label learning problem," in *Proc. of Conference of Hybrid Intelligent Systems*, 2006, pp. 42–42.

[170] L. Sun, S. Ji, and J. Ye, "Canonical correlation analysis for multilabel classification: a least-squares formulation, extensions, and analysis," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 33, no. 1, pp. 194–200, 2011.

[171] O. Yakhnenko and V. Honavar, "Multiple label prediction for image annotation with multiple kernel correlation models," in *IEEE Computer Vision and Pattern Recognition Workshops*, 2009.

[172] W. Zhang, X. Xue, J. Fan, X. Huang, B. Wu, and M. Liu, "Multi-kernel multi-label learning with max-margin concept network," in *Proc. of Int. Joint Conference on Artificial Intelligence*, 2011, pp. 1615–1620.

[173] L.-J. Li, H. Su, E. P. Xing, and F.-F. Li, "Object bank: A high-level image representation for scene classification & semantic feature sparsification." in *Proc. of Neural Information Processing Systems*, 2010, p. 5.

[174] L. G. Roberts, "Machine perception of three-dimensional solids," Ph.D. dissertation, Massachusetts Institute of Technology, 2007.

[175] T. O. Binford, "Survey of model-based image analysis systems," *The International Journal of Robotics Research*, vol. 1, no. 1, pp. 18–64, 1982.

[176] R. T. Chin and C. R. Dyer, "Model-based recognition in robot vision," *ACM Computing Surveys*, vol. 18, no. 1, pp. 67–108, 1986.

[177] R. Bergevin and M. Levine, "Generic object recognition: building and matching coarse descriptions from line drawings," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 15, no. 1, pp. 19 –36, 1993.

[178] F. Stein and G. Medioni, "Structural indexing: efficient 2d object recognition," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 14, no. 12, pp. 1198 –1204, 1992.

[179] E. Saber, A. M. Tekalp, R. Eschbach, and K. Knox, "Automatic image annotation using adaptive color classification," *Graphical Models and Image Processing*, vol. 58, no. 2, pp. 115 – 126, 1996.

[180] M. J. Swain and D. H. Ballard, "Color indexing," *Int. Journal of Compututer Vision*, vol. 7, no. 1, pp. 11–32, 1991.

[181] J. Mao and A. K. Jain, "Texture classification and segmentation using multiresolution simultaneous autoregressive models," *Pattern Recognition*, vol. 25, no. 2, pp. 173 – 188, 1992.

[182] B. S. Manjunath and W.-Y. Ma, "Texture features for browsing and retrieval of image data," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 18, no. 8, pp. 837 –842, 1996.

[183] P. Duygulu, K. Barnard, N. de Freitas, and D. Forsyth, "Object recognition as machine translation: Learning a lexicon for a fixed image vocabulary," in *Computer Vision–ECCV*. Springer, 2002, pp. 97–112.

[184] J. Jeon, V. Lavrenko, and R. Manmatha, "Automatic image annotation and retrieval using crossmedia relevance models," in *Proc. of Int. ACM SIGIR conference on Research and development in informaion retrieval*, 2003, pp. 119–126.

[185] F. Monay and D. Gatica-Perez, "On image auto-annotation with latent space models," in *Proc. of ACM Int. Conference on Multimedia*, 2003, pp. 275–278.

[186] J.-Y. Pan, H.-J. Yang, P. Duygulu, and C. Faloutsos, "Automatic image captioning," in *Proc. of IEEE Int. Conference on Multimedia and Expo*, 2004, pp. 1987–1990.

[187] C. Schmid and R. Mohr, "Matching by local invariants," INRIA, Tech. Rep. RR-2644, 1995.

[188] C. Schmid, R. Mohr, and C. Bauckhage, "Comparing and evaluating interest points," in *Proc. of Int. Conference on Computer Vision*, 1998, pp. 230 –235.

[189] D. G. Lowe, "Distinctive image features from scale-invariant keypoints," *International Journal of Computer Vision*, vol. 60, pp. 91–110, 2004.

[190] G. Csurka, C. R. Dance, L. Fan, J. Willamowski, and C. Bray, "Visual categorization with bags of keypoints," in *ECCV Workshop on Statistical Learning in Computer Vision*, 2004.

[191] C. Harris and M. Stephens, "A combined corner and edge detector," in *Proc. of Alvey Vision Conference*, 1988, pp. 50–50.

[192] N. Dalal and B. Triggs, "Histograms of oriented gradients for human detection," in *Proc. of IEEE Conference on Computer Vision and Pattern Recognition*, 2005, pp. 886–893.

[193] F. Perronnin, J. Snchez, and T. Mensink, "Improving the fisher kernel for large-scale image classification," in *Computer Vision–ECCV*. Springer, 2010, pp. 143–156.

[194] B. C. Russell, A. Torralba, K. P. Murphy, and W. T. Freeman, "LabelMe: a database and web-based tool for image annotation," *International Journal of Computer Vision*, vol. 77, no. 1-3, pp. 157–173, 2008.

[195] A. Farhadi, I. Endres, D. Hoiem, and D. Forsyth, "Describing objects by their attributes," in *Proc. of IEEE Conference on Computer Vision and Pattern Recognition*, 2009, pp. 1778–1785.

[196] LSVRC Challenge, http://www.image-net.org/challenges/LSVRC/2013/.

[197] S. Nowak and M. J. Huiskes, "New strategies for image annotation: Overview of the photo annotation task at ImageCLEF 2010," in *CLEF (Notebook Papers/LABs/Workshops)*, vol. 1, no. 3, 2010, p. 4.

[198] L. von Ahn and L. Dabbish, "Labeling images with a computer game," in *Proc. of the Conference on Human Factors in Computing Systems*, 2004, pp. 319–326.

[199] A. Krizhevsky, I. Sutskever, and G. Hinton, "ImageNet classification with deep convolutional neural networks," in *Proc. of Neural Information Processing Systems*, 2012, pp. 1106–1114.

[200] N. Cesa-Bianchi and G. Lugosi, *Prediction, Learning, and Games*. Cambridge University Press, 2006.