Open-ended Learning of Visual and Multi-modal Patterns

THÈSE N^{o} 5233 (2011)

PRÉSENTÉE LE 14 DÉCEMBRE, 2011
À LA FACULTÉ SCIENCES ET TECHNIQUES DE L'INGÉNIEUR
LABORATOIRE DE L'IDIAP
PROGRAMME DOCTORAL EN INFORMATIQUE, COMMUNICATIONS ET INFORMATION

ÉCOLE POLYTECHNIQUE FÉDÉRALE DE LAUSANNE

POUR L'OBTENTION DU GRADE DE DOCTEUR ÈS SCIENCES

PAR

Jie Luo

acceptée sur proposition du jury:

Prof. Jean-Philippe Thiran, président du jury
Prof. Hervé Bourlard, directeur de thèse
Dr. Barbara Caputo, co-directeur de thèse
Dr. Samy Bengio, rapporteur
Prof. Aude Billard, rapporteur
Prof. Bastian Leibe, rapporteur

ÉCOLE POLYTECHNIQUE
FÉDÉRALE DE LAUSANNE

Abstract

A common trend in machine learning and pattern classification research is the exploitation of massive amounts of information in order to achieve an increase in performance. In particular, learning from huge collections of data obtained from the web, and using multiple features generated from different sources, have led to significantly boost of performance on problems that have been considered very hard for several years. In this thesis, we present two ways of using these information to build learning systems with robust performance and some degrees of autonomy. These ways are Cue Integration and Cue Exploitation, and constitute the two building blocks of this thesis.

In the first block, we introduce several algorithms to answer the research question on how to integrate optimally multiple features. We first present a simple online learning framework which is a wrapper algorithm based on the high-level integration approach in the cue integration literature. It can be implemented with existing online learning algorithms, and preserves the theoretical properties of the algorithms being used. We then extend the Multiple Kernel Learning (MKL) framework, where each feature is converted into a kernel and the system learns the cue integration classifier by solving a joint optimization problem. To make the problem practical, We have designed two new regularization functions making it possible to optimize the problem efficiently. This results in the first online method for MKL. We also show two algorithms to solve the batch problem of MKL. Both of them have a guaranteed convergence rate. These approaches achieve state-of-the-art performance on several standard benchmark datasets, and are order of magnitude faster than other MKL solvers.

In the second block, We present two examples on how to exploit information between different sources, in order to reduce the effort of labeling a large amount of training data. The first example is an algorithm to learn from partially annotated data, where each data point is tagged with a few possible labels. We show that it is possible to train a face classification system from data gathered from Internet, without any human labeling, but generating in an automatic way possible lists of labels from the captions of the images. Another example is under the transfer learning setting. The system uses existing models from potentially correlated tasks as experts, and transfers their outputs over the new incoming samples, of a new learning task where very few labeled data are available, to boost the performance.

Keywords: machine learning, multiple cue integration, visual recognition, online learning, multiple kernel learning, weakly supervised learning, transfer learning

Résumé

Dans les domaines de l'apprentissage automatique et de la classification de formes, une tendance courante est l'exploitation d'énormes quantités d'informations afin de parvenir à une amélioration des performances. En particulier, l'apprentissage à partir d'immenses collections de données collectées sur le Web, ainsi que l'utilisation de plusieurs primitives générées à partir de différentes sources, ont permis d'améliorer de façon significative les résultats de problèmes longtemps considérés comme très difficiles. Dans cette thèse, nous présentons deux façons d'utiliser ces informations pour construire des systèmes d'apprentissage aux performances robustes et avec un certain degré d'autonomie. Ces moyens sont l'intégration de primitives et l'exploitation de primitives, et constituent les deux composantes fondamentales de cette thèse.

Dans la première composante, nous présentons plusieurs algorithmes pour répondre à la question portant sur la façon optimale d'intégrer plusieurs primitives. Nous présentons tout d'abord un cadre d'apprentissage en ligne basé sur l'approche d'intégration de haut niveau dans la littérature d'intégration de primitives. Ce cadre peut être mis en œuvre avec les algorithmes d'apprentissage en ligne existants, et préserve les propriétés théoriques des algorithmes utilisés. Nous avons ensuite étendu le cadre de l'apprentissage par noyaux multiples (MKL), où chaque primitive est convertie en un noyau et où le système apprend un classifieur d'intégration de primitives en résolvant un problème d'optimisation conjointe. Pour rendre le problème réalisable en pratique, nous avons conçu deux nouvelles fonctions de régularisation permettant d'optimiser le problème efficacement. Cela conduisit à la première méthode en ligne pour MKL. Nous montrons également deux algorithmes pour résoudre le problème séquentiel de MKL. Les deux ont une vitesse de convergence garantie. Ces approches atteignent des performances de pointes sur plusieurs jeux de données de référence, et sont un ordre de grandeur plus rapide que les autres trouveurs de solution MKL.

Dans la seconde composante, nous présentons deux exemples sur la façon d'exploiter l'information entre différentes sources, afin de réduire l'effort demandé par l'étiquetage d'une grande quantité de données d'entraînement. Le premier exemple est un algorithme d'apprentissage à partir de données partiellement annotées, où chaque donnée est marquée avec plusieurs étiquettes possibles. Nous montrons qu'il est possible de former un système de classification de visages à partir de données recueillies sur Internet, sans aucun étiquetage humain, mais générant de manière automatique les listes d'étiquettes possibles à partir de la légende des images. Un autre exemple est décrit dans le cadre du transfert de connaissances. Le système utilise, en tant qu'experts, des modèles existants de tâches potentiellement corrélées, et transfert leurs réponses sur les nouveaux échantillons entrants d'une nouvelle tâche d'apprentissage où très peu de données étiquetées sont disponibles, et ce, en vue d'améliorer les performances.

Mots-clés : apprentissage automatique, intégration de plusieurs primitives, reconaissance visuelle, apprentissage en ligne, apprentissage par noyaux multiples, apprentissage faiblement supervisé, transfert de connaissances

Sommario

La ricerca in riconoscimento di pattern e apprendimento artificiale negli ultimi anni si é concentrata su come utilizzare al meglio vaste quantitá di informazione per riuscire a raggiungere migliori risultati di classificazione. In particolare, il problema di come imparare modelli da enormi raccolte di dati ottenute dal Web, e il problema complementare di come combinare in maniera efficiente, efficace ed ottimale descrittori calcolati da diverse sorgenti, é stato affrontato portando a miglioramenti considerevoli dei risultati in problemi che per anni erano risultati ostici alla comunitá. In questa tesi, presentiamo due modalitá per usare questo tipo di informazione per poi costruire sistemi artificiali capaci di imparare ed ottenere risultati robusti in maniera autonoma. Le due metodologie che consideriamo sono la combinazione di caratteristiche da modalitá diverse, e il loro reciproco rafforzarsi in vari scenari. Questi due filoni di ricerca costituiscono le due parti fondamentali di questa tesi.

Nella prima parte, introduciamo vari algoritmi per rispondere alla domanda concettuale su come combinare in maniera ottimale diverse caratteristiche percettive. Per prima cosa presentiamo uno schema basilare per l'apprendimento continuo che consiste in un algoritmo a due livelli, con classificatori a cascata. Lo schema é tale da poter essere implementato combinando insieme algoritmi di apprendimento continuo giá presentati nella letteratura. Proponiamo una analisi teorica che mostra come il nostro schema preservi le proprietá degli algoritmi scelti per la specifica implementazione. Come secondo contributo proponiamo una estensione dello schema di apprendimento a Kernel multipli, dove ogni caratteristica estratta viene trasformata in un Kernel, e il sistema risultante impara come combinare le diverse caratteristiche risolvendo un problema di ottimizzazione congiunto. Per rendere il problema accessibile, abbiamo disegnato due nuove funzioni di regolarizzazione che rendono possibile ottimizzare l'algoritmo in maniera efficiente. Il risultato é il primo algoritmo online di apprendimento su Kernels multipli. Inoltre introduciamo due algoritmi per risolvere il problema dell'addestramento di questa famiglia di algoritmi in modalitá classica. Entrambi hanno un tasso di convergenza garantito. Questi approcci hanno ottenuto risultati competitivi con i migliori presentati nella letteratura internazionale su diverse collezioni di dati usate per valutazioni comparative pubbliche, e sono piú rapidi nell'apprendimento di altri algoritmi noti di un ordine di grandezza.

Nella seconda parte della tesi, presentiamo due esempi di come si puó trarre vantaggio dall'utilizzare in contemporanea differenti sorgenti di informazione nel apprendere un modello, con l'obiettivo di ridurre la quantitá di dati annotati necessari per apprendere modelli. Il primo esempio é un algoritmo capace di apprendere modelli da dati annotati solo parzialmente, dove ciascun dato é associato a piú di una possibile etichetta. Noi mostriamo che é possibile addestrare un algoritmo per l'apprendimento di facce usando dati collezionati dal Web senza nessun tipo di annotazione da parte di umani, generando automaticamente una lista delle annotazioni possibili usando le didascalie scritte associate a queste foto. Un altro esempio che proponiamo riguarda il trasferimento automatico di conoscenza pregressa. Il nostro algoritmo usa modelli esistenti da categorie in qualche modo correlate al nuovo problema come esperti, e trasferisce le loro confidenze per l'apprendimento di una nuova classe da dati impoveriti.

parole chiave: apprendimento automatizzato, integrazione di informazioni multiple, riconoscimento visivo, apprendimento continuo, apprendimento di Kernels multipli, apprendimento debolmente supervisionato, trasferimento di conoscenza pregressa.

To my family

Acknowledgement

The completion of this thesis was a great adventure. This last step in my life-long journey as student would not have been possible without many people who have supported, encouraged and inspired me over the past five years. I have been lucky to work with and surround by many great people who contributed to this thesis in one way or another. First and foremost, I would like to express my gratitude to my main advisor Barbara Caputo, for introducing me to research, and for her supervision, support, kindness and friendship. Interactions with you are always enthusiastic and fruitful. Thank you for giving me the freedom to explore my ideas independently, for allowing me to refresh my mind by visiting other research groups and performing internships while I am frustrated and procrastinating on my thesis, and last but certainly not the least for your patience in helping me improving my technical writing. I would also like to thank my thesis director Prof. Hervé Bourlard and Idiap Research Institute for providing me the great research facilities and inspiring environments. I am grateful to my thesis juries – Prof. Jean-Philippe Thiran (président), Dr. Samy Bengio, Prof. Aude Billard, Prof. Bastian Leibe for kindly agreeing to read my thesis. There are two other colleagues whose names deserve special mention although they did not appear in the list of my thesis committees. A significant amount of the work reported in this thesis was influenced by discussion with Francesco Orabona and Vittorio Ferrari. Many thanks go to Francesco for being a good office neighbor, for teaching me all the technical details with great patience and persistence, and for the brainstormings on new research ideas and life in general. I will not forget our NIPS, CVPR and ICML journeys plus the exciting post-conference adventures. Many thanks go to Vittorio for supervising me during my visit at at ETHZ. The collaborations afterward have been very fruitful. I have learned a lot during my interaction with you. Thank you for sharing with me your research vision, and for keeping challenging me on my ideas which shaped the works better and better. Besides the people mentioned above, the thesis has been made possible in collaboration with several other colleagues. My thanks go to Prof. Nicolò Cesa-Bianchi, Marco Fornoni, and Tatiana Tommasi.

It was a pleasure to share doctoral studies and life with other colleagues from Idiap. Work and life in Idiap would not have been the same without my office mate: Anindya, Charles, Cosmin, Elisa, Francesco, Guillaume, Hugo, Jagan, Joseph, Marco, Nik, Leo, Ta-

tiana, Xavier; outdoor sport buddies: Carl, Deepu, Ganga, Guillermo, Ferran, Laurent, Nicolas, Petr, Thomas; and many others including Arjan, Bastien, Bogdan, Chris, David, Dinesh, Dong, Eileen, Hamed, Hari, Hayley, Hui, Jian, Joan, John, Jean-Marc, JFP, Kate, Le, Majid, Mathew, Mert, Paco, Phil, Radu, Ricardo, Serena, Sileye, Stephanie, Tamara, Venky, Wanjun, Weifeng, Weina, Xiangxin. There are many more, former and current Idiapers that have influenced my work and life in positive ways.

In the early period of my PhD, I also have the opportunity to collaborate with Prof. Jörn Anemüller, Jörg-Hendrik Bach, Prof. Hynek Hermansky, Muhammad Muneeb Ullah, Anrzej Pronobis, Prof. Patric Jensfelt, Prof. Daphana Weinshall, Alon Zweig, and all the great people involved in the DIRAC and CoSy project. I learned a lot from these collaborations. In the same period, I also have the great opportunities to visit the CBCL at MIT and BIWI at ETHZ. Thanks Prof. Tommy Poggio for hosting me in CBCL. I have met many talent people there, including Tommy's students Huei-han, Jake, Sharat and Tony. Thanks Prof. Luc Van Gool for hosting me in BIWI. I had a great summer in Zurich, and met many friends there. Thanks to Alessandro, Andreas, Angela, Bogdan, Dagan, Gabriele, Gang, Marcin, Matthieu, Peter, Stefano, and Thomas for your kindness everytime I visited BIWI.

Finally, thanks to all the Chinese friends lived and studied in Lausanne, Martigny and Zurich, for adding life to Switzerland. Mom, dad, grandma and grandpa, I can't express in words how thankful I am to all of you. I will not be able to finish my thesis without your support,

Contents

1 Introduction **1**

1.1 Learning from Multiple Cues: Advantages and Challenges 1

1.2 Organization and Contribution of this Thesis 3

1.3 Preliminaries 5

I Cues Integration **11**

2 Background and Related Works **13**

2.1 Motivation for cue integration 13

2.2 Review of Cue Integration Techniques 14

2.2.1 Fusion Types 15

2.2.2 Comparison and Learning 17

2.3 Motivation for Online Learning 19

3 Two-Layers Approach: An Online Learning Framework **21**

3.1 Introducing the OMCL Framework 22

3.2 OMCL-BM Algorithm 23

3.2.1 Projectron++ Algorithm 24

3.2.2 Passive-Aggressive Algorithm 25

3.3 Online to Batch Conversion 26

3.4 Experiments and Results 27

3.4.1 First Scenario: Place Recognition 28

3.4.2 Second Scenario: Object Categorization 30

3.5 Summary 32

4 One-Layer Approach: MKL with Online-Batch Optimization 35

4.1 Introduction 36

4.2 The l_p Norm MKL Formulation 38

4.3 Online MKL: the OM-2 algorithm 39

4.3.1 FTRL Framework 39

4.3.2 Algorithm and Analysis 40

4.3.3 Implementation 43

4.3.4 Experiments 43

4.4 Batch MKL: the OBSCURE algorithm 48

4.4.1 Algorithm 49

4.4.2 Efficient Implementation 50

4.4.3 Analysis of the OBSCURE Batch Stage 51

4.4.4 Analysis of the OBSCURE Online Stage 53

4.4.5 Experiments 54

4.5 Sparse MKL: the UFO algorithm 58

4.5.1 Sparse MKL Formulation 58

4.5.2 Algorithm 59

4.5.3 Analysis 60

4.5.4 Experiments 61

4.6 Conclusion 63

II From Cues Integration to Cues Exploitation 65

5 Weakly Supervised Learning using Co-Occurring Cue(s) 67

5.1 Introduction 68

5.2 Related Work 69

5.3 Problem Definition 71

5.4 Learning from the Candidate Labeling Sets and the MMS Algorithm 74

5.4.1 Prediction functions . . . 75
5.4.2 Ambiguous loss functions . . . 76
5.4.3 Maximum Margin Set (MMS) . . . 78
5.4.4 Optimizing the MMS problem with CCCP . . . 78
5.4.5 Solving the relaxed MMS optimization problem using the Pegasos framework . . . 80
5.5 Experiments on artificial data . . . 81
5.6 Real case 1: Who is in the picture? . . . 83
5.6.1 Modeling . . . 84
5.6.2 Inference . . . 85
5.6.3 Experiments . . . 86
5.7 Real case 2: Who is doing what? . . . 87
5.7.1 Modeling . . . 88
5.7.2 Inference . . . 89
5.7.3 Experiments . . . 89
5.8 Conclusion . . . 91

6 Knowledge Transfer Across Different Cues 93
6.1 Introduction . . . 94
6.2 Problem Definition . . . 95
6.3 Multiple Kernel Transfer Learning . . . 97
6.4 Comparison with Existing Methods . . . 99
6.5 Experiments . . . 100
6.5.1 Binary Transfer Learning . . . 101
6.5.2 Multiclass Transfer Learning . . . 102
6.6 Conclusions . . . 105

7 Summary and Future Direction 107
7.1 Discussion . . . 107
7.2 Future Directions . . . 108
7.2.1 Multiple Kernel Active & Semi-supervised Learning . . . 108
7.2.2 Approximate Inference for Training MMS . . . 108

A Datasets and Features **111**

B Miscellaneous Proofs **115**

Curriculum Vitae **137**

List of Figures

2.1 Example images of a horse, a zebra, a car and a motorbike. 14

3.1 Average online training error rate and classification error rate on the test set of the IDOL2 dataset. 28

3.2 Average size of support set for different algorithms on the IDOL2 dataset as a function of the number of training samples. 29

3.3 Visualization of the average normalized weights obtained by OMCL-BM on the IDOL2 dataset at the last training round for combining the confidence outputs of Projectron++. 29

3.4 Average online training error rate and classification error rate for classifying never seen objects on the ETH-80 dataset. 30

3.5 Average size of support set for different algorithms on the ETH80 dataset. 31

3.6 Visualization of the average normalized weights obtained by OMCL-BM at the last training round for combining the confidence outputs of Projectron++. 31

4.1 Experiment on the multiclass synthetic data which illustrates the performance of different types of loss functions. 45

4.2 Behaviors of OM-2 w.r.t. different values of p 45

4.3 Performance comparison of OM-2 and other baselines on the Caltech-101 dataset 46

4.4 Performance comparison of OM-2 and other baselines on the IDOL2 dataset dataset . . . 47

4.5 Comparison of running time performance on the Oxford flowers dataset between OBSCURE, Pegasos and SILP. 56

4.6 Behaviors of OBSCURE on the Caltech-101 w.r.t. different number of kernels randomly sampled from the 39 kernels. 56
4.7 Performance comparison on the Caltech-101 dataset using different cue integration methods. 57
4.8 The generalization performance of OBSCURE on the MNIST dataset over different sizes of training samples. 57
4.9 Performance of UFO-MKL and other baselines on the Caltech-101 dataset. 62
4.10 Running time performance of UFO-MKL and other baseline approaches w.r.t. different number of kernels and varying values of α. 63

5.1 Two examples of image-captions pairs, and one example of our candidate labeling set notation. 70
5.2 Example of an image-caption pair containing spatial indicators. 71
5.3 Classification performance of different weakly supervised learning and supervised learning algorithms on four artificial datasets. 83
5.4 An example image with its associated caption, and its candidate labeling set. 89
5.5 Candidate labeling vectors generation for the new item in Figure 5.1 (left) 89
5.6 Name and verb assignment and recognition accuracy on 1600 training images from the Idiap/ETHZ face+pose database 91

6.1 A graphical representation of how to use the outputs from the prior models as auxiliary features when computing the score of a new class. 95
6.2 Results obtained on the object category detection scenario, when learning 1 out of 30 categories with the rest categories as prior models. 103
6.3 Results obtained on the multiclass object categorization scenario 104

List of Tables

2.1 Comparison of different cue integration techniques. 18

3.1 Place recognition error rate using different cues after the last training round. 29

3.2 Categorization error rate for different objects using different cues after finishing the last training round. 31

5.1 Protocol I -Overall name assignment accuracy on the training set 87

5.2 Protocol I - Overall face recognition accuracy on the test set 87

5.3 Protocol II -Overall name assignment accuracy on the training set 88

5.4 Protocol II - Overall face recognition accuracy on the test set 88

Chapter 1

Introduction

This introduction presents an overview of learning from multimodal patterns (Section 1.1) and the contribution of this thesis (Section 1.2). The main motivation presented here is elaborated in more details in the following chapters. At the end of this chapter, we also introduces the notations and the necessary mathematical tools used in this thesis (Section 1.3).

1.1 Learning from Multiple Cues: Advantages and Challenges

A common trend in machine learning and pattern classification research is the exploitation of massive amounts of information to achieve an increase in classification performance. In particular, learning using multiple discriminative features and multiple source inputs, and from huge collections of data obtained from the web, have led to significant boost of performance on problems considered very hard for several years.

In recent years there has been a lot of interest in designing principled classification algorithms over multiple cues, based on the intuitive notion that using more features should lead to better performance. There is plenty of evidence showing that integrating multiple cue inputs, especially from different sensory modalities, can greatly enhance the ability of animals and humans to cope economically and flexibly with complex and ever-changing environments. For instance, we do recognize people on the basis of their visual appearance and their voice. Linen can be easily recognized because of its distinctive textual visual and tactile properties; and so forth. Many researchers started following this route, and used multiple cues and multiple modalities (*e.g.,* vision, sound and touch) in several applications, such as image categorization systems, biometric identity verification systems and robots, with success. While research on this topic is gaining momentum, how to combine cues so to achieve optimal performance is still an open issue. Furthermore, algorithms are interesting in the real world only if they are scalable. Since multiple cue features expand the input spaces, computational and memory efficiency become an essential requirement for multiple cue integration algorithms. Previous cue integration algorithms (Wolpert, 1992; Nilsback and Caputo, 2004; Lanckriet *et al.*, 2004a; Bach *et al.*, 2004; Bosch *et al.*, 2007; Gehler and Nowozin, 2009b) have not paid much attentions to these aspects during their design.

Another important aspect which boosts performance is due to the enhancement of computer hardware. As of today, it is possible to process efficiently a huge amount of data that was not even possible to store a few years ago. With the avalanche of multimedia data on the web, it has become easier and cheaper to obtain data for the development of statistical learning systems. We can thus expect that the classification performance to be further boosted with an amount of data approximating even only a small fraction of what is available on the web. On the other hand, although collecting the data is cheap, due to the noisy nature of the Web, obtaining clean annotations of these data is still expensive. Therefore, how to reduce supervision becomes an important question. Ideally, learning should require as little manual supervision as possible. Multi-cue and multi-modality inputs provide us one possible solution to it, as one cue may contain useful information for guiding the learning on the other cue.

Our aim is to use a large amount of information from multiple cues to build learning systems with robust performance and some degrees of autonomy. In this thesis, we will address specifically the following research questions:

- How to integrate multiple cues optimally, so to achieve *robust* performance?
- How to learn continuously from experience, so to achieve *adaptability*?
- How to exploit the relationship between different cues and transfer knowledge across them, so to achieve *autonomy*?

We propose to learn the optimal perceptual integration from data using statistical learning techniques. We will consider learning as an ongoing continuous process trying to exploit different inputs so to obtain artificial systems able to transfer knowledge from one sensory experience to the others. This would imply self-supervision across modalities and eventually the emergence of cognitive loops, where a cognitive loop is the mutual transfer of higher knowledge between different system components and algorithms, which overcomes the weaknesses of any single system component and algorithm by combining them.

1.2 Organization and Contribution of this Thesis

According to the proposed research questions, this dissertation can be divided into two building blocks, *Cues Integration*, and *Cues Exploitation*. In the first block, I will present several algorithms to answer the research question on how to integrate optimally multiple features. In the second block, I will go beyond cues integration, and discuss about possible ways to do cues exploitation, that is, make use of informations extracted from certain cue(s) in order to facilitate the learning of the other cues with less supervision. As opposed to the first block where a general framework exists, cues exploitation problems are usually task specific. How to make use of the data relies on exploiting the domain knowledge and the structure of the data. Here I will present two particular examples, one on learning visual recognition models using images and their accompanying captions; the other is to transfer priors in multiclass visual categorization tasks, in order to boost performance with few labelled training samples.

The main contribution of this thesis are:

- **Chapter 3**: A online learning framework which learns one independent classifier for each separate cue and the weights to combine the outputs of these cue classifiers together in an online fashion. The proposed framework is a wrapper algorithm based on the high-level integration technique in the cue integration literature. It can be implemented with existing online learning algorithms, and it preserves the theoretical properties of the algorithms being used. The algorithms designed using this framework is very efficient. Experiments conducted on two real-world datasets show the algorithm developed using our framework outperforms the case when we use each cue alone. This chapter is partly based on Jie *et al.* (2009a).

- **Chapter 4**: Two novel Multiple Kernel Learning (MKL) formulations and several theoretically motivated online and batch algorithms for solving the proposed formulations. To the best of our knowledge, our online MKL algorithm, called OM-2, is the first online version of the MKL formulation. All the algorithms use the stochastic gradient descent method and the mirror descent method to solve the corresponding MKL formulations, which are very efficient and orders of magnitude faster compare to other state-of-art MKL solvers. We prove theoretical properties of the proposed algorithms, including guaranteed convergence rate for the batch algorithms, which is also new in the literature. The proposed algorithms achieve state-of-art performance on standard benchmark databases. This chapter is mainly based on Jie *et al.* (2010); Orabona *et al.* (2010); Orabona and Jie (2011).

- **Chapter 5**: A new type of weakly supervised learning problem where a bag of instances has a set of several possible labeling vectors associated with it, and among them only one is fully correct. Each labeling vector consists of labels for each corresponding instance in the bag. We also propose a discriminative learning algorithm for solving the proposed problem. Our learning formulation uses a tighter convex relaxation compared to the previous formulation of a similar problem (Cour *et al.*, 2009), and it also results in a better performance. This new framework allows to incorporate in a principled way different types of constrains for labeling instances in the bag. The problem of learning visual classifiers from unlabeled images by exploiting their accompanying captions can be casted within this setting. Experiments show that our algorithm can learn the concepts effectively using the weakly supervised information extracted from the caption (Chapter 5). This chapter is partly based on Jie *et al.* (2009b); Jie and Orabona (2010); Jie *et al.* (2011a).
- **Chapter 6**: A multiclass transfer learning algorithm that allows to take advantage from priors built over different features and with different learning methods than the one used for learning the new correlated task. The system uses existing models from potentially correlated tasks as experts, and transfers their outputs over the new incoming samples, of a new learning task where very few labelled data are available, as additional information to boost the performance. The learning process is defined as solving an optimization problem which considers both from where and how much to transfer using a principled multiclass formulation (Chapter 6). The new transfer learning algorithm improves over many previous transfer learning algorithms (discussed in Chapter 6) which make strong assumptions on the way the prior models were constructed and the type of features used. It is also one of the first few algorithms addressing the transfer learning problem in the multiclass setting in the computer vision community. This chapter is based on Jie *et al.* (2011b).
- We have released the MATLAB codes of all the algorithms presented in this thesis, and they can be downloaded freely from `http://dogma.sourceforge.net/` (Orabona, 2009).

In this thesis, our experiments will mainly focus on visual patterns. However, all the algorithms we propose are general and could be used in other problems. A brief introduction of the databases as well as their features descriptors can be founded in Chapter A in the appendix. The proofs of all the theorems in the thesis are given in Chapter B in the appendix.

1.3 Preliminaries

In this section we introduce formally the notations as well as the necessary mathematical tools and concepts for the thesis.

Notations. We indicate scalars with lower case letters (*e.g.,* x and λ), vectors and matrices with bold letters (*e.g.,* $\boldsymbol{x}$ and $\boldsymbol{w}$), and sets with calligraphic font (*e.g.,* $\mathcal{X}$). We also introduce two notations that will help us to synthetize the following formulas. We indicate by $[w^j]_1^F := \left[w^1, w^2, \cdots, w^F\right]$, and with a bar, e.g. $\bar{\boldsymbol{w}}$, the vector formed by the concatenation of the F vectors $\boldsymbol{w}^j$, hence $\bar{\boldsymbol{w}} = [\boldsymbol{w}^j]_1^F = [\boldsymbol{w}^1, \boldsymbol{w}^2, \cdots, \boldsymbol{w}^F]$.

The set of real number is denoted by $\mathbb{R}$, and the set of non-negative number is denoted by $\mathbb{R}^+$.

Norms. A generic norm of a vector $\boldsymbol{w} \in \mathbb{X}$ is indicated by $\|\boldsymbol{w}\|$, where $\mathbb{X}$ is an Euclidean vector space, and its *dual norm* $\|\cdot\|_*$ is defined as $\|\boldsymbol{v}\|_* = \sup\{\boldsymbol{w} \cdot \boldsymbol{v} : \|\boldsymbol{w}\| \leq 1\}$.

For $\boldsymbol{w} \in \mathbb{R}^d$ and $p \geq 1$, we denote by $\|\boldsymbol{w}\|_p$ the p-norm of $\boldsymbol{w}$, *i.e.,* $\|\boldsymbol{w}\|_p = (\sum_{i=1}^d |w_i|^p)^{1/p}$. The dual norm of $\|\cdot\|_p$ is $\|\cdot\|_q$, where p and q satisfy $1/p + 1/q = 1$. In the following p and q will always satisfy this relation.

It is possible to define a $(2, p)$ *group norm* $\|\bar{\boldsymbol{w}}\|_{2,p}^2$ on $\bar{\boldsymbol{w}}$ as

$$\|\bar{\boldsymbol{w}}\|_{2,p} := \left\| \left[\|\boldsymbol{w}^1\|_2, \|\boldsymbol{w}^2\|_2, \cdots, \|\boldsymbol{w}^F\|_2\right] \right\|_p ,$$

that is the p-norm of the vector of F elements, formed by 2-norms of the vectors $\boldsymbol{w}^j$. The dual norm of $\|\cdot\|_{2,p}$ is $\|\cdot\|_{2,q}$ (Kakade *et al.*, 2009).

Convex analysis. Throughout this dissertation, we also make extensive use of some concepts from convex analysis. Here we summarize some of the notations and concepts that are needed in the following parts. For a more thorough introduction see for example Boyd and Vandenberghe (2004).

We consider closed convex functions $g : \mathbb{S} \to \mathbb{R}$, where in the following $\mathbb{S}$ will always denote a proper subset of $\mathbb{X}$[1]. We will indicate the inner product between two vectors of $\mathbb{X}$, $\boldsymbol{w}$ and $\boldsymbol{w}'$, as $\boldsymbol{w} \cdot \boldsymbol{w}'$. Given a convex function $g : \mathbb{S} \to \mathbb{R}$, its Fenchel conjugate $g^* : \mathbb{X} \to \mathbb{R}$ is defined as $g^*(\boldsymbol{u}) = \sup_{\boldsymbol{v} \in \mathbb{S}}(\boldsymbol{v} \cdot \boldsymbol{u} - g(\boldsymbol{v}))$. A vector $\boldsymbol{x}$ is a subgradient of a function g at $\boldsymbol{v}$, if $\forall \boldsymbol{u} \in \mathbb{S}, g(\boldsymbol{u}) - g(\boldsymbol{v}) \geq (\boldsymbol{u} - \boldsymbol{v}) \cdot \boldsymbol{x}$. The differential set of g at $\boldsymbol{v}$, indicated with $\partial g(\boldsymbol{v})$, is the set of all the subgradients of g at $\boldsymbol{v}$. If g is convex and differentiable at $\boldsymbol{v}$ then $\partial g(\boldsymbol{v})$ consists of a single vector which is the gradient of g at $\boldsymbol{v}$ and is denoted by $\nabla g(\boldsymbol{v})$.

A function $g : \mathbb{S} \to \mathbb{R}$ is said to be λ-strongly convex w.r.t. a norm $\|\cdot\|$ if for all $\boldsymbol{u}, \boldsymbol{v} \in \mathbb{S}$ and $\alpha \in (0,1)$, $g\left(\alpha \boldsymbol{u} + (1-\alpha)\boldsymbol{v}\right) \leq \alpha g(\boldsymbol{u}) + (1-\alpha) g(\boldsymbol{v}) - \frac{1}{2}\lambda\alpha(1-\alpha)\|\boldsymbol{u} - \boldsymbol{v}\|^2$. A function $g^* : \mathbb{S} \to \mathbb{R}$ is said to be λ-strongly smooth w.r.t a norm $\|\cdot\|_*$ if g^* is everywhere differentiable and if for all $\boldsymbol{u}, \boldsymbol{v} \in \mathbb{S}$ we have, $g^*\left(\boldsymbol{u} + \boldsymbol{v}\right) \leq g^*(\boldsymbol{u}) + \nabla g^*(\boldsymbol{u}) \cdot \boldsymbol{v} + \frac{1}{2}\lambda\|\boldsymbol{v}\|_*^2$. Note that a closed and convex function g is λ-strongly convex w.r.t. a norm $\|\cdot\|$ iff its Fenchel conjugate function g^* is $\frac{1}{\lambda}$ w.r.t. the corresponding dual norm $\|\cdot\|_*$ (Kakade *et al.*, 2009).

Classifiers. Let $\{\boldsymbol{x}_i, y_i\}_{i=1}^N$, where N is the number of samples, with $\boldsymbol{x}_i \in \mathbb{X}$ and $y_i \in \mathbb{Y}$, be the training set drawn from an unknown probability distribution. We want to learn a classification function $f : \mathbb{X} \to \mathbb{Y}$, which best predicts the label for any future samples drawn from the same distribution. This prediction of the classification function is performed based on a hypothesis, $s : \mathbb{X} \times \mathbb{Y} \to \mathbb{R}$, which is called a score function. For multiclass classification $\mathbb{Y} = \{1, \ldots, K\}$, the final predicted class by the prediction function can be written as:

$$f(\boldsymbol{x}) = \operatorname*{arg\,max}_{y \in \mathbb{Y}} \; s(\boldsymbol{x}, y) \; . \tag{1.1}$$

In case of the binary classification $\mathbb{Y} = \{-1, +1\}$, the prediction function could be simplified as

$$f(\boldsymbol{x}) = \text{sign } s(\boldsymbol{x}) \; . \tag{1.2}$$

In both cases, the score function $s(\boldsymbol{x}, y)$ could be considered as a confidence measure of predicting sample $\boldsymbol{x}$ to the class y.

In this thesis, we will constrain our score function to the form of linear model, which is widely used in the literature, and can extend its description power using kernel mapping (Schölkopf and Smola, 2001). Consider a joint feature mapping function $\phi(\boldsymbol{x}, y) : \mathbb{X} \times \mathbb{Y} \to \mathbb{H}$ (Tsochantaridis *et al.*, 2004) that maps the

---

[1] We allow the functions to assume infinite values, as a way to restrict their domains to proper subsets of $\mathbb{X}$. However in the following the convex functions of interest will always be evaluated on vectors that belong to their domains.

samples into a high, possibly infinite, dimensional space. This will also define kernels $K((x,y),(x',y'))$ as $\phi(\boldsymbol{x},y)\cdot\phi(\boldsymbol{x},y)$. We use the standard setting for learning with kernels, the score function is written as a function of the scalar product between a hyperplane $\boldsymbol{w}$ and the transformed sample $\phi(\boldsymbol{x},y)$ [2]:

$$s(\boldsymbol{x},y)=\boldsymbol{w}\cdot\phi(\boldsymbol{x},y)\ .$$

This definition includes the case of training M different hyperplanes, one for each class. Indeed $\phi^j(\boldsymbol{x},y)$ can be defined as

$$\phi^j(\boldsymbol{x},y)=[\mathbf{0},\cdots,\mathbf{0},\underbrace{\phi'^j(\boldsymbol{x})}_{y},\mathbf{0},\cdots,\mathbf{0}],$$

where $\phi'^j(\cdot)$ is a transformation that depends only on the data. Similarly $\boldsymbol{w}$ will be composed by M blocks, $[\boldsymbol{w}_1,\cdots,\boldsymbol{w}_K]$. Hence, by construction, $\boldsymbol{w}\cdot\phi^j(\boldsymbol{x},r)=\boldsymbol{w}_r\cdot\phi'^j(\boldsymbol{x})$. These definitions allow us to have a general notation for the binary and multiclass setting. In the binary case, only one hyperplane is needed. Therefore, the joint feature mapping function is no longer needed, and the score function could be simplified as $s(\boldsymbol{x})=\boldsymbol{w}\cdot\phi(\boldsymbol{x})$. We will mainly focus on multiclass classification tasks across this thesis.

Learning Optimization Problem. Given a set of training samples $\{\boldsymbol{x}_i,y_i\}_{i=1}^N$, the supervised learning optimization problem, which is the main focus of this thesis, is to find the modeling parameter $\bar{\boldsymbol{w}}$ that minimizes the structural risk:

$$\min_{\boldsymbol{w}}\ \lambda h(\boldsymbol{w})+\sum_{i=1}^{N}\ell\left(\boldsymbol{w},\boldsymbol{x}_i,y_i\right)\ , \tag{1.3}$$

where $h(\boldsymbol{w})$ is a regularizer which avoids overfitting, λ is the regularization coefficient that controls the bias-variance tradeoff, and ℓ is some convex, non-negative Lipschitz loss functions, which asses the quality of the hypothesis $\boldsymbol{w}$ on the instance and label pair $(\boldsymbol{x},y)$. Typically, the loss function is a convex upper bound to the true binary or multiple class misclassification loss, to have a continuous and easier to be optimized function.

In the thesis, we consider two different loss functions for the binary and multiple classification tasks respectively. For the binary classification, we will consider the widely used hinge loss (Cristianini and

---

[2] For simplicity we will not use the bias here, it can be easily added by modifying the kernel definition.

Shawe-Taylor, 2000), which can be written as:

$$\ell^{\mathrm{HL}}(\boldsymbol{w}, \boldsymbol{x}, y) = |1 - y\boldsymbol{w} \cdot \phi(\boldsymbol{x})|_+, \tag{1.4}$$

where $|t|_+$ is defined as $\max(t, 0)$.

For multiclass classification, we consider the following multiclass loss function (Crammer and Singer, 2002; Tsochantaridis *et al.*, 2004):

$$\ell^{\mathrm{MC}}(\boldsymbol{w}, \boldsymbol{x}, y) = \max_{y' \neq y} |1 - \boldsymbol{w} \cdot (\phi(\boldsymbol{x}, y) - \phi(\boldsymbol{x}, y'))|_+ \ . \tag{1.5}$$

This loss function is convex and it upper bounds the multiclass misclassification loss.

Online Learning. Online learning is performed in a sequence of T consecutive rounds. On round t, the online classifier (also known as learner) receives a question, usually casted as a vector $\boldsymbol{x}_t$, and is required to provide an answer, denoted by $\hat{y}_t$. After predicting an answer, the learner receives the correct answer, denoted as y_t, and suffers a loss $\ell(s_t(\boldsymbol{x}_t), y_t)$. The ultimate goal of the learner is to minimize the cumulative loss it suffers along the T rounds. To achieve this goal, the learner is allowed to choose a new hypothesis after each round so to minimize the future loss in later rounds.

In this setting, perhaps Rosenblatt's Perceptron algorithm (Rosenblatt, 1958) is the earliest and simplest online learning algorithm. We will start from a brief introduction of it. The Perceptron is designed for the binary setting. On the t-th round the instance $\boldsymbol{x}_t$ is given, and the algorithm makes a prediction

$$\hat{y}_t = \mathrm{sign}(\boldsymbol{w}_t \cdot \boldsymbol{x}_t) \ , \tag{1.6}$$

where $\boldsymbol{w}_t$ is the current hypothesis which is a real number vector of the same dimension of the vector $\boldsymbol{x}_t$. Then the true label is revealed: if there is a prediction mistake, i.e. $\hat{y}_t \neq y_t$, it updates the hypothesis with the rule:

$$\boldsymbol{w}_{t+1} = \boldsymbol{w}_t + y_t \boldsymbol{x}_t \ . \tag{1.7}$$

The hypothesis space is the same as the linear classification function presented in function (1.2), which includes the special case of mapping the sample to a high or even infinite space. In this case, the parameter $\boldsymbol{w}_t$ could not be explicitly written. We assume that the hypothesis space $\mathbb{H}$ is a Reproducing Kernel Hilbert Space (RKHS) with a positive definite kernel function $k : \mathbb{X} \times \mathbb{X} \to \mathbb{R}$ implementing the inner product which satisfies the reproducing property. Therefore, the updating rule (1.7) can be written as:

$$s_{t+1} = s_t + y_t k(\boldsymbol{x}_t, \cdot) \ , \tag{1.8}$$

where y_t and $\boldsymbol{x}_t$ can be stored in the hypothesis. Given the nature of this update, the hypothesis s_t can be written as a kernel expansion (Schölkopf *et al.*, 2000),

$$s_t(\boldsymbol{x}) = \sum_{(\boldsymbol{x}_i, y_i) \in \mathbb{S}_t} y_i k(\boldsymbol{x}_i, \boldsymbol{x}) \ ,$$

where $\mathbb{S}_t$ is the set of samples stored in the hypothesis up to the current time t, usually called the *support set.* The same representation trick can also be used in similar algorithms to update and store the hypothesis function with high or infinite dimensional feature mapping.

In this thesis, several problems are formulated as an online learning problem. We would like to clarify in the beginning that, in most of the experimental scenarios from this thesis, the samples are independent without any sequential relation. Therefore, the online learning algorithms could also be considered as an online optimization process for the batch problems. However, since we did not make any assumption on the generation of the samples, all of our algorithms can work under the online information setting in which unlabeled data comes, a prediction is made, feedback is acquired, and then the model is updated according to the prediction and feedback. The environment could even be adversarial.

Online Convex Programming (Shalev-Shwartz, 2007). The online learning setting can be cast as the task of online convex programming, when 1) the hypothesis space are parameterized and belongs to a convex set denoted as $\mathbb{S}$; 2) the loss function, $g(\boldsymbol{w}) = \ell(\boldsymbol{w}, \boldsymbol{x}, y)$, is a convex function w.r.t. $\boldsymbol{w}$. In online convex programming, the goal of the learner is to minimize the cumulative loss

$$\sum_{t=1}^{T} g_t(\boldsymbol{w}_t) = \sum_{t=1}^{T} \ell_t(\boldsymbol{w}_t, \boldsymbol{x}_t, y_t) \ , \tag{1.9}$$

where the loss function may change over time. The performance of an online convex programming procedure also is usually assessed using the notion of *regret*. Given any fixed vector $\boldsymbol{u} \in \mathbb{S}$, the regret is defined as the excess loss for not consistently choosing the vector $\boldsymbol{u} \in \mathbb{S}$,

$$r(u, T) = \sum_{t=1}^{T} g_t(\boldsymbol{w}_t) - \sum_{t=1}^{T} g_t(\boldsymbol{u}) \ .$$

Ideally, in a separable case, the cumulative loss of the best vector $\boldsymbol{u}^*$ will be zero. In the more realistic case there is no $\boldsymbol{u}$ that correctly predicts the correct answers for all the samples, and $\boldsymbol{u}^*$ is the vector within $\mathbb{S}$ which minimizes the equivalent offline convex programming problem of the objective function (1.9).

Part I

Cues Integration

It might be asked why we have more senses than one. Is it to prevent a failure to apprehend the common sensibles, e.g. movement, magnitude, and number, which go along with the special sensibles? Had we no sense but sight, and that sense no object but white, they would have tended to escape our notice and everything would have merged for us into an indistinguishable identity...
Aristotle, De Anima (350 B.C.E)

Chapter 2

Background and Related Works

This chapter sets the scene for the remaining chapters of Part I. It first elaborates on our motivation for studying cue integration (Section 2.1). Then it provides a brief overview of different cue integration techniques and discusses their connections and differences (Section 2.2). Following that it presents the framework for cue integration learning algorithms, which is the main focus of this thesis. As we use extensively the concepts of online learning and have online optimization techniques as our main optimization tools, the chapter also motivates our choice to formulate our algorithms as within the online learning framework in Section 2.3.

2.1 Motivation for cue integration

In recent years, there has been a lot of interest in designing regression and classification algorithms over multiple cues, based on the intuitive notion that using more features should lead to better performance. In machine vision, in order to overcome the high intraclass variability within each image class, different feature descriptors were designed to be robust to the variations within the classes. However, no single feature type can cope with the variations from every image classes. For example, color information could be used by the classifier to distinguish between zebras and horses, whereas the classifier should be invariant to the shape of both classes (Figure 2.1 (left)). On the other hand, artifacts usually have similar color and texture, and shape information could be essential for classifying these classes, such as cars and motorbikes (Figure 2.1 (right)). Hence it has become popular to combine a set of diverse and

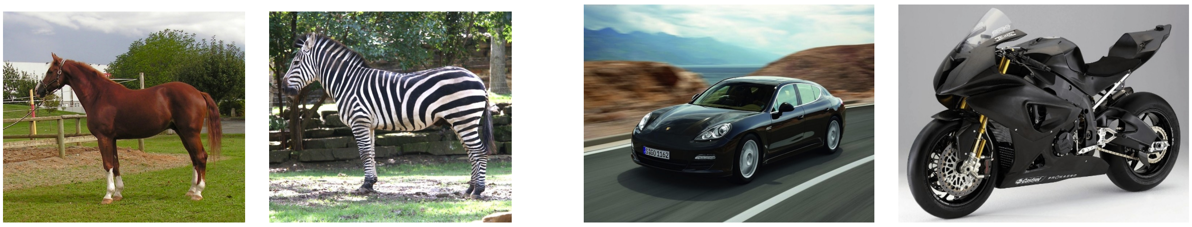


Figure 2.1. Example images of a horse, a zebra, a car and a motorbike. (left) Zebras and horses share similar shape, while color and texture information can be used to distinguish between these two classes. (right) Artifacts, such as cars and motorbikes, usually have similar color and texture, but shape information can be used to distinguish between them.

complementary features to discriminate each class from all other classes in multiclass image classification tasks (Bosch *et al.*, 2007; Varma and Ray, 2007; Gehler and Nowozin, 2009b). Compared to unimodal systems, multimodal systems enjoy even more advantages. Multimodal systems guarantee independent, diverse and information-rich sensory inputs, which makes it possible to achieve robust performance in varied unconstrained settings. Evidences from a wide range of psychophysical studies have found that humans achieve robust perception through the combination and integration of information from multiple sensory modalities (Ernst and Banks, 2002; Ernst and Bulthoff, 2004). Moreover, besides the purpose of decreasing the generalization error, practitioners are often interested in more flexible algorithms which can perform feature selection while training (Chapelle *et al.*, 2002). These motivations have been translated into various cue integration algorithms. In the next section, we will give a general overview of these algorithms.

2.2 Review of Cue Integration Techniques

We first formally define cue integration for classification task.

Definition 2.1 (Classification with Multiple Cue Integration). *Given a set of* N *training samples* $\{\boldsymbol{x}_i, y_i\}_{i=1}^N$ *drawn from an unknown probability distribution consisting of an instance* $\boldsymbol{x}_i \in \mathbb{X}$ *and a class label* $y_i \in \mathbb{Y}$, *and given a set of* F *feature extractors* $e^j : \mathbb{X} \to \mathbb{R}^{D^j}$, $j = 1, \cdots, F$, *where* D^j *denotes the dimensionality of the* j*-th feature. The goal is to learn a classification function* $f : \mathbb{X} \to \mathbb{Y}$ *from the* F *features and the training set, which achieves low generalization error on the new samples.*

For notational convenience, we also define the feature mapping function $\phi^j(\cdot)$ to include the special case of feature extraction. Hence, it could be considered as two independent steps, where in the first step a feature vector is computed from the raw input using the feature extractor e^j, then $e^j(\boldsymbol{x})$ is mapped into a high, possibly infinite, dimensional space. In the case when the identity function is used, the mapped vector $\phi^j(\boldsymbol{x})$ equals to the original feature vector $e^j(\boldsymbol{x})$.

For visual patterns, the sample $\boldsymbol{x}_i$ can be an image or a video clip. For multimodal patterns, we define $\boldsymbol{x}_i$ as the inputs from all the sensors. In this thesis, we do not consider the dynamic information within each pattern. The images considered in our experiments are all still images. However, the cue integration techniques presented here are general, and could be extended to classifiers on time series data. When integrating inputs from several sensors, synchronization between these inputs is another important issue for time series data. However, this is out of the scopes of this thesis, and we will assume that all the inputs are perfectly synchronized. For more through discussion on the synchronization techniques, we refer to the papers of (Sanderson and Paliwal, 2004; Polikar, 2006).

2.2.1 Fusion Types

This chapter is a review of the most important approaches to information fusion. In information fusion literature (Sanderson and Paliwal, 2004), cue integration is often divided into three main categories: *low-level* integration, *middle-level* integration, and *high-level* integration. We do not plan to cover exhaustively every methods in each category. We will only present the most common approaches, and illustrate them with the linear classifier as basic hypothesis. For a through review, we refer the interested readers to (Sanderson and Paliwal, 2004; Polikar, 2006).

Low-level Integration. The low-level integration is also known as feature level fusion. Features extracted from data are combined. Feature vectors could be concatenated together and form a new feature vector. Then the parameters of the classifier could be learned using any supervised learning algorithm. Using the concepts introduced in Section 1.3, the score function using multiple cues could be written as:

$$S(\boldsymbol{x}, y) = \bar{\boldsymbol{w}}_y \cdot \bar{\boldsymbol{\phi}}(\boldsymbol{x}) = \bar{\boldsymbol{w}}_y \cdot [\phi^1(\boldsymbol{x}), \phi^2(\boldsymbol{x}), \cdots, \phi^F(\boldsymbol{x})] \ , \tag{2.1}$$

where $\bar{\boldsymbol{w}}_y$ is the hyperplane for class y, of dimension $\sum_j D^j$. If we decompose the hyperplane $\bar{\boldsymbol{w}}_y$ into F blocks, $[\boldsymbol{w}_y^1, \boldsymbol{w}_y^2, \cdots, \boldsymbol{w}_y^F]$, with each block $\boldsymbol{w}_y^j$ corresponding to a feature, the score function (2.1) can then be transformed into:

$$S(\boldsymbol{x}, y) = \sum_{j=1}^{F} \boldsymbol{w}_y^j \phi^j(\boldsymbol{x}) \ . \tag{2.2}$$

Because the new feature vector is created by the concatenation of several feature descriptors, those feature mapping functions $\phi^j(\boldsymbol{x})$ which map $\boldsymbol{x}$ into an infinite dimensional space could not be easily used, particularly if the optimal kernel for each cue comes from different families with different kernel parameters. However, it is possible to apply these kernels to the cues respectively and sum them together, because the summation of kernels is still a kernel (Schölkopf and Smola, 2001). This approach can be categorized as the Middle-level Integration technic.

Middle-level Integration. Combining two cues at a middle level means that the different features descriptors are processed separated, but they are integrated into a single classifier generating the final hypothesis. Example of this type of fusion is the linear combination of kernels. We define the new kernel function as:

$$\bar{k}(\boldsymbol{x}, \boldsymbol{x}') = \sum_{j=1}^{F} \beta^j k^j(\boldsymbol{x}, \boldsymbol{x}') = \sum_{j=1}^{F} \beta^j \phi^j(\boldsymbol{x}) \cdot \phi^j(\boldsymbol{x}') \ ,$$

where $k^j(\cdot, \cdot)$ is a kernel for the j-th feature descriptor defined by the feature mapping function $\phi^j(\cdot)$, and β^j are non-negative weights corresponding to the importance of the j-th feature. Subsequently, the new combined kernel could be used in a kernel learning algorithm, such as a support vector machine (SVM), to learn the classifier. When a SVM is used, using the representer's theorem (Schölkopf *et al.*, 2000), the resulting score function could be written as

$$\begin{aligned} S(\boldsymbol{x}, y) &= \boldsymbol{\alpha}^y \cdot [\bar{k}(x, x_1), \bar{k}(x, x_2), \cdots, \bar{k}(x, x_N)] \\ &= \sum_{j=1}^{F} \beta^j \boldsymbol{\alpha}^y \cdot [k^j(x_1, x), \cdots, k^j(x_N, x)] \\ &= \sum_{j=1}^{F} \beta^j \boldsymbol{\alpha}^y \cdot [\phi^j(x_1) \cdot \phi^j(x), \cdots, \phi^j(x_N) \cdot \phi^j(x)] \ , \end{aligned} \tag{2.3}$$

where $\boldsymbol{\alpha}^y \in \mathbb{R}^N$ is the SVM dual multipliers. By substitution, equation (2.3) can be converted into the form

$$S(\boldsymbol{x}, y) = \sum_{j=1}^{F} \boldsymbol{w}_y^j \cdot \phi^j(x) \ , \tag{2.4}$$

where $\boldsymbol{w}_y^j$ is a weighted summation of $\phi^j(\cdot)$. It can be written as $\boldsymbol{w}_y^j = \sum_{i=1}^{N} \beta^j \boldsymbol{\alpha}^y[i] \ \phi^j(x_i)$, where $\boldsymbol{\alpha}^y[i]$ is the i-th element of vector $\boldsymbol{\alpha}^y$. In case that $\phi^j(\cdot)$ maps the feature vector into an infinite dimensional space, where $\boldsymbol{w}^j$ could not be stated explicitly, the algorithm can easily store β^j and $\boldsymbol{\alpha}^y$ instead of $\boldsymbol{w}^j$.

Several different approaches have been used to set the weights β^j. Arguably the simplest method is to make $\beta^j = \frac{1}{F}, \forall j = 1, \cdots, F$, which is equivalent to average all the kernels. Several learning based approaches have also been developed, typically by minimizing errors on training data.

High-level Integration. High-level integration is also known as decision or opinion fusion. In high-level integration, a classifier is trained for each cue, then each classifier provides a decision (hard labels) or opinions (usually confidence scores) for the new sample. Depending on the type of outputs from the cue classifiers, these outputs can be combined to make a final decision using approaches such as majority voting, logic operators such as AND and OR, or weighted summation. Typically, decision fusions suffer from many constrains. For example, majority voting requires an odd number of classifiers for binary classification problem, and much more classifiers than the number of classes for a multiclass problem in order to prevent ties. Weighted summation usually suffers less constrains, and empirically this approach also obtained good performance on various tasks (Nilsback and Caputo, 2004; Gehler and Nowozin, 2009b). In this thesis, we will focus on the weighted summation fusion technique. In weighted summation, the scores regarding class y from F experts are combined using:

$$S(\boldsymbol{x}, y) = \sum_{j=1}^{F} \beta_y^j s^j(\boldsymbol{x}, y) = \sum_{j=1}^{F} \beta_y^j \boldsymbol{w}_y^j \cdot \phi^j(\boldsymbol{x}) \ , \tag{2.5}$$

where β_y^j are weights (could possibly be negative), which intuitively define how much the integration classifier should trust the j-th classifier. Since β_y^j is a scalar, it can be eliminated by product with $\boldsymbol{w}_y^j$.

High-level integration could also be perceived as a two-layers structure. In the first layer, a classifier is trained for each cue, where we could use different types of learning algorithms to obtain the classifiers. In

Name	Prediction Function	Coefficients	Learning
Low-level	$\hat{y} = \arg\max_{y \in \mathbb{Y}} \bar{\boldsymbol{w}}_y \cdot \bar{\phi}(\boldsymbol{x})$	$\bar{\boldsymbol{w}}_y \in \mathbb{R}^{\sum_j D^j}$ $\forall j = 1, \cdots, F,\ y = 1, \cdots, K$	$\boldsymbol{w}_y$, jointly
Middle-level ($\text{MKL}_{\text{primal}}$)	$\hat{y} = \arg\max_{y \in \mathbb{Y}} \sum_{j=1}^{F} \boldsymbol{w}_y^j \cdot \phi^j(x)$	$\boldsymbol{w}_y^j \in \mathbb{R}^{D^j}$ $\forall j = 1, \cdots, F,\ y = 1, \cdots, K$	$\boldsymbol{w}_y^j$, jointly
Middle-level (MKL_{dual})	$\hat{y} = \arg\max_{y \in \mathbb{Y}} \sum_{j=1}^{F} \beta^j \boldsymbol{\alpha}^y \cdot$ $[k^j(x_1, x), \cdots, k^j(x_N, x)]$	$\boldsymbol{\beta} \in \mathbb{R}^F,\ \boldsymbol{\alpha}^y \in \mathbb{R}^N$ $\forall y = 1, \cdots, K$	$\boldsymbol{\beta}$ and $\boldsymbol{\alpha}^y$, jointly
High-level	$\hat{y} = \arg\max_{y \in \mathbb{Y}} \sum_{j=1}^{F} \beta_y^j \boldsymbol{w}_y^j \cdot \phi(\boldsymbol{x})$	$\boldsymbol{\beta} \in \mathbb{R}^F$ or $\mathbb{R}^{K \times F},\ \boldsymbol{w} \in \mathbb{R}^{D^j}$ $\forall j = 1, \cdots, F,\ y = 1, \cdots, K$	1. $\boldsymbol{w}_y^j$, independently 2. $\boldsymbol{\beta}$, jointly

Table 2.1. Comparison of different cue integration techniques.

the second layer, the outputs of the classifiers are combined with different flavors (Nilsback and Caputo, 2004; Jie *et al.*, 2009a; Gehler and Nowozin, 2009b).

2.2.2 Comparison and Learning

In this chapter we compare the cue integration techniques presented in the previous section, and discuss the specific learning algorithms. Table 2.2.2 summarizes the cue combination techniques in our unified setting.

From function (2.2), (2.4) and (2.5), it can be observed that the prediction functions of the three integration techniques can be expressed in the same form while using a linear combination hypothesis without considering the bias term. Hence, these three integration formulas are essentially equivalent, and their difference is given only by the specific training procedures used. Which integration techniques should be preferred over the others? These techniques will, in general, give different decision boundaries when trained on the same dataset.

For both low-level and high-level integration algorithms, existing learning algorithms designed for single cue could be directly applied. For low-level integration, the resulting variation of the algorithm will enjoy the same theoretical guarantee of the used learning algorithms. However, in practice, vector concatenation does not work as well as expected, because there is no explicit control over how much each vector contributes to the final decision. Another downside is the dimensionality of the resulting feature vector, which can lead to the curse of dimensionality problem (Duda *et al.*, 2001; Bishop, 2006).

The concept of high-level integration has become more popular in recent years and achieved state-of-art performance in several image recognition benchmark tasks (Gehler and Nowozin, 2009b). This approach dates back to the seminal work of Wolpert (Wolpert, 1992) in the '90s. Both papers (Wolpert, 1992; Gehler and Nowozin, 2009b) use cross-validation methods to gather training set for the second layer. In general, various types of learning algorithms could be used in both layers. The classifiers of the first layer are usually considered as "experts" which could also be treated as "black boxes", and flavors of the algorithms (Nilsback and Caputo, 2004; Gehler and Nowozin, 2009b) usually come from the methods used in the second layer to learn the weights for combining the experts' outputs. For example, a classifier with regularizer that imposes sparsity (Gehler and Nowozin, 2009b) is used which favors the selection of only a subset of experts. The theoretical properties of this family of algorithm are usually due to specifically selected algorithms. In case an algorithm for learning with expert advices is used (Cesa-Bianchi and Lugosi, 2006), it is possible to prove that the combined classifiers will perform at least as well as the classifier using the best cue. Another big advantage of this approach is that it allows us to use the developed learning algorithms without much modification. On the other hand, due to the two-layers structure which splits the optimization process into two phases, this approach is usually sub-optimal. In Chapter 3, we will introduce a multi-cue online learning framework using the two-layers structure, and present its theoretical and empirical results.

Middle-level integration is a relatively new concept. Focusing on the domain of kernel learning, instead of combining kernel classifiers as in high-level integration, the focus of research is on how to build an optimal new kernel as a weighted combination of kernels. A conceptually simple approach (Bosch *et al.*, 2007; Tommasi *et al.*, 2008) is to divide the training data into training and validation set, then find the best combination of the weights which achieve best performance on the validation set. A disadvantage of this approach is that it soon becomes intractable when the number of kernels, *i.e.,* the number of weighting parameters, grows. A different approach with a joint optimization formulation is the Multi Kernel Learning (MKL) (Lanckriet *et al.*, 2004b; Bach *et al.*, 2004; Sonnenburg *et al.*, 2006; Zien and Ong, 2007; Rakotomamonjy *et al.*, 2008; Xu *et al.*, 2008; Nath *et al.*, 2009; Varma and Babu, 2009; Kloft *et al.*, 2009), which solves a joint optimization problem while also learning the optimal weights for combing the kernels. Recent findings seems to indicate that current MKL algorithms do not improve much over the naive baseline of averaging all the kernels (Gehler and Nowozin, 2009b; Cortes, 2009). We argue that the negative results for MKL is mainly due to the complexity of its learning process as well their optimization formulation. The learning usually stops early, before reaching the optimal solution. Again due to the computational complexity, most MKL algorithms are binary, and could not be extend to other types of loss easily. It prevents MKL algorithms from using loss functions which are more suitable for the tasks, such as the multiclass hinge loss (Crammer and Singer, 2002) and structured loss (Tsochantaridis *et al.*, 2004). In Chapter 4 we will present a new online optimization framework to solve efficiently the MKL problem, with a guaranteed convergence rate to the optimal solution. The new MKL algorithms developed from this framework is independent on the particular convex loss function used and achieve state-of-the art performance on many classification tasks.

2.3 Motivation for Online Learning

Cognitive systems learn continuously from experience, updating and enriching their internal models of the environment. This learning mechanism is the main reason why cognitive systems are capable of achieving a robust, yet flexible capability to react to novel stimuli. Moreover, in realistic setting, many problems are intrinsically sequential and data will not be available at the same time. Sometimes the learned concept might change over time. Autonomous robots, for example, need to learn continuously from their surroundings, to adapt to the ever changing environment and maintain satisfactory performance. Another example is the human hearing system gradually adapting to a special accent or a very noisy environment. All these interesting scenarios demand learning algorithms able to update their internal representation efficiently when new training sample arrives. This is opposed to the traditional batch learning algorithms which are not designed to be updated often. Most of the time updating the solution is possible only through a complete re-training, using the training set consisting by the existing samples plus the new training samples. We may still be far from the ultimate dream of life-long learning (Thrun, 1996), but our algorithms move one step forward towards this direction. On the other hand, the algorithms developed in the online learning framework are intrinsically designed to be updated after each sample is received. We believe that online updating is an essential component for learning algorithms carried by artificial intelligent systems. Therefore, in this thesis, we look at the same time at the problem from a learning and optimization point of view, and we design algorithms capable of updating their solutions efficiently when new samples arrive.

From the computational point of view, the computational complexity per update of most online learning algorithms is extremely low compared to the batch methods. Thanks to recent theoretical developments in the machine learning community, several powerful online learning frameworks (Shalev-Shwartz, 2007; Xiao, 2010; Duchi and Singer, 2009) have been proposed, which allows us to minimize the structural risk minimization problem with various regularizers and convex loss functions efficiently.

Using the proposed frameworks, we are able to design efficient online learning algorithms to solve a few complex batch learning problems efficiently with theoretical guarantees.

Chapter 3

Two-Layers Approach: An Online Learning Framework

In this chapter we propose an online learning framework for learning over multiple cues using the concept of *high-level* integration introduced in Section 2.2. We call this framework Online Multi-Cue Learning (OMCL). After an Introduction of the basic framework, Section 3.2 provides a practical algorithm with bounded memory requirement designed using the proposed framework. Section 3.3 presents an approach to transform the solution of the online algorithm to a batch one, which have better generalization performance. Section 3.4 reports experiments on two different scenarios: the first is place recognition, simulating the scenario where a robot is shown an indoor environment composed of several rooms (kitchen, corridor, etc), and later it is supposed to localize and navigate to perform assigned tasks. The second is object categorization, simulating the scenario where the autonomous agent is presented a collection of new objects. For both scenarios, results show that the algorithm learns the new concepts in real time and generalizes well to new concepts. The chapter concludes with a summary discussion. This chapter is partly based on the following publication:

> Jie, L., Orabona, F., and Caputo, B. (2009). An online framework for learning novel concepts over multiple cues. In *Proceedings of the 9th Asian Conference on Computer Vision.*

Algorithm 1 OMCL Framework

```
**Initialize:** cue classifier $f^j$, $\forall i = 1, 2, \cdot, F$; integration classifier $f'$
for $t = 1, 2, \ldots, T$ do
    Receive a new instance $\boldsymbol{x}_t$
    Classifiers in the first layer output scores $s^j(\boldsymbol{x}_t, y)$, $\forall y \in \mathbb{Y}$, $j = 1, \cdots, F$
    Classifier in the second layer predict a label $\hat{y}_t = f'(\boldsymbol{S}_t) = f'([s^1(\boldsymbol{x}_t, 1), \cdot, s^j(\boldsymbol{x}_t, y), \cdot, s^F(\boldsymbol{x}_t, K)])$
    The true label $y_t$ is revealed to the classifiers and they suffer losses
    Classifiers $f$ and $f'$ update their representations
end for
```

3.1 Introducing the OMCL Framework

The OMCL framework is a two-layers wrapper algorithm, as illustrate in Algorithm 1. It is designed using the high-level integration technique. In the first layer, a classifier is trained for each cue, where each classifier provides a confidence interpretation for the target class. On top of these classifiers, another online learning algorithm is added to learn the linear combination of different cues. As in the standard online learning setup, learning takes place at each round, estimating each new hypothesis as a function of the previous one.

Several works have been proposed to combine advices of multiple experts, such as the weighted majority algorithm (Littlestone and Warmuth, 1989) and the exponential weights algorithm (Cesa-Bianchi and Lugosi, 2006, Chapter 2). However, they assume black-box classifiers. Nevertheless, algorithms like the exponential weights algorithm could still be used in the second layer of our framework to combine the inputs from the first layer, which will preserve the same kind of performance guarantees and identify the best cue among the classifiers. Here we want to learn the best combination of classifiers, not just picking the best one. A theoretically motivated method for online learning over multiple cues has been proposed in (Cavallanti *et al.*, 2009), however they assume that all the cues live in the same space, meaning that the same kernel must be used on all the cues. On the other hand, in our framework, we are allowed to use different kernels for different cues.

The OMCL framework allows us to design interesting online multi-cue learning algorithms satisfying various design requirements, such as memory bound and performance guarantees, by plugging in specifical algorithms. Given the regret bounds of the algorithm implemented at the second layer, it is possible to bound the cumulative loss of the resulting multi-cue algorithm.

Theorem 3.1. *Let $(\boldsymbol{x}_1, y_1), \cdots, (\boldsymbol{x}_T, y_T)$ be a sequence of examples where $\boldsymbol{x}_t \in \mathbb{X}$, $y \in \mathbb{Y}$. Suppose that the regret of the linear classifier f' which combines the outputs of classifiers in the second layer is $r'(\boldsymbol{u}, T)$*

for any $\boldsymbol{u} \in \mathbb{R}^{F*K}$*, and the same type of loss functions are used for classifiers at both layers. Then, the cumulative loss of the algorithms under the OMCL framework (Algorithm 1) on this sequence of example is bounded by*

$$\sum_{t=1}^{T} g'(\boldsymbol{w}_t) \leq \min_j (\sum_{t=1}^{T} g^j(\boldsymbol{v}_t^j)) + r'(\boldsymbol{u}, T) \ ,$$

where $g'(\boldsymbol{w}_t) = \ell(\boldsymbol{w}_t, \boldsymbol{S}_t, y_t)$ *is the loss of the classifier from the second layer, and* $g^j(\boldsymbol{v}_t^j) = \ell(\boldsymbol{v}_t^j, \boldsymbol{x}_t, y_t)$ *is the loss of the* j*-th cue classifier from the first layer.*

The theorem suggests that the worst performance of the algorithms developed using OMCL framework is not much worst than the best performance for using a single cue. Suppose that only one unknown cue is useful for the classification task at hand while the others $F-1$ are irrelevant. Algorithms developed using the OMCL framework guarantee that we don't pay much price for not knowing which one is the useful cue. Although the theorem assumes that all the classifiers use the same type of loss function, it still covers most of the interesting cases in the classification tasks. For popular loss functions such as the hinge loss and multiclass loss, similar upper bound also holds for the relative number of mistakes on the same sequence of samples. How to design concrete algorithms using the proposed framework will the topic of the next Section.

3.2 OMCL-BM Algorithm

In this section we introduce an online multi-cue learning algorithm with bounded memory (OMCL-BM) using the proposed framework.

When designing a learning algorithm, we have to keep in mind three properties: performance, memory and learning time. Most online learning algorithms have very low computational complexity and can update efficiently, but their performance is usually lower than similar batch algorithms. On the other hand, multi-cue inputs guarantee diverse and information-rich data, which make it possible to achieve higher and robust performance in varied, unconstrained settings. However, when using multiple inputs, the memory requirements as well as the computational time are linearly or even super linearly (*e.g.,* computational time) proportional to the number of inputs, for both the training and test phase.

We try to design an algorithm to combine the best of both worlds. In the first layer, we first sacrifice

Algorithm 2 Multiclass OMCL-BM Algorithm

```
Input: $\eta^j, \forall j = 1, \cdots, F;\ C$
Initialize: $f_0^j : s_0^j = \mathbf{0},\ \forall j = 1, \cdots, F;\ \ f_0' : \boldsymbol{W}_0 = [\boldsymbol{w}_{1,0}, \cdots, \boldsymbol{w}_{y,0}, \cdots, \boldsymbol{w}_{K,0}] = \mathbf{0}$
for $t = 1, 2, \ldots, T$ do
  Receive new instance $\boldsymbol{x}_t$
  Compute $s_t^j(\boldsymbol{x}_t, y),\ \forall y = 1, \cdots, K$
  Predict $\hat{y}_t = \arg\max_{y \in \mathbb{Y}} \boldsymbol{w}_{y,t-1} \cdot \boldsymbol{S}_t = \arg\max_{y \in \mathbb{Y}} \boldsymbol{w}_{y,t-1} \cdot [s_t^1(\boldsymbol{x}_t, 1), \cdots, s_t^j(\boldsymbol{x}_t, y), \cdots, s_t^F(\boldsymbol{x}_t, K)]$
  Receive $y_t$
  for $j = 1, 2, \cdots, F$ do
    Compute $y_t' = \arg\max_{y \neq y_t} s_t^j(\boldsymbol{x}_t, y)$
    Compute $\ell_t^j = \max(0, 1 - s_t^j(\boldsymbol{x}_t, y_t) + s_t^j(\boldsymbol{x}_t, y_t'))$
    if $\ell_t^j > 0$ then
      Set $U^j = k^j\left((\boldsymbol{x}_t, y_t), \cdot\right) - k\left((\boldsymbol{x}_t, y_t'), \cdot\right)$
      Compute projection operator $P_{t-1}^j$ (Orabona et al., 2009, Figure 4)
      Compute projection error $\Delta = \|U^j - P_{t-1}^j\left(U^j\right)\|$
      if $y_t = f_{t-1}^j(\boldsymbol{x}_t)$ or $\Delta \leq \eta^j$ then
        Compute $\alpha_t^j = \min\{\frac{\ell_t^j}{\|P_{t-1}^j(U^j)\|^2}, 2\frac{\ell_t^j - \|\Delta\|/\eta}{\|P_{t-1}^j(U^j)\|^2}, 1\}$
        Projection update: $s_t^j = s_{t-1}^j + \alpha_t^j P_{t-1}^j\left(U^j\right)$
      else
        Normal update: $s_t^i = s_{t-1}^i + U^j$
      end if
    end if
  end for
  Compute $y_t' = \arg\max_{y \neq y_t} \boldsymbol{w}_{y,t-1} \cdot \boldsymbol{S}_t$
  Compute $\ell_t = \max(0, 1 - (\boldsymbol{w}_{y_t,t-1} - \boldsymbol{w}_{y_t',t-1}) \cdot \boldsymbol{S}_t)$
  Set $\tau_t = \min\{C, \frac{\ell_t}{2\|\boldsymbol{S}_t\|^2}\}$
  Update $\boldsymbol{w}_{y_t,t} = \boldsymbol{w}_{y_t,t-1} + \tau_t \boldsymbol{S}_t,\ \ \boldsymbol{w}_{y_t',t} = \boldsymbol{w}_{y_t',t-1} - \tau_t \boldsymbol{S}_t$
end for
```

performance in favor of bounded memory growth and fast update of the solution for each separate cue, by using a budget online learning algorithm with bounded memory usage (Orabona *et al.*, 2009). We then recover the performance by using multiple cues with a linear Passive-Aggressive (Crammer *et al.*, 2006) algorithm which tries to learn an optimal combination of the outputs of the classifiers in the first layer. The pseudo code of the multiclass version of the proposed algorithm is shown in Algorithm 2. In the rest of this section, we briefly introduce the basic online learning algorithms adopted in OMCL-BM.

3.2.1 Projectron++ Algorithm

In the kernel Perceptron, each time an update occurs, a new instance is added to the support set, denoted by $\mathcal{S}_t$. This will eventually lead to a memory explosion. As we aim to use the algorithm in applications where data must be acquired continuously in time, we need an algorithm with slower memory growth or even bounded memory requirements. The Projectron++ (Orabona *et al.*, 2009) algorithm is a Perceptron-like algorithm bounded in space and time complexity. The core idea of the algorithm comes from the work of Downs et. al. (Downs *et al.*, 2001) on simplifying the Support Vector Machine solutions. Hence, instead of updating the hypothesis every time a prediction mistake is made, or when the prediction is correct with low confidence, *i.e.,* when $0 < y_t f_{t-1}(\boldsymbol{x}_t) < 1$, the Projectron++ first checks if the update can be expressed as a linear combination of vectors in the support set. In case of kernels with infinite dimensional space such as the Gaussian kernel, it is not possible to find a finite number of linearly independent vector which span the whole space. The algorithm could still approximate the concept of linear independence with a finite number of vectors. When such linear combination (possibly approximately) is possible, the kernel transformation of the new instance $\boldsymbol{x}_t$ can be written as

$$k(\boldsymbol{x}_t, \cdot) \approxeq P_{t-1}\left(k\left(\boldsymbol{x}_t, \cdot\right)\right) = \sum_{\boldsymbol{x}_i \in \mathcal{S}_{t-1}} d_i k(\boldsymbol{x}_i, \cdot) \ ,$$

where P_{t-1} is the projection operator, which could be computed efficiently and incrementally at each step using the method described in (Orabona *et al.*, 2009, Figure 4). Then the coefficients in the old hypothesis are changed to reflect the addition of the instance during updating. The concept of linear independence can be approximated and tuned by a parameter η that measures the quality of the approximation. If the instance can be approximated within an error η, in another word, if $||s_t^{\mathrm{N}} - s_t^{\mathrm{P}}|| \leq \eta$, where s_t^{N} is a temporary hypothesis using normal update, and s_t^{P} is a projected hypothesis using the projection, the projected hypothesis will be used. If the instance and the support set are linearly independent, the instance is added to the set, as for the Perceptron.

The size of the support set of the Projectron algorithm is bounded (Orabona *et al.*, 2009, Theorem 1). As a result, the support set of the OMCL-BM algorithm is also bounded because in the second layer the algorithm will only use a linear classifier whose memory requirement is a constant number.

3.2.2 Passive-Aggressive Algorithm

The Passive-Aggressive (PA) algorithm was introduced by Crammer *et al.* (2006). It is a margin based online learning algorithm. The main idea of PA is to update classifiers by the smallest width, so that the currently referred training sample may be classified correctly. In multiclass classification task, we consider the loss function ℓ^{MC}. On round t, when the learner suffers a loss, PA updates the new weight vector $\boldsymbol{w}_t$ to be the solution of the following convex optimization problem

$$\boldsymbol{W}_t = \arg\min_{\boldsymbol{w}} \frac{1}{2}\|\boldsymbol{W} - \boldsymbol{W}_{t-1}\|^2 + C\xi \ , \quad \text{s.t.} \quad \ell^{\mathrm{MC}}(\boldsymbol{W}_t, \boldsymbol{S}_t, y_t) \leq \xi \ \text{ and } \ \xi \geq 0 \ . \tag{3.1}$$

It can be verified that the solution to the optimization problem in equation (3.1) has a closed form solution,

$$\boldsymbol{w}_{y_t,t} = \boldsymbol{w}_{y_t,t-1} + \tau_t \boldsymbol{S}_t \ , \quad \text{and} \quad \boldsymbol{w}_{y'_t,t} = \boldsymbol{w}_{y'_t,t-1} - \tau_t \boldsymbol{S}_t \ ,$$

with,

$$y'_t = \arg\max_{y \neq y_t} \boldsymbol{w}_{y,t} \cdot \boldsymbol{S}_t \ , \quad \text{and} \quad \tau_t = \min\{C, \ \frac{\ell^{\mathrm{MC}}}{2\|\boldsymbol{S}_t\|^2}\} \ .$$

Here, $\boldsymbol{w}_{y,t}$ is the y-th block of the parameter $\boldsymbol{W}_t$, which corresponds to the hyperplane of the y-th class. The PA algorithm makes aggressive updates in the sense that even a small loss forces an update of the hypothesis. Empirically it obtains very good performances on many classification tasks.

3.3 Online to Batch Conversion

Online algorithms are meant to be constantly used in teacher-student scenarios. Hence the update process will never stop. However it is possible to transform them to batch algorithms, that is to stop the training and to test the current hypothesis on a separate test set. It is known that when an online algorithm stops the last hypothesis found can have an extremely high generalization error. This is due to the fact that the online algorithms are not converging to a fixed solution, but they are constantly trying to "track" the best solution. If the samples are Independent & Identically Distributed (IID), to obtain a good batch solution one can use the *average* of all the solutions found, instead of the last one. This choice gives also theoretical guarantees on the generalization error (Cesa-Bianchi *et al.*, 2004).

Our system produces $F+1$ different hyperplanes at each round. In principle we could simply average each hyperplane separately, but this would break the IID assumption of the inputs for the second layer. So we propose an alternative method: given that the entire system is linear, it can be viewed as producing only one hyperplane at each round, that is the product of the two hyperplanes. Hence we average this unique hyperplane and in the testing phase we predict with the formula:

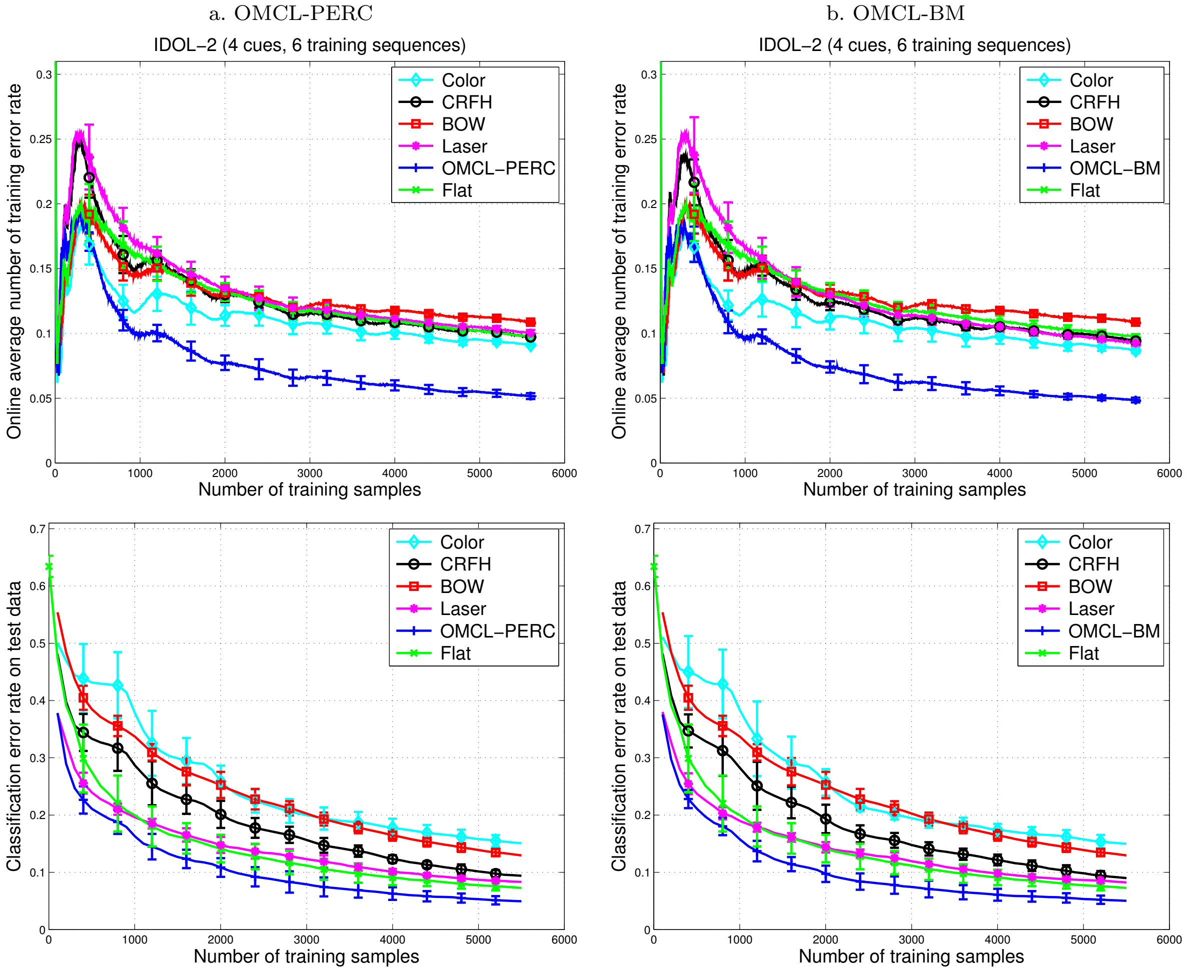
$$\arg\max_{y \in \mathbb{Y}} \left(\frac{1}{T} \sum_{t=1}^{T} \boldsymbol{w}_{y,t} \cdot [s_t^1(\boldsymbol{x}_t, 1), \cdots, s_t^j(\boldsymbol{x}_t, y), \cdots, s_t^F(\boldsymbol{x}_t, K)] \right).$$

Note that, as f_t^j can be written as a kernel expansion, the averaging does not imply any additional computational cost, but just an update of the coefficients of the expansion. We use this approach as it guarantees a theoretical bound (Cesa-Bianchi *et al.*, 2004) and it was also found to perform better in practice.

3.4 Experiments and Results

We present an experimental evaluation of our approach on two different scenarios, supported by two publicly available databases. The first scenario is place recognition for a mobile robot, and the experiments were conducted on the IDOL2 dataset. The second scenario consists of learning new object categories, and the experiments were conducted on the ETH80 dataset. As most of the databases used within this thesis are popular publicly available databases and feature extraction is not the focus of this thesis, while explaining the experimental setup here, as well as in the rest part of this thesis, we will omit the introduction of the datasets and the details of feature extraction. These details can be found in Appendix A. Both experiments can be considered as a *teacher-student* scenario, that is when a new concept (rooms, or objects) is presented to the robot, the system (student) can ask the human teacher to provide a label. Since the algorithms are intended to be used by a mobile system, where both computational and memory resources are limited, OMCL-BM algorithm is particularly suitable for these applications.

For all experiments, we compared the performance and the memory requirements to the standard Perceptron algorithm by replacing the Projectron++ algorithm in our framework (OMCL-PERC). We also compared our algorithms to two different cues combination algorithms: a low-level integration method and the majority voting algorithm. For the low-level integration baseline (Flat), we concatenate all the features of different cues into a long feature vector, then learn the classifier using the Projectron++ classifier. The majority voting algorithm (Vote) predicts the label by choosing the class which receives the highest number of votes. As for a majority voting algorithm in the multiclass case, the number of experts (number of cues in our experiments) required by the algorithm which guarantees a unique


a. OMCL-PERC

b. OMCL-BM

Figure 3.1. Average online training error rate (top) and classification error rate (bottom) on the test set of the IDOL2 dataset as a function of the number of training samples.

solution will grow exponentially with the number of classes. Although it does not happen very often in practice, we show that sometimes two or more classes receive an equal number of votes, especially when the number of cues is relatively small compared to the number of classes. We determined all of our online learning and kernel parameters via cross-validation.

3.4.1 First Scenario: Place Recognition

We performed the first series of experiments on the IDOL2 database (see Appendix A). For experiments, we used the same setup described in (Luo *et al.*, 2007, Section V, Part B). We considered the 12 sequences acquired by the robot Dumbo, and divided them into training (6 sequences) and test sets (6 sequences),

cue	OO	CR	TO	KT	PA	ALL
Color	28.4	9.5	8.1	9.9	31.4	17.4
CRFH	14.2	4.1	15.4	15.4	11.1	12.0
BOW	21.5	6.9	17.5	11.4	8.4	13.1
Laser	7.6	3.7	8.5	10.7	12.9	8.7
OLMC-BM	7.6	2.5	1.9	6.7	13.3	6.4
Flat	5.7	2.7	7.5	8.5	11.8	7.2
Vote	11.7 (6%)	3.3 (2%)	5.3 (3%)	5.2 (3%)	9.1 (7%)	6.9 (4%)

Table 3.1. Place recognition error rate using different cues after the last training round. Each room is considered separately during testing, and it contributes equally to the overall results as an average. It shows that the OMCL algorithm achieves better performance than that of using each single cue. For Vote, the percentage of the test data which have two or more classes receive equal number of votes is reported in the bracket below the error rate. Hence the algorithm can not make a definite prediction. Therefore we consider that the algorithm makes a prediction error.

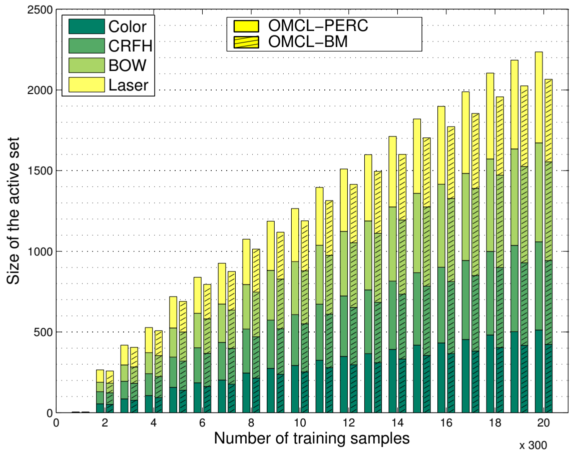


Figure 3.2. Average size of support set for different algorithms on the IDOL2 dataset as a function of the number of training samples.

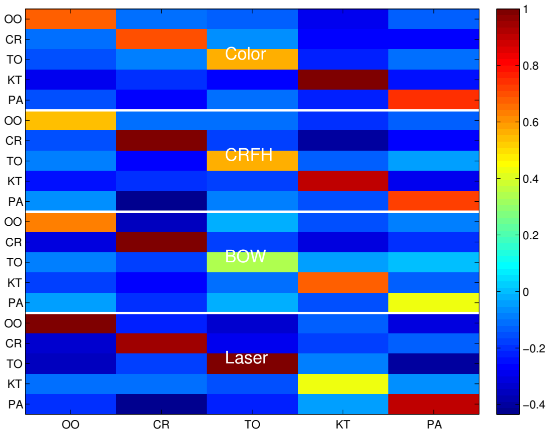


Figure 3.3. Visualization of the average normalized weights obtained by OMCL-BM on the IDOL2 dataset at the last training round for combining the confidence outputs of Projectron++.

where each training sequence has a corresponding one in the test sets, captured under roughly similar conditions. Similarly, each sequence was divided into 5 subsequences. In total, we considered 12 different permutations of training and test sets. Learning is done in chronological order, i.e. how the images were captured during the acquisition of the dataset. During testing, all the sequences in the test set were taken into account.

Figure 3.1 reports the average online training and recognition error rate on the test set, where the average online training error rate is the number of prediction mistakes the algorithm makes on a given input sequence normalized by the length of the sequence. Figure 3.2 shows the size of the support sets as a function of the number of training samples. Here, the size of the support set of OMCL-BM is close to that of OMCL-PERC. This is because the support set of OMCL-PERC is already very compact. Since the online training error rate is low, both algorithms do not update very frequently. In Table 3.1 we summarize the results using each cue after finishing the last training round. We see that our algorithm outperforms both the low-level integration techniques and the majority vote algorithm. The majority vote algorithm could not make a definite prediction on approximately 4% of the test data, because there are two or more classes which received an equal number of votes. Moreover, we also visualize the weights obtained by our algorithm at the last learning round in Figure 3.3. We can see that the weights on the diagonal of the matrix, which corresponds to the multiclass classifiers' confidence interpretations on the same target category, have the highest values.

3.4.2 Second Scenario: Object Categorization

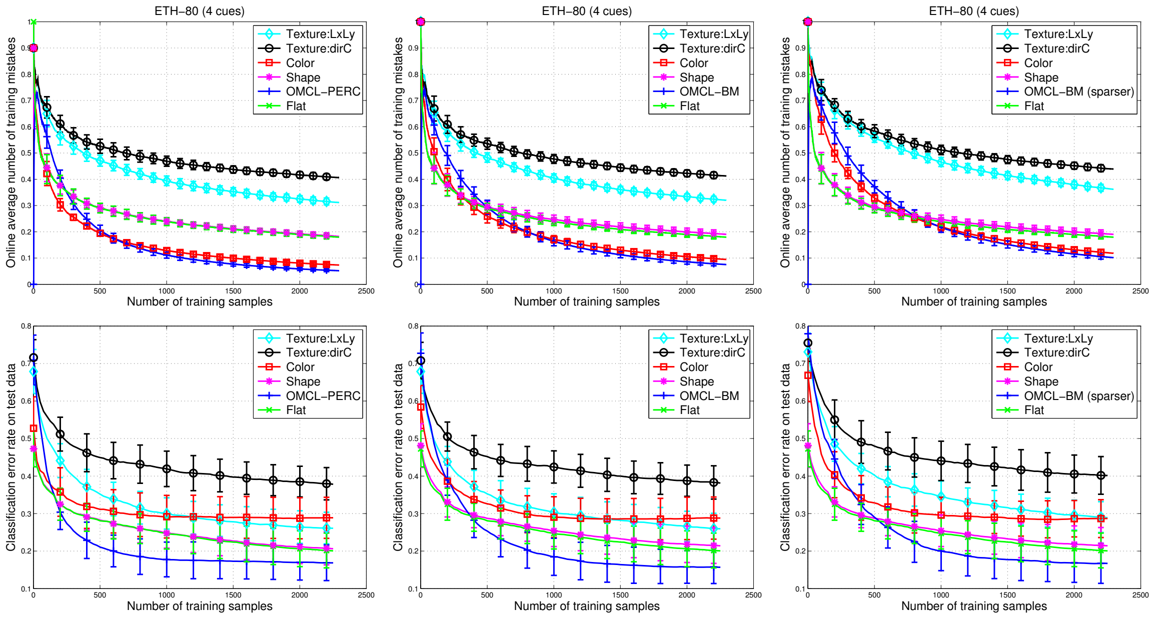

a. OMCL-PERC b. OMCL-BM c. OMCL-BM (sparser)

Figure 3.4. Average online training error rate (top) and classification error rate (bottom) for classifying never seen objects on the ETH-80 dataset as a function of the number of training samples. We see that all algorithms achieve roughly similar performance (OMCL-BM is slightly better), OMCL-PERC converges earlier than OMCL-BM.

We tested the algorithm on the ETH-80 objects dataset. We randomly selected 7 out of the 10 objects for each category as training set, and the rest of them as test set. All the experiments were performed over 20 different permutations of the training set.

Cues	apple	car	cow	cup	dog	horse	pear	tomato	all
$L_x L_y$	24.0	3.41	53.4	29.4	29.6	49.1	7.11	9.4	25.6
$DirC$	22.4	21.9	57.0	40.4	63.9	56.4	31.0	3.6	37.1
Color	37.8	20.1	13.7	19.2	46.5	63.5	32.6	0.6	29.2
Shape	30.9	1.5	28.5	2.3	32.2	28.1	3.6	35.9	20.3
OLMC-BM	4.1	5.2	25.3	1.7	33.1	39.5	6.6	6.5	15.2
Flat	29.3	1.5	28.2	2.0	32.1	28.0	3.3	35.1	20.0
Vote	24.2 (19%)	1.3 (1%)	30.1 (15%)	11.3 (9%)	34.0 (18%)	41.2 (20%)	3.6 (3%)	6.1 (5%)	19.0 (11%)

Table 3.2. Categorization error rate for different objects using different cues after finishing the last training round. We could see that our algorithm outperforms the Flat baseline, the Vote algorithm and the cases when using each cue alone. It also shows that some cues are very descriptive of certain objects, but not of the others. For example, the color feature achieves almost perfect performance on tomato, but its performance on other objects is low. It also supports our motivation on designing multi-cues algorithms. For the Vote algorithm, the percentage of test data which have two or more classes receives equal number of votes is reported in the bracket.

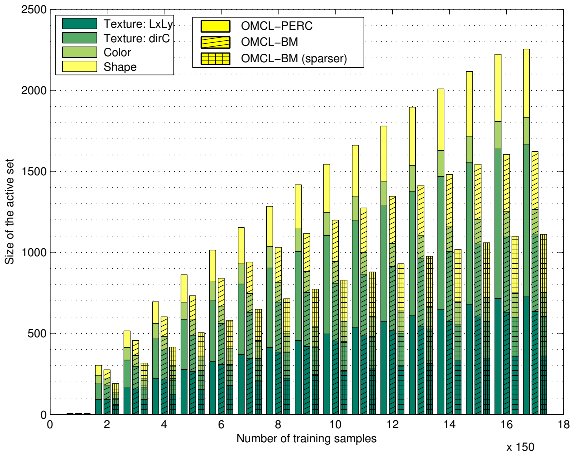


Figure 3.5. Average size of support set for different algorithms on the ETH80 dataset as a function of the number of training samples.

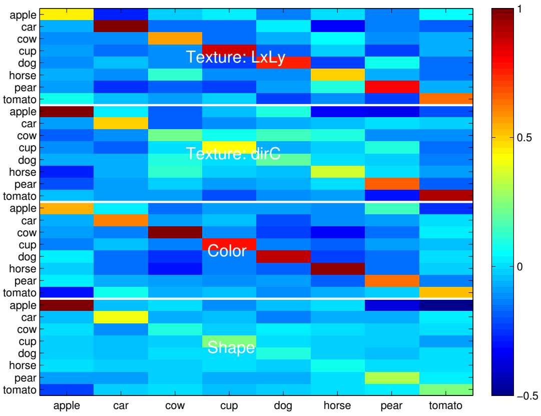


Figure 3.6. Visualization of the average normalized weights obtained by OMCL-BM at the last training round for combining the confidence outputs of Projectron++.

We first show the behavior of the algorithms over time. In Figure 3.4, we show average online training error rate (top) as well as the average classification error rate (bottom) on never seen objects as a function of the number of learned samples. In the experiments, we used two different settings of the η parameters, labeled as OMCL-BM and OMCL-BM (sparser). The growth of the support set as a function of the number of samples is depicted in Figure 3.5. We see that the Projectron++ algorithm obtains similar performance as the Perceptron algorithm with less than 3/4 (OMCL-BM) and 1/2 (OMCL-BM (sparser)) of the size of the support set. It is also interesting to observe that although the color feature achieve very low online training error rate (see Figure 3.4, top), it has high test error rate because the color feature is less discriminative in categorization tasks. However, our cue integration algorithm still achieve robust performance using other features.

Finally, in Table 3.2 we summarize the error rate using different cues for each category after finishing the last training round (Projectron++). In this table we can see that our algorithm outperforms its competitors. Figure 3.6 shows the normalized average learned weights for the linear Passive-Aggressive classifiers. From the figures, we could see that the weights on the diagonal of the matrix, which corresponds to a multiclass classifier's confidence interpretation on the same target category, have highest values. As opposed to this, confidence output corresponding to the most confusing category usually have lowest weights: for instance, when predicting apple using shape cue, the weight for the confidence corresponding to tomato have smallest value (bottom left value in the matrix).

3.5 Summary

In this chapter, we presented an online framework using a two-layers structure for learning from multiple cues. This high-level cue integration algorithm does not require using cross-validation techniques to collect training data in order to learn optimal combination of different cues. In our algorithm, both the cue classifiers and the combination classifier are trained in an online fashion and update their models when a new sample arrives. Therefore, the algorithm is very efficient and easy to implement, and can be used in real time on mobile platforms with limited computational resources. Using the proposed online framework, we designed the OMCL-BM algorithm with bounded memory. Experiments on two datasets showed that the developed algorithm is able to learn a linear weighted combination of the marginal output of classifiers on each source. This method outperforms the case when we use each cue alone. Moreover, it achieves a performance comparable to the batch performance (Leibe and Schiele, 2003; Luo *et al.*, 2007) with a much lower memory and computational cost. More specifically, using the budget Projectron++ algorithm, the new multi-cue algorithm can reduce the problem of the expansion of the input space and memory requirement. Thanks to the robustness gained by using multiple cues, the algorithm can reduce more the support set without any significant loss in performance. This trade-off would be a potentially useful function for applications working in a highly dynamic environment and with limited memory resources, particularly for systems equipped with multiple sensors.

However, since the classifier for each cue is learned independently and the two-layers structure splits the optimization process into two phases, this type of approach is usually sub-optimal. In the next chapter, we will present a different approach which solves a joint optimization problem while also learns the optimal weights to combine the cues. This approach is theoretically found, and empirically it also consistently achieves better performance on several different tasks.

Chapter 4

One-Layer Approach: MKL with Online-Batch Optimization

In this chapter we present several Multi Kernel Learning (MKL) algorithms which can be categorized as a type of middle-level feature integration algorithm. Section 4.1 introduces briefly the original MKL formulation and reviews the related works. In Section 4.2, we introduce the l_p norm MKL formulation which is strongly convex. This opens the possibility to design efficient online (OM-2, Section 4.3) and batch (OBSCURE, Section 4.4) algorithms for solving the proposed formulation efficiently. Then a sparse MKL formulation using an elastic net form of regularization and its optimization procedure are presented in Section 4.5, that achieves better performance on problems which are sparse. For all the algorithms we show their theoretical guarantees and conduct experiments to support our claims. It can be found that our algorithms achieve state-of-the-art performance on many classification tasks. This chapter is mainly based on the following publications:

> Jie, L., Orabona, F., Fornoni, M., Caputo, B., and Cesa-Bianchi, N. (2010). OM-2: An online multiclass multi-kernel learning algorithm. In *Proceeding of the 4th IEEE Online Learning for Computer Vision Workshop.*
>
> Orabona, F., Jie, L., and Caputo, B. (2010). Online-Batch Strongly Convex Multi Kernel Learning. In *Proceedings of the 23rd IEEE Conference on Computer Vision and Pattern Recognition.*
>
> Orabona, F., and Jie, L., (2011). Ultra-Fast Optimization Algorithm for Sparse Multi Kernel

Learning. In *Proceedings of the 28th International Conference on Machine Learning.*

4.1 Introduction

In the MKL framework (Lanckriet *et al.*, 2004a; Bach *et al.*, 2004; Sonnenburg *et al.*, 2006; Zien and Ong, 2007; Rakotomamonjy *et al.*, 2008; Varma and Babu, 2009), different feature descriptors are transformed into kernels, where each feature corresponds to one or several kernels (computed using various families of kernels, or the same type of kernel with different kernel hyper-parameters). The integration is done through a weighted linear combination of kernels, where the importance of each kernel is reflected through the weight assigned to it. Rather than taking a brute-force approach to set the weights through cross-validation, the MKL algorithms solve a joint optimization problem which learns the classifier while also learning the optimal weights for combining the kernels. MKL methods are theoretically founded, because they are based on the minimization of an upper bound of the generalization error (Kakade *et al.*, 2009; Cortes *et al.*, 2010), like in standard SVM. The MKL optimization problem was first proposed in Bach *et al.* (2004) and extended to multiclass in Zien and Ong (2007). Using the group norm notation introduced before (Section 1.3), the classic MKL optimization problem can be written as

$$\min_{\bar{\boldsymbol{w}}} \quad \frac{\lambda}{2}\|\bar{\boldsymbol{w}}\|_{2,1}^{2} + \frac{1}{N}\sum_{t=1}^{T} \ell\left(\bar{\boldsymbol{w}}, \boldsymbol{x}_t, y_t\right) \ , \tag{4.1}$$

where $\bar{\boldsymbol{w}}$ is the concatenation of F vector $\boldsymbol{w}^j$ as defined in Section 1.3, with each each $\boldsymbol{w}^j$ corresponds to a kernel. The l_1 norm constraint is used by most MKL approaches to impose sparsity on the weights of the combination. Hence the l_1 group norm favors a solution in which only few hyperplanes have a norm different from zero. An equivalent formulation can be derived from the above one through a variational argument (Bach *et al.*, 2004):

$$\min_{\boldsymbol{w}^j, \beta^j \geq 0} \ \frac{\lambda}{2}\left(\sum_{j=1}^{F} \frac{\|\boldsymbol{w}^j\|_2}{\alpha^j}\right)^2 + \frac{1}{N}\sum_{i=1}^{N} \ell\left(\bar{\boldsymbol{w}}, \boldsymbol{x}_t, y_t\right) \ , \qquad \text{s.t.} \quad \|\beta\|_1^2 \leq 1 \ , \tag{4.2}$$

where β^j is a positive weight for the linear summation of the F kernels. This formulation has been used in Bach *et al.* (2004); Sonnenburg *et al.* (2006), while in Rakotomamonjy *et al.* (2008) the proposed formulation is slightly different, although it can be proved to be equivalent. The reason to introduce this variational formulation is to use alternating optimization strategy to efficiently solve the constrained minimization problem. However, due to the fact that the l_1 norm is not smooth, its optimization algorithm is rather complex and its rate of convergence is usually slow.

Studies in the past few years focused on how to solve the MKL optimization problem efficiently. The original MKL problem by Lanckriet *et al.* (2004a) was cast as a semidefinite programming (SDP). SDP are known to have poor scalability, hence much of the subsequent research focused on devising more efficient optimization procedures. The first step towards practical MKL algorithms was to restrict the weights coefficients to be non-negative. In this way, it was possible to recast the problem as a much more efficient semi-infinite linear programming (SILP) (Sonnenburg *et al.*, 2006). This has allowed to solve the MKL problem with alternating optimization approaches (Sonnenburg *et al.*, 2006; Rakotomamonjy *et al.*, 2008; Xu *et al.*, 2008; Nath *et al.*, 2009), first optimizing over the kernel combination weights, with the current SVM solution fixed, then finding the SVM solution, given the current weights. One advantage of the alternating optimization approach is that it is possible to use existing efficient SVM solvers, such as (Joachims, 1998; Chang and Lin, 2001), for the SVM optimization step.

For algorithms using the alternating optimization approach, it is usually not possible to prove a bound on the maximum number of iterations needed, even if they are known to converge. They need to solve the inner SVM problem till optimality. In fact, to guarantee convergence, the solution needs to be of a high enough precision so that the kernel weight gradient computation is accurate. On the other hand, due to it computational complexity, the learning process is usually stopped early, before reaching the optimal solution, based on the common assumption that it is enough to have an approximate solution of the optimization function. Considering the fact that these MKL algorithms are solved based on their dual representation, this might mean being stopped far from the optimal solution (Hush *et al.*, 2006; Chapelle, 2007), with unknown effects on the convergence. Another important issue is that almost all the previous approaches focus on binary classification, and could not be extended to other types of loss easily due to their optimization process. To the best of our knowledge, before our papers were published (Jie *et al.*, 2010; Orabona *et al.*, 2010), the only multiclass MKL solver was proposed by Zien and Ong (2007). It solves the problem using SILP, which does not scale very well and is infeasible for problems with hundreds of classes and a few thousand examples. In the following we show that it is possible to efficiently minimize directly the formulation in (4.1), or at least one variation of it. Since the algorithms directly solve the optimization in the primal formulation, it allows us to use any complex convex loss function, such as the multiclass hinge loss (Crammer and Singer, 2002) and the structured loss (Tsochantaridis *et al.*, 2004), with minimal changes to the algorithm. Moreover, we are able to prove theoretical guarantees on the convergence rate, which is new in the MKL algorithms.

4.2 The l_p Norm MKL Formulation

In the original MKL formulation (4.1), the l_1 norm is used to induce sparsity in the domain of the kernels. This means that the solution of the optimization problem will select a subset of the F kernels. However, even if sparsity can be desirable for specific applications, it could bring to a decrease in performance. Moreover the problem in (4.1) is not strongly convex (Kakade *et al.*, 2009), so its optimization algorithm is rather complex and its rate of convergence is usually slow.

We generalize the original optimization problem using a generic group norm

$$\min_{\bar{\boldsymbol{w}}} \quad \frac{\lambda}{2}\|\bar{\boldsymbol{w}}\|_{2,p}^2 + \frac{1}{N}\sum_{i=1}^{N} \ell\left(\bar{\boldsymbol{w}}, \boldsymbol{x}_i, y_i\right), \tag{4.3}$$

where $1 < p \leq 2$. We define $g(\bar{\boldsymbol{w}}) = \frac{\lambda}{2}\|\bar{\boldsymbol{w}}\|_{2,p}^2 + \frac{1}{N}\sum_{i=1}^{N} \ell\left(\bar{\boldsymbol{w}}, \boldsymbol{x}_i, y_i\right)$ and $\bar{\boldsymbol{w}}^*$ equals to the optimal solution of (4.3), that is $\bar{\boldsymbol{w}}^* = \arg\min_{\bar{\boldsymbol{w}}} g(\bar{\boldsymbol{w}})$.

The additional parameter p allows us to decide the level of sparsity of the solution. Moreover this formulation has the advantage of being λ/q-strongly convex (Kakade *et al.*, 2009). Strongly convexity is a key property to design fast batch and online algorithms: the more a problem is strongly convex the easier it is to optimize it (Shalev-Shwartz and Singer, 2007; Kakade *et al.*, 2009). Many optimization problems are strongly convex, as the SVM objective function. When p tends to 1, the solution gets close to the sparse solution obtained by solving (4.1), but the strong convexity vanishes. Setting p equals to 2 corresponds to using the unweighted sum of the kernels. In the following we will show how to take advantage of the strong convexity to design a fast algorithm to solve (4.3), and how to have a good convergence rate even when the strong convexity is close to zero. Another similar l_p norm formulation has been proposed independently from ours (Orabona *et al.*, 2010) in Kloft *et al.* (2009). As for (4.1) and (4.2), using (Micchelli and Pontil, 2005, Lemma 26) it is possible to prove that they are equivalent through a variational argument. Based on the formulation of (Kloft *et al.*, 2009), Vishwanathan *et al.* (2010) derived the dual of a variation of the l_p MKL problem, suited to be optimized with the popular

Algorithm 3 Follow the Regularized Leader

```
Initialize: $\boldsymbol{w}_1 = \mathbf{0}$
for $t = 1, 2, \dots, T$ do
    Receive new instance $\boldsymbol{x}_t$ and predict using $\boldsymbol{w}_t$
    Receive label $y_t$
    $\boldsymbol{w}_{t+1} = \nabla h^*(-\sum_{i=1}^{t} \eta_t \partial l(\boldsymbol{w}_i, (x_i, y_i)))$
end for
```

Sequential Minimal Optimization algorithm (Platt, 1999). However, their algorithm is limited to the hinge loss function. This limitation on the use of a particular loss functions is common to all the recent MKL optimization algorithms.

We have chosen to weight the regularization term by λ and divide the loss term by N, instead of the more common formulation with only the loss term weighted by a parameter C. This choice greatly simplifies the math of our algorithm. However the two formulations are fully equivalent when setting $\lambda = \frac{1}{CN}$. Hence a big value of C will correspond to a small value of λ.

4.3 Online MKL: the OM-2 algorithm

In this section, we propose a theoretically motivated online learning algorithm, which we call OM-2 for Online Multi-loss Multi-kernel learning. This algorithm was originally proposed by Jie *et al.* (2010) for multiclass classification task using a multiclass loss function, but it can be extended to any convex loss. In the rest of this section, we first introduce the "Follow The Regularized Leader" (FTRL) framework (Shalev-Shwartz, 2007; Kakade *et al.*, 2009) which we used to solve the online l_p norm MKL problem (Section 4.3.1). Then we describe the algorithm and analyze its performance on any arbitrary sequence of observations (Section 4.3.2). Finally, we conduct experiments on standard benchmark datasets (Section 4.3.4).

4.3.1 FTRL Framework

An approach to solve the online learning problem is to use the FTRL framework. This corresponds to using at each step the solution of the optimization problem

$$\bar{\boldsymbol{w}}_{t+1} = \arg\min_{\bar{\boldsymbol{w}}} \quad h(\bar{\boldsymbol{w}}) + \bar{\boldsymbol{w}} \cdot \sum_{i=1}^{t} \eta_i \partial \ell(\bar{\boldsymbol{w}}_i, \boldsymbol{x}_i, y_i)$$

where $h(\bar{\boldsymbol{w}})$ is the regularizer and $\eta_i > 0$ are a set of parameters. Intuitively, this amounts to solving at each step a problem similar to (4.3), the difference being that the loss function has been replaced by its subgradient. The linearization of the loss function through the subgradient provides an efficient closed formula update and allows to prove regret bounds (Shalev-Shwartz, 2007; Kakade *et al.*, 2009). The solution of the above minimization problem gives an update of the form

$$\bar{\boldsymbol{w}}_{t+1} = \nabla h^* \left(-\sum_{i=1}^{t} \eta_i \partial \ell(\bar{\boldsymbol{w}}_i, \boldsymbol{x}_i, y_i) \right)$$

where h^* is the Fenchel conjugate of h. The pseudo code of the FTRL framework is illustrated in Algorithm 3.

Next, we use the FTRL framework and propose an online algorithm for solving the l_p norm MKL problem. We prove that the cumulative number of mistakes made on any sequence of T observations is roughly equal to the optimum value of the MKL problem (4.3).

4.3.2 Algorithm and Analysis

The design and analysis of the OM-2 algorithm is based on the framework and machinery developed in (Kakade *et al.*, 2009). It is similar to the l_p norm matrix Perceptron of (Cavallanti *et al.*, 2008), but it overcomes the disadvantage of using the same kernel on each feature. The regularizer of our online MKL problem is defined by $h(\bar{\boldsymbol{w}}) = \frac{q}{2}\|\bar{\boldsymbol{w}}\|^2_{2,p}$. It can be verified that

$$\begin{aligned} h^*(\bar{\boldsymbol{\theta}}) &= \frac{1}{2q}\|\bar{\boldsymbol{\theta}}\|^2_{2,q} \\ \nabla h^*(\boldsymbol{\theta}^j) &= \frac{1}{q}\left(\frac{\|\boldsymbol{\theta}^j\|_2}{\|\bar{\boldsymbol{\theta}}\|_{2,q}}\right)^{q-2} \boldsymbol{\theta}^j, \quad \forall j = 1, \dots, F \ , \end{aligned} \tag{4.4}$$

where $\bar{\boldsymbol{\theta}}$ is the dual weight of $\bar{\boldsymbol{w}}$, and p and q are dual coefficients, which satisfy $1/p + 1/q = 1$. Hence, using the equation (4.4) within the FTRL framework, we obtain the designed algorithm. The pseudocode of the OM-2 algorithm is given in Algorithm 4). As in the mirror descent algorithm, two sets of weights are maintained, a primal one $\bar{\boldsymbol{w}}_t$ and a dual one $\bar{\boldsymbol{\theta}}_t$.

In the following, we will first state the following theorem which bounds the number of updates of the OM-2 algorithm.

Algorithm 4 OM-2

```
Input: $q$
Initialize: $\bar{\boldsymbol{\theta}}_1 = \mathbf{0}, \bar{\boldsymbol{w}}_1 = \mathbf{0}$
for $t = 1, 2, \ldots, T$ do
    Receive new instance $\boldsymbol{x}_t$
    Predict $\hat{y}_t$
    Receive label $y_t$
    $\bar{\boldsymbol{z}}_t = \partial \ell\left(\bar{\boldsymbol{w}}_t, \boldsymbol{x}_t, y_t\right)$
    $\bar{\boldsymbol{\theta}}_{t+1} = \bar{\boldsymbol{\theta}}_t - \eta_t \bar{\boldsymbol{z}}_t$
    $\boldsymbol{w}_{t+1}^j = \frac{1}{q}\left(\frac{\|\boldsymbol{\theta}_{t+1}^j\|_2}{\|\bar{\boldsymbol{\theta}}_{t+1}\|_{2,q}}\right)^{q-2} \boldsymbol{\theta}_{t+1}^j, \quad \forall j = 1, \cdots, F$
end for
```

Lemma 4.1. *Let* $(\boldsymbol{x}_1, y_1), \ldots, (\boldsymbol{x}_T, y_T)$ *be a sequence of examples where* $\boldsymbol{x}_t \in \mathbb{X}$, $y_t \in \mathbb{Y}$. *Suppose that the loss function* ℓ *satisfies the following properties*

- $\|\partial \ell\left(\bar{\boldsymbol{w}}, \boldsymbol{x}, y\right)\|_{2,q} \leq L,\ \forall\ \bar{\boldsymbol{w}} \in \mathbb{S},\ \boldsymbol{x}_t \in \mathbb{X},\ y_t \in \mathbb{Y}, L \in \mathbb{R}$;
- $\ell(\bar{\boldsymbol{u}}, \boldsymbol{x}, y) \geq 1 + \bar{\boldsymbol{u}} \cdot \partial \ell(\bar{\boldsymbol{w}}, \boldsymbol{x}, y),\ \forall\ \bar{\boldsymbol{u}} \in \mathbb{S},\ \bar{\boldsymbol{w}} \in \mathbb{S} : \ell(\bar{\boldsymbol{w}}, \boldsymbol{x}, y) > 0,\ \boldsymbol{x} \in \mathbb{X},\ y \in \mathbb{Y}$;
- $\bar{\boldsymbol{w}} \cdot \partial \ell(\bar{\boldsymbol{w}}, \boldsymbol{x}, y) \geq -1,\ \forall\ \bar{\boldsymbol{w}} \in \mathbb{S},\ \boldsymbol{x} \in \mathbb{X},\ y \in \mathbb{Y}$.

Then, for any $\eta_t > 0$ *and any* $\bar{\boldsymbol{u}}$, *Algorithm 4 satisfies*

$$\begin{aligned}\sum_{t=1}^T \eta_t &\leq \sum_{t=1}^T \eta_t \ell(\bar{\boldsymbol{u}}, \boldsymbol{x}_t, y_t) + \|\bar{\boldsymbol{u}}\|_{2,p} \sqrt{q \sum_{t=1}^T \left(\eta_t^2 \|\bar{\boldsymbol{z}}_t\|_{2,q}^2 - 2\eta_t \bar{\boldsymbol{w}}_t \cdot \bar{\boldsymbol{z}}_t\right)} \\ &\leq \sum_{t=1}^T \eta_t \ell(\bar{\boldsymbol{u}}, \boldsymbol{x}_t, y_t) + \|\bar{\boldsymbol{u}}\|_{2,p} \sqrt{q \sum_{t=1}^T \left(\eta_t^2 L^2 + 2\eta_t\right)}\ ,\end{aligned}$$

where $\bar{\boldsymbol{z}}_t = \partial \ell\left(\bar{\boldsymbol{w}}_t, \boldsymbol{x}_t, y_t\right)$.

This Lemma can appear difficult to interpret, but it gives us some intuitions about the total amount of updates that Algorithm 4 will perform on any sequence of examples. It is straightforward to use the lemma to get mistake bounds for the OM-2 algorithm with different loss functions. This lemma and the pseudocode in Algorithm 4 allows us to design fast and efficient online MKL algorithms for a while range of convex loss.

The hinge loss ℓ^{HL} and multiclass loss ℓ^{MC} satisfy the conditions of the lemma. If we consider a normalized kernel, i.e. $\|\phi^j(\boldsymbol{x}_t, y_t)\|_2 \leq 1, \forall j = 1, \cdots, F,\ t = 1, \cdots, N$, we have that $L \leq F^{\frac{1}{q}}$ for ℓ^{HL} and $L \leq \sqrt{2} F^{\frac{1}{q}}$ for ℓ^{MC}.

For multiclass classification with ℓ^{MC}, we first propose to set η_t using the following update rules

$$\eta_t = \begin{cases} \min\left\{1 + \frac{2\bar{\boldsymbol{w}}_t \cdot \bar{\boldsymbol{z}}_t}{\|\bar{\boldsymbol{z}}_t\|^2_{2,q}}, 1\right\} & \text{if } \ell(\bar{\boldsymbol{w}}_t, \boldsymbol{x}_t, y_t) > 0 \\ 0 & \text{otherwise} \end{cases} . \tag{4.5}$$

Algorithm 4 updates the weights $\bar{\boldsymbol{w}}_t$ each time the loss is greater zero. This aggressive update strategy improves the theoretical performance of the algorithm. With this update rule, it is possible to prove a mistake bound for the multiclass classification version of the Algorithm 4. We say that the algorithm makes a “mistake” each time the prediction $\hat{y}$ is different from the true class y_t.

Theorem 4.1. *Let $(\boldsymbol{x}_1, y_1), \ldots, (\boldsymbol{x}_T, y_T)$ be a sequence of examples where $\boldsymbol{x}_t \in \mathbb{X}$, $y \in \mathbb{Y}$. If the update rule of (4.5) is used to set η_t, and assume the multiclass loss function ℓ^{MC} satisfies $\|\partial \ell^{MC}(\bar{\boldsymbol{w}}, \boldsymbol{x}, y)\|_{2,q} \leq L$. Denote separately by $\mathcal{M}$ and $\mathcal{I}$ the set of rounds in which there is a mistake and the set of rounds in which there is no mistake but there is an update, and by M and I its cardinality. Then, for any $\bar{\boldsymbol{u}}$, Algorithm 4 has the following bound on the maximum number of mistakes M,*

$$M \leq C + S + \sqrt{SC} - \sum_{t \in \mathcal{I}} \eta_t$$

where $C = qL^2 \|\bar{\boldsymbol{u}}\|^2_{2,p}$ and $S = \sum_{t=1}^{T} \ell^{MC}(\bar{\boldsymbol{u}}, \boldsymbol{x}_t, y_t)$.

Apart from the negative term $-\sum_{t \in I} \eta_t$, the bound has the standard form of mistake bounds for online learning multiclass algorithms that only update on mistakes (Fink *et al.*, 2006). The presence of this negative term theoretically motivates the use of updates in margin error rounds. Note also that when the problem is linearly separable, the algorithm converges to a solution that has training error equal to zero. Moreover, when p goes to 1, the term C in the bound has a strongly sublinear dependence on the number of kernels, but unfortunately q goes to infinity. Similarly to (Gentile, 2003), we trade-off these two terms to obtain a logarithmic dependence on the number of kernels. In particular, we obtain the following corollary.

Corollary 4.1. *Under the hypotheses of Theorem 4.1 and consider a normalized kernel such that $\|\phi^j(\boldsymbol{x}_t, y_t)\|_2 \leq 1$, if $p = \frac{2\ln(F)}{2\ln(F)-1}$ and the problem is linear separable, then for any $\bar{\boldsymbol{u}}$, the number* *of mistakes M of Algorithm 4 is less than,*

$$M \leq C + S + \sqrt{SC} - \sum_{t \in \mathcal{I}} \eta_t \ ,$$

where $C = 4e \ln(F) \|\bar{\boldsymbol{u}}\|_{2,1}^2$, and e is the Euler's number.

This corollary tells us that, under the hypothesis that only one kernel is useful for the classification task at hand while the other $F-1$ kernels are irrelevant, with the optimal tuning of p the price we pay in the bound for not knowing which kernel is the right one is just logarithmic in their number F.

4.3.3 Implementation

The training time of OM-2 depends on the complexity of each step. This in turn is dominated by the calculation of the the prediction (line 5) and gradient of the loss (line 7). For example, for the multiclass hinge loss ℓ^{MC}, the time required to execute line 5 has complexity $\mathcal{O}(S_t F M)$, where S_t is the number of updates made before the t-th iteration.

Following (Shalev-Shwartz *et al.*, 2007, Section 4), it is possible to use Mercer kernels without introducing explicitly the dual formulation of the optimization problem. In both algorithms, $\bar{\boldsymbol{\theta}}_{t+1}$ can be written as a weighted linear summation of $\bar{\phi}(\boldsymbol{x}_t, \cdot)$. For example, when using the multiclass loss function ℓ^{MC}, we have that $\bar{\boldsymbol{\theta}}_{t+1} = -\sum_t \eta_t \bar{\boldsymbol{z}}_t = \sum_t \eta_t (\bar{\phi}(\boldsymbol{x}_t, y_t) - \bar{\phi}(\boldsymbol{x}_t, \hat{y}_t))$. Therefore, the algorithm can easily store η_t, y_t, $\hat{y}_t$, and $\boldsymbol{x}_t$ instead of storing $\bar{\boldsymbol{\theta}}_t$. Observing line 9 in Algorithm 4, we have that at each round, $\boldsymbol{w}_{t+1}^j$ is proportional to $\boldsymbol{\theta}_{t+1}^j$, which can be written as $\boldsymbol{w}_{t+1}^j = \alpha_t^j \boldsymbol{\theta}_{t+1}^j$, with $\alpha_t^j \in \mathbb{R}$. Hence $\bar{\boldsymbol{w}}_{t+1}$ can also be represented using $\alpha_t^j \eta_t$, y_t, $\hat{y}_t$ and $\boldsymbol{x}_t$. In prediction the dot product between $\bar{\boldsymbol{w}}_t$ and $\bar{\phi}(\boldsymbol{x}_t, \cdot)$ can be expressed as a sum of terms $\bar{\boldsymbol{w}}_t^j \cdot \phi^j(\boldsymbol{x}_t, \cdot)$, that can be calculated using the definition of kernel.

The l_2 norms of $\boldsymbol{\theta}_{t+1}^j$ can be calculated in an efficient incremental way as

$$\|\boldsymbol{\theta}_{t+1}^j\|_2^2 = \|\boldsymbol{\theta}_t^j\|_2^2 - 2\eta_t \boldsymbol{\theta}_t^j \cdot \boldsymbol{z}_t^j + \eta_t^2 \|\boldsymbol{z}_t^j\|_2^2 \ .$$

4.3.4 Experiments

In this section we present an experimental evaluation of our algorithm on two publicly available databases: the Caltech-101 dataset and the IDOL2 dataset. Since both dataset are multiclass, we use the multiclass loss ℓ^{MC}. Notice that it is much harder to be optimized than the binary hinge loss ℓ^{HL}, especially in the

MKL setting. We first carry out two toy experiments: one on a synthetic data which shows that it is more appropriate to use a multiclass loss instead of dividing the multiclass classification problem into several binary subproblems, another on subsets from Caltech-101 and IDOL2 to study the behavior of the OM-2 algorithm w.r.t. different values of p. Then we conduct experiments on the same two datasets using the full dataset. In the later experiments, we compared the performance of OM-2 to the OMCL algorithm, to the Passive-Aggressive algorithm (PA-I) (Crammer *et al.*, 2006) using the average kernel and the single best feature, and to a batch MKL algorithm, SILP (Sonnenburg *et al.*, 2006) [1]. For the experiment on Caltech-101, we also compare against the LP-β algorithm (Gehler and Nowozin, 2009b) which is a batch algorithm and achieves the highest performance on the dataset under the same step. Since SILP is binary, we use the 1-vs-All strategy for their multiple classes extension (1vA-MKL). We determined all of our online learning parameters via cross-validation. In all of our experiments, the parameter p is chosen from the set $\{1.01, 1.05, 1.10, 1.25, 1.50, 1.75, 2\}$.

To evaluate the performance, we reported the average online training error and the generalization performance on a separate test set. However, it is known that the generalization performance of an online algorithm trained with only one pass (epoch) is typically inferior to batch algorithms, especially when the number of training instances is small. Hence we cycled through the training samples several times (epochs), and we reported the performance on a separate test set as a function of the number of epochs. Experiments show that OM-2 achieves better performance than the other online learning methods, and comparable performance as the batch MKL method at much lower computational cost.

Multiclass synthetic data

Multiclass problems are often decomposed into several binary sub-problems using methods like 1-vs-All. However, solving the multiclass learning problem jointly using a multiclass loss can yield much sparser solutions. Intuitively, when a l_1-norm is used to impose sparsity in the domain of kernels, different subsets of kernels can be selected for the different binary classification sub-problems. Therefore, the combined multiclass classifier might not obtain the desired properties of sparsity. Moreover, the confidence outputs of the binary classifiers may not lie in the same range, so it is not clear if the winner-takes-all hypothesis

---

[1] The SHOGUN-0.9.2 toolbox is available at `http://www.shogun-toolbox.org`, which is implemented in C++. To the best of our knowledge, SILP is the most efficient MKL implementation publicly available. Although SimpleMKL (Rakotomamonjy *et al.*, 2008) is found to be more efficient, a good implementation for large problems is not available. Despite that the SHOGUN-0.9.2 toolbox also includes the implementation of MC-MKL (Zien and Ong, 2007) using the same multiclass loss as we do, this approach is computationally infeasible for the experiments presented here due to performance issues.

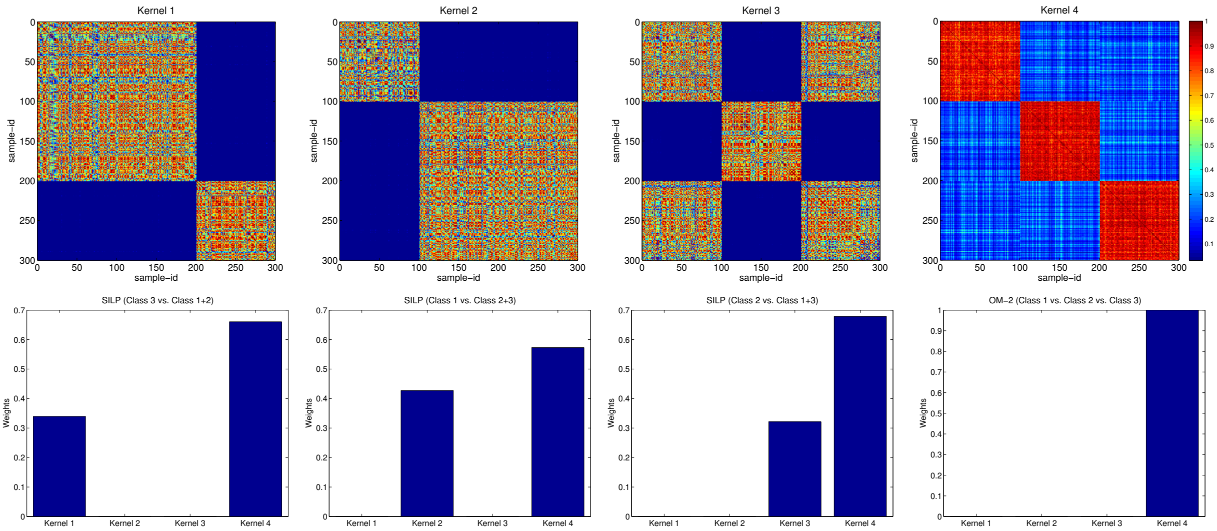


Figure 4.1. (top) Kernel matrices of the 3-classes synthetic experiments correspond to 4 different features. Sample 1–100, 101–200 and 201–300 are from class 1, 2 and 3 respectively. (bottom) Corresponding kernel combination weights, normalized to have sum equal to 1, obtained by SILP (binary) and by OM-2 (last figure).

is the correct approach for combing them.

To prove our points, we have generated a 3-classes classification problem consisting of 300 samples, with 100 samples for each class. There are in total 4 different features, the kernel matrices corresponding to them are shown in Figure 4.1 (top). These features are generated in a way that Kernels 1–3 are useful only for distinguishing one class (class 3, class 1 and class 2, respectively) from the other two, while Kernel 4 can separate all the 3 classes. The corresponding kernel combination weights obtained by the SILP algorithm using the 1-vs-All extension and our multiclass OM-2 are shown in Figure 4.1 (bottom). We set the parameter p of OM-2 algorithm to 1.01 in order to get a sparse solution. It can be observed that each of the binary SILP classifiers pick two kernels. OM-2 selects only the 4th kernel, achieving a much sparser solution.

Behavior w.r.t the value of p

This experiment aims at showing the behavior of OM-2 for varying values of p. We consider $p \in (1, 2]$, and train OM-2 on the KTH-IDOL2 and Caltech-101. The results for the two datasets are shown in Figure 4.2. For the IDOL2 dataset (Figure 4.2 (left)), the best performance is achieved when p is large, which corresponds to give all the kernels similar weights in the decision. On the contrary, a sparse solution achieves lower accuracy. It indicates that all the kernels carry discriminative information, and

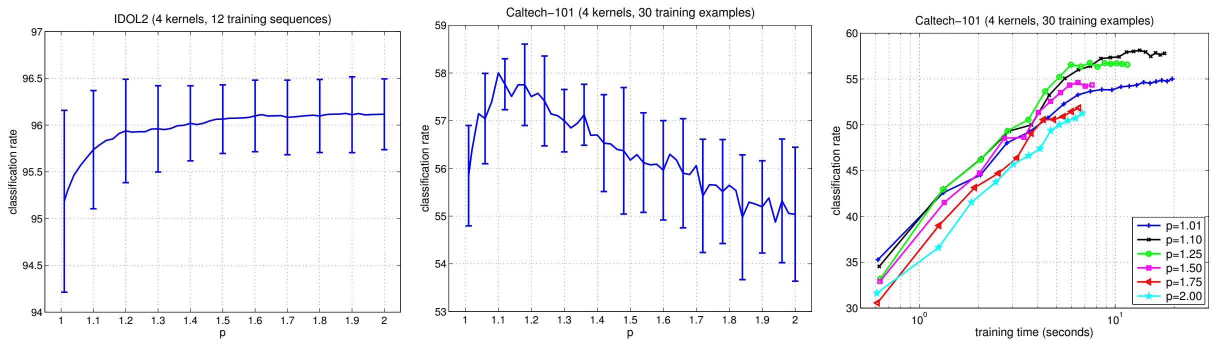


Figure 4.2. Behaviors of OM-2 w.r.t. different values of p: (left) the effect of p on the IDOL2 dataset and (middle) the Caltech-101 dataset using 4 PHOG (Bosch *et al.*, 2007) kernels; (right) running time for different values of p on the Caltech-101 dataset.

excluding some of them can decrease the performance. For the Caltech-101 dataset (Figure 4.2 (middle, right)), following (Gehler and Nowozin, 2009b), we consider four PHOG kernels computed at different spatial pyramid level. It can be observed that by adjusting p it is possible to improve the performance – sparser solutions (i.e. when p tends to 1) achieve higher accuracy compared to non-sparse solutions (when p tends to 2). However, the optimal p here is 1.10. In other words the optimal performance is achieved for a setting of p different from 1 or 2, fully justifying the presence of this parameter. It could be explained as although some of the kernels may contain redundant information, since they were generated using the same type of feature descriptor, all of them may be informative for classification. Therefore, imposing full sparsity does not help to increase the performance. Furthermore, Figure 4.2 (right) shows the running time of OM-2 using the same four kernels, with varying values of p. It can be observed that OM-2 converges faster when p is large.

Object categorization: Caltech-101

In this experiment, we use all the 102 classes with 30 training images per category and all the 39 kernels as described in Appendix A. The results are averaged over 5 different splits. For each split, the experiments are performed over 10 different permutations of the training images. Figure 4.3 (left) shows the average online training error rate using different online learning algorithms as a function of the number of training examples. OM-2 converges faster compared to the other online learning baselines. The OMCL algorithm suffers from the hardness of the task, and has a larger training error compared to the other online algorithms. Figure 4.3 (Right) reports the generalization performance obtained using different online learning algorithms, and 1vA-MKL and LP-β algorithm. It can be observed from the plots that the OM-

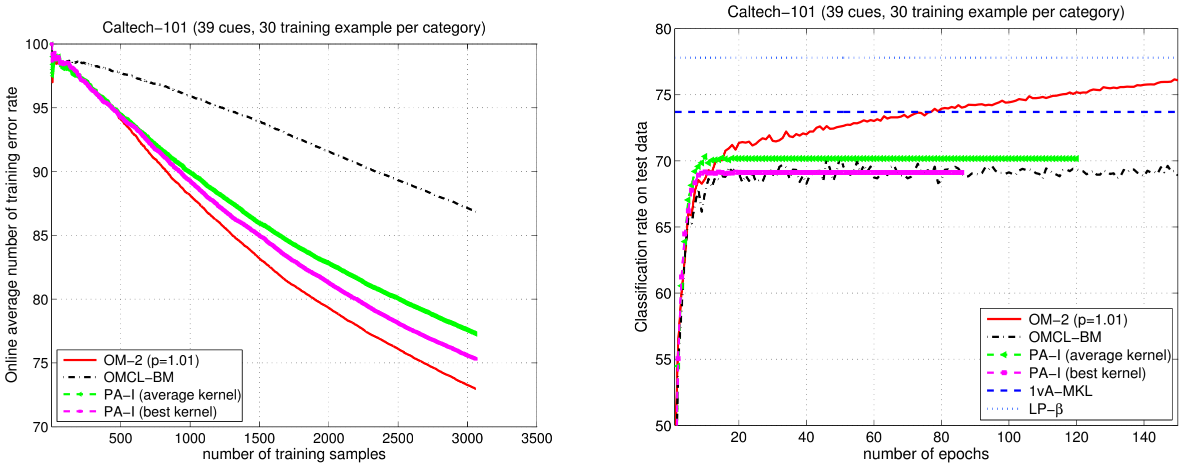


Figure 4.3. Performance of different learning algorithms on the Caltech-101 dataset: (left) average online training error rate as a function of the number of training examples; (right) classification rate on the test set.

2 algorithm achieves better performance on both training and test phase compared to the other online learning algorithms. The best results are obtained when p is small (1.01 in our setup). This is probably because among these 39 kernels (computed from eight different image descriptors) many of them were redundant. Thus a sparse solution is favored. After about 80 epochs, the OM-2 achieves comparable results as the batch MKL algorithm at a relatively low computational time. Although our MATLAB implementation is not optimized for speed, training on the Caltech-101 dataset using 30 examples per class and 39 kernels takes about 15 mins[2] (150 epochs, 3060×150 iterations). For both 1vA-MKL and LP-β the training time is more than 2 hours. The performance advantage of OM-2 over 1vA-MKL is due to the fact that OM-2 is based on a native multiclass formulation (also used by LP-β), while SILP solves the multiclass problem by decomposing it into multiple independent binary classification tasks. The results support our claim that multiclass loss function is more suitable for this type of problem. As the sparse MKL algorithm may choose different subset of kernels in different independent binary classification tasks, this may cause the outputs of different 1-vs-All binary classifiers to lie in a different range and introduces a bias on some classes during the final decision process. Thanks to the online learning framework, we can solve the new MKL formulation using the multiclass loss function efficiently. The performance difference between OM-2 and LP-β could due to the fact that LP-β is a batch algorithm.

---

[2] Notice that this time is measured using precomputed kernel matrix as inputs. In practice, if not all the data points are available from the beginning, the kernel value could be computed "on the fly" and the total computation time of several epochs could be reduced using methods like kernel caching.

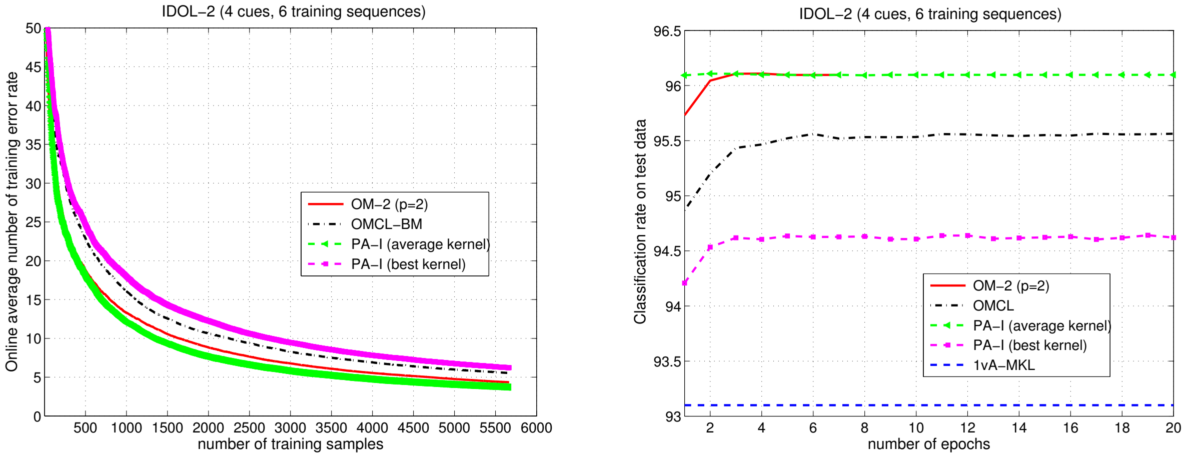


Figure 4.4. Performance of different learning algorithms on the IDOL2 dataset: (left) average online training error rate as a function of the number of training examples; (right) classification rate on the test set.

Place recognition: IDOL-2

For these experiments, we used the same setup described in Section 3.4, with 4 different kind of features which results in 4 kernels in total. We repeated the experiments with 12 different permutations of training and test sets. Figure 4.4 reports the average online training and recognition error rate on the test set. In contrast to the previous experiments, here the best performance is obtained at $p = 2$, which means that there are no redundant features and all of them are discriminative for the given task. OM-2 and PA-I using average kernel achieve better performance compared to the other online learning algorithms. Surprisingly, the performance of batch MKL is worse than all the other online feature combination method, including the performance of the single best feature. Hence, in agreement with recent findings (Cortes *et al.*, 2009b), promoting sparsity hurts performance when the problem is not sparse at all.

4.4 Batch MKL: the OBSCURE algorithm

Although Algorithm OM-2 solves an online variant of the l_p norm of the MKL problem (4.3), obtaining a solution close to the optimum value of the original problem, an algorithm which can find the exact optimal solution with theoretical guarantee is still desirable. In this section, we introduce a new optimization algorithm to solve efficiently the MKL problem (4.3), with a guaranteed convergence rate to the optimal solution. We show that the presence of a large number of kernels helps the optimization process instead of hindering it, obtaining, theoretically and practically, a faster convergence rate with more kernels.

Moreover, the new algorithm has a training time that depends linearly on the number of training examples. Like the OM-2 algorithm, it has a convergence rate sub-linear in the number of features/kernels used, when a sparse solution is favored.

The new algorithm is composed of two stages. The first one is an (online) initialization procedure that determines quickly the region of the space where the optimal solution lives, using the OM-2 algorithm proposed in the previous section. The second stage refines the solution found by the first stage and obtains the batch optimal solution, using a stochastic gradient descent algorithm. We call this algorithm OBSCURE, Online-Batch Strongly Convex mUlti keRnel lEarning.

In the rest of this section, Section 4.4.1 presents the algorithm and Section 4.4.2 discusses the implementation details, while Section 4.4.3 and Section 4.4.4 show our theoretical guarantees. Section 4.4.5 describes our experimental findings.

4.4.1 Algorithm

Our basic optimization tool is the framework developed in Shalev-Shwartz and Singer (2007); Shalev-Shwartz *et al.* (2007). It is a general framework to design and analyze stochastic sub-gradient descent algorithms for any strongly convex function. At each step the algorithm takes a random sample of the training set and calculates a sub-gradient of the objective function evaluated on the sample. Then it performs a sub-gradient descent step with decreasing learning rate, followed by a projection of the solution inside the space where the solution lives. The algorithm Pegasos, based on this framework, is among the state-of-art solvers for linear SVM (Shalev-Shwartz *et al.*, 2007; Shalev-Shwartz and Srebro, 2008). Given that the l_p norm is $1/q$-strongly convex, we could use this framework to design an efficient MKL algorithm. It would inherit all the properties of Pegasos (Shalev-Shwartz *et al.*, 2007; Shalev-Shwartz and Srebro, 2008). In particular the convergence rate, and hence the training time, would be proportional to $\frac{q}{\lambda}$.

Although in general this convergence rate can be quite good, it becomes slow when λ is small and/or p is close to 1. Moreover it is common knowledge that in many real-world problems (*e.g.,* visual learning tasks) the best setting for λ is very small, or equivalently C is very big (the order of $10^2 - 10^3$). Notice that this is a general problem. The same problem also exists in the other SVM optimization algorithms such as SMO and similar approaches, as their training time also depends on the value of the parameter C (Hush *et al.*, 2006). Do *et al.* (2009) proposed a variation of the Pegasos algorithm called proximal projected sub-gradient descent. This formulation has a better convergence rate for small values of λ, while retaining the fast convergence rate for big values of λ. A drawback is that the algorithm needs to know in advance an upper bound on the norm of the optimal solution. In Do *et al.* (2009) the authors proposed an algorithm that estimates this bound while training, but it gives a speed-up only when the norm of the optimal solution $\bar{\boldsymbol{w}}^*$ is small. This is not the case in most of the MKL problems for categorization tasks.

Our OBSCURE algorithm takes the best of the two solutions. We first extend the framework of Do *et al.* (2009) to the generic non-Euclidean norms, to use it with the l_p norm. Then we solve the problem of the upper bound of the norm of the optimal solution using a new online algorithm. This is designed to take advantage of the characteristics of the MKL task and to quickly converge to a solution close to the optimal one. Hence OBSCURE is composed by two stages: the online step and the batch step.

In the online stage, we used the OM-2 algorithm to quickly estimate the region of the space where the optimal solution lives. Since the goal for this step is only to roughly estimate the optimal region, we define a simple updating rule

$$\eta_t = \begin{cases} \eta_0 & \text{if } \ell(\bar{\boldsymbol{w}}_t, \boldsymbol{x}_t, y_t) > 0 \\ 0 & \text{otherwise} \end{cases} , \tag{4.6}$$

where $\eta_0 > 0$ is a predefined learning rate. We show that this update rule not only simplifies the computational complexity, but also makes the analysis of the algorithm easy. We could stop the online stage after a fixed number of iteration, or until it converges to a solution which has null loss on each training sample (if the examples are linear separable), then pass the founded solution, denoted by $\bar{\boldsymbol{\theta}}_{\mathrm{O}}$ and $\bar{\boldsymbol{w}}_{\mathrm{O}}$, to the second stage.

The batch stage (Algorithm 5), starts from the approximated solution found by the online stage, and by exploiting the information on the estimated region, it uses a stochastic proximal projected sub-gradient descent algorithm.

4.4.2 Efficient Implementation

The training time of OBSCURE is proportional to the number of steps required to converge to the optimal solution, that will be bounded in Theorem 4.2, multiplied by the complexity of each step. This in turn is dominated by the calculation of the gradient of the loss (line 5 in Algorithms 5), that, for example, for

Algorithm 5 OBSCURE Batch

```
Input: $q$, $\lambda$, $\bar{\boldsymbol{\theta}}_1 = \bar{\boldsymbol{\theta}}_{\mathrm{O}}$, $\bar{\boldsymbol{w}}_1 = \bar{\boldsymbol{w}}_{\mathrm{O}}$,
Initialize: $s_0 = 0$, $R = \sqrt{\|\bar{\boldsymbol{w}}_{\mathrm{O}}\|_{2,p}^2 + \frac{2}{\lambda N}\sum_{i=1}^N \ell(\bar{\boldsymbol{w}}_{\mathrm{O}}, \boldsymbol{x}_i, y_i)}$
for $t = 1, 2, \ldots, T$ do
    Sample at random $(\boldsymbol{x}_t, y_t)$
    $\bar{\boldsymbol{z}}_t = \partial\ell(\bar{\boldsymbol{w}}_t, \boldsymbol{x}_t, y_t)$
    $d_t = \lambda t + s_{t-1}$
    $s_t = s_{t-1} + 0.5\left(\sqrt{d_t^2 + q\frac{(\frac{\lambda}{q}\|\bar{\boldsymbol{\theta}}_t\|_{2,q} + \|\bar{\boldsymbol{z}}_t\|_{2,q})^2}{R^2}} - d_t\right)$
    $\eta_t = \frac{q}{\lambda t + s_t}$
    $\bar{\boldsymbol{\theta}}_{t+\frac{1}{2}} = (1 - \frac{\lambda\eta_t}{q})\bar{\boldsymbol{\theta}}_t - \eta_t\bar{\boldsymbol{z}}_t$
    $\bar{\boldsymbol{\theta}}_{t+1} = \min\left(1, \frac{qR}{\|\bar{\boldsymbol{\theta}}_{t+\frac{1}{2}}\|_{2,q}}\right)\bar{\boldsymbol{\theta}}_{t+\frac{1}{2}}$
    $\boldsymbol{w}_{t+1}^j = \frac{1}{q}\left(\frac{\|\boldsymbol{\theta}_{t+1}^j\|_2}{\|\bar{\boldsymbol{\theta}}_{t+1}\|_{2,q}}\right)^{q-2}\boldsymbol{\theta}_{t+1}^j, \ \forall j = 1, \cdots, F$
end for
```

the multiclass hinge loss ℓ^{MC} has complexity $\mathcal{O}(NFM)$. Note that this complexity is common to any other similar algorithm, and it can be reduced using methods like kernel caching (Chang and Lin, 2001). We could use the same technique described in Section 4.3.3 to represent Mercer kernels in our primal optimization process.

Another important speed-up can be obtained by considering the nature of the updates of the batch stage. If the optimal solution has a loss equal to zero or close to it, when the algorithm is close to convergence most of the updates will consist just of a scaling. Hence it is possible to cumulate the scalings in a variable, to perform the scaling of the coefficients just before an additive update must be performed, and to take it into account for each prediction. Moreover, when using the multiclass loss ℓ^{MC}, each update touches only two classes at a time, so to minimize the number of scalings we can keep a vector of scaling coefficients, one for each class, instead of a single number.

4.4.3 Analysis of the OBSCURE Batch Stage

We now show the theorems that give a theoretical guarantee on the convergence rate of OBSCURE to the optimal solution of (4.3).

The following lemma contains useful properties to prove the performance guarantees of Algorithm 5 and 6.

Lemma 4.2. *Let* $B \in \mathbb{R}^+$, *define* $S = \{\bar{\boldsymbol{w}} : \|\bar{\boldsymbol{w}}\|_{2,p} \le B\}$. *Let* $h(\bar{\boldsymbol{w}}) : S \to \mathbb{R}$ *defined as* $\frac{q}{2}\|\bar{\boldsymbol{w}}\|_{2,p}^2$, *define also* $\mathrm{Proj}(\bar{\boldsymbol{w}}, B) = \min\left(1, \frac{B}{\|\bar{\boldsymbol{w}}\|_{2,p}}\right)\bar{\boldsymbol{w}}$, *then*

Algorithm 6 Proximal projected sub-gradient descent

```
Input: $R$, $\sigma$, $\boldsymbol{w}_1 \in \mathbb{S}$
Initialize: $s_0 = 0$
for $t = 1, 2, \ldots, T$ do
    Receive $g_t$
    $\boldsymbol{z}_t = \partial g_t(\boldsymbol{w}_t)$
    $s_t = s_{t-1} + \frac{\sqrt{(\alpha\sigma t + s_{t-1})^2 + \frac{\alpha L_t}{R^2}} - \alpha\sigma t - s_{t-1}}{2}$
    $\eta_t = \frac{1}{\sigma t + \frac{s_t}{\alpha}}$
    $\boldsymbol{w}_{t+1} = \nabla h^*(\nabla h(\boldsymbol{w}_t) - \eta_t \boldsymbol{z}_t)$
end for
```

1. $B = +\infty \;\Rightarrow\; h^*(\bar{\boldsymbol{\theta}}) = \frac{1}{2q}\|\bar{\boldsymbol{\theta}}\|_{2,q}^2$

2. $\nabla h(\bar{\boldsymbol{w}}) = q\left[\left(\frac{\|\boldsymbol{w}^j\|_2}{\|\bar{\boldsymbol{w}}\|_{2,p}}\right)^{p-2}\boldsymbol{w}^j\right]_1^F, \;\forall \bar{\boldsymbol{w}} \in \mathbb{S}$

3. $\nabla h^*(\bar{\boldsymbol{\theta}}) = \mathrm{Proj}\left(\frac{1}{q}\left[\left(\frac{\|\boldsymbol{\theta}^j\|_2}{\|\bar{\boldsymbol{\theta}}\|_{2,q}}\right)^{q-2}\boldsymbol{\theta}^j\right]_1^F, B\right)$

4. $\|\bar{\boldsymbol{w}}\|_{2,p} = \frac{1}{q}\|\nabla h(\bar{\boldsymbol{w}})\|_{2,q}, \;\forall \bar{\boldsymbol{w}} \in \mathbb{S}$

We first introduce Algorithm 6, that forms the basis for Algorithm 5, and a lemma that bounds its performance, that is a generalization of (Do *et al.*, 2009, Theorem 1) to general norms, using the framework in Shalev-Shwartz and Singer (2007).

Lemma 4.3. *Let* $h(\cdot) = \frac{\alpha}{2}\|\cdot\|^2$ *be a 1-strongly convex function w.r.t. a norm* $\|\cdot\|$ *over* $\mathbb{S}$*. Assume that for all* t*,* $g_t(\cdot)$ *is a* σ*-strongly convex function w.r.t.* $h(\cdot)$*, and* $\|\boldsymbol{z}_t\|_* \leq L_t$*. Then for any* $\boldsymbol{u} : \|\boldsymbol{u} - \boldsymbol{w}_t\| \leq 2R$*, and for any sequence of non-negative* $\xi_1, \ldots, \xi_T$*, Algorithm 6 achieves the following bound for all* $T \geq 1$*,*

$$\sum_{t=1}^{T}(g_t(\boldsymbol{w}_t) - g_t(\boldsymbol{u})) \leq \sum_{t=1}^{T}\left[4\xi_t R^2 + \frac{L_t^2}{\sigma t + \frac{\sum_{i=1}^{t}\xi_i}{\alpha}}\right] .$$

With this Lemma we can now design stochastic sub-gradient algorithms. In particular, setting $\|\cdot\|_{2,p}$ as norm, $h(\bar{\boldsymbol{w}}) = \frac{q}{2}\|\bar{\boldsymbol{w}}\|_{2,p}^2$, and $g_t(\bar{\boldsymbol{w}}) = \frac{\lambda}{q}h(\bar{\boldsymbol{w}}) + \ell\left(\bar{\boldsymbol{w}}, \boldsymbol{x}_t, y_t\right)$, we obtain Algorithm 5 that solves the l_p-norm MKL problem (4.3).

In Algorithm 5, the updates are done on the dual variables $\bar{\boldsymbol{\theta}}_t$, in lines 9-10, and transformed into $\bar{\boldsymbol{w}}_t$ in line 11, through a simple scaling. We can now prove the following bound on the convergence rate for Algorithm 5.

Theorem 4.2. *Suppose that* $\|\partial \ell(\bar{\boldsymbol{w}}, \boldsymbol{x}_t, y_t)\|_{2,q} \leq L$ *and* $\|\bar{\boldsymbol{w}}^*\|_{2,p} \leq R$, *where* $\bar{\boldsymbol{w}}^*$ *is the optimal solution of (4.3), that is* $\bar{\boldsymbol{w}}^* = \arg\min_{\bar{\boldsymbol{w}}} f(\bar{\boldsymbol{w}})$. *Let* $1 < p \leq 2$, $\delta \in (0,1)$, *and* $c = \lambda R + L$. *Then with probability at least* $1 - \delta$ *over the choices of the random samples we have that, after* T *iterations of the 2nd stage of the OBSCURE algorithm, the difference between* $f(\bar{\boldsymbol{w}}_T)$ *and* $f(\bar{\boldsymbol{w}}^*)$, *is less than*

$$\frac{c\sqrt{q}\sqrt{1+\log T}}{\delta} \min\left(\frac{c\sqrt{q}\sqrt{1+\log T}}{\lambda T}, \frac{4R}{\sqrt{T}}\right) .$$

The most important thing to note is that the converge rate is independent from the number of samples, as in Pegasos (Shalev-Shwartz *et al.*, 2007), and the relevant quantities on which it depends are λ and q. Given that for most of the losses, each iteration has a linear complexity in the number of samples, as stated in Section 4.4.2, the training time will be linearly proportional to the number of samples.

The parameter R is basically an upper bound on the norm of the optimal solution. In Section 4.4.4 we show how to have a good estimate of R in an efficient way. The theorem first shows that a good estimate of R can speed-up the convergence of the algorithm. In particular if the first term is dominant, the convergence rate is $\mathcal{O}(\frac{q \log T}{\lambda T})$. If the second term is predominant, the convergence rate is $\mathcal{O}(\frac{R\sqrt{q \log T}}{\sqrt{T}})$, so it becomes independent from λ. The algorithm will always optimally interpolate between these two different rates of convergence. Note that R can also be set to the trivial upper bound of ∞. Another important point is that Algorithm 5 can start from any vector, while this is not possible in the Pegasos algorithm, where at the very first iteration the starting vector is multiplied by 0 (Shalev-Shwartz *et al.*, 2007).

As said before, the rate of convergence depends on p, through q. A p close to 1 will result in a sparse solution, with a rate of at most $\mathcal{O}(\frac{R\sqrt{q \log T}}{\sqrt{T}})$. However in the experimental section we show that the best performance is not always given by the most sparse solution.

This theorem and the pseudocode in Algorithm 5 allows us to design fast and efficient MKL algorithms for a wide range of convex losses. Hence, in binary and multiclass classification cases where ℓ^{HL} and ℓ^{MC}, if $p < 2$, the convergence rate has a sublinear dependency on the number of kernels, F, and if the problem is linearly separable it can have a faster convergence rate using more kernels. We will explain this formally in the next section.

4.4.4 Analysis of the OBSCURE Online Stage

In Theorem 4.2 we saw that if we have a good estimate of R, the convergence rate of the algorithm can be much faster. Moreover starting from a *good* solution could speed-up the algorithm even more.

We propose to initialize Algorithm 5 with Algorithm 4 using the update rule defined in (4.6). We can run it just for few iterations and then evaluate its norm and its loss. In Algorithm 5, R is then defined as

$$R := \sqrt{\|\bar{\boldsymbol{w}}_{\mathrm{O}}\|_{2,p}^2 + \frac{2}{\lambda N}\sum_{i=1}^{N} \ell\left(\bar{\boldsymbol{w}}_{\mathrm{O}}, \boldsymbol{x}_i, y_i\right)} \geq \sqrt{\|\bar{\boldsymbol{w}}^*\|_{2,p}^2 + \frac{2}{\lambda N}\sum_{i=1}^{N} \ell\left(\bar{\boldsymbol{w}}^*, \boldsymbol{x}_i, y_i\right)} \geq \|\bar{\boldsymbol{w}}^*\|_{2,p} \ .$$

So at any moment we can stop the algorithm and obtain an upper bound on $\|\bar{\boldsymbol{w}}^*\|_{2,p}$. However if the problem is linearly separable we can prove that the online stages will converge in a finite number of updates. The proof technique is based on the well-known "difference of norms" method, see for example (Gentile, 2003), that we have generalized to take into account algorithms with aggressive updates too.

Theorem 4.3. *Let $(\boldsymbol{x}_1, y_1), \ldots, (\boldsymbol{x}_T, y_T)$ be a sequence of examples where $\boldsymbol{x}_t \in \mathbb{X}$, $y \in \mathbb{Y}$. Denote by $\mathcal{U}$ the set of rounds in which there is an update, and by U its cardinality. Under the hypothesis of Lemma 4.1, for any $\bar{\boldsymbol{u}}$, the number of updates U of the online stage of OBSCURE satisfies*

$$U \leq q(2/\eta_0 + L^2)\|\bar{\boldsymbol{u}}\|_{2,p}^2 + \sum_{t \in \mathcal{U}} \ell(\bar{\boldsymbol{u}}, \boldsymbol{x}_t, y_t) + \|\bar{\boldsymbol{u}}\|_{2,p}\sqrt{q(2/\eta_0 + L^2)}\sqrt{\sum_{t \in \mathcal{U}} \ell(\bar{\boldsymbol{u}}, \boldsymbol{x}_t, y_t)} \ .$$

In particular, if the problem (4.3) is linearly separable by a hyperplane $\bar{\boldsymbol{u}}$, then the algorithm will converge to a solution in a finite number of steps less than $q(2/\eta_0 + L^2)\|\bar{\boldsymbol{u}}\|_{2,p}^2$. In this case the returned value of R will be less than $(2 + \eta_0 L^2)\|\bar{\boldsymbol{u}}\|_{2,p}$.

From the theorem it is clear the role of η_0: a bigger value will speed up the convergence, but it will decrease the quality of the estimate of R. So η_0 governs the trade-off between speed and precision of the first stage.

As noted for Theorem 4.2 when p is close to 1, the dependency on the number of kernels in this theorem is strongly sublinear. Moreover, under the separability assumption, if we increase the number of kernels, we have that $\|\bar{\boldsymbol{u}}\|_{2,p}^2$ cannot increase, and in most of the cases it will decrease. This means that, under the separability assumption, we expect Algorithm 4 to converge to a solution which has null loss on each training sample, in a finite number of steps that is almost independent on F and in some cases even *decreasing* while increasing F. The same consideration holds for the value of R returned by the algorithm, that can decrease when we increase the number of kernels. A smaller value of R will mean a faster convergence of the second stage.

4.4.5 Experiments

In this section, we study the behavior of OBSCURE in terms of classification accuracy, computational efficiency and scalability. We again focus on the multiclass loss ℓ^{MC}. Although our MATLAB implementation is not optimized for speed, it is already possible to observe the advantage of the low runtime complexity. This is particularly evident when training on datasets containing large numbers of categories and lots of training samples. Similar to setup used in the previous section, in all the experiments the parameter p is chosen from the set $\{1.01, 1.05, 1.10, 1.25, 1.50, 1.75, 2\}$. The regularization parameter λ is set through CV, as $\frac{1}{CN}$, where $C \in \{0.1, 1, 10, 100, 1000\}$.

We compare our algorithm with 1vA-MKL, the multiclass MKL (MC-MKL) algorithm (Zien and Ong, 2007), LP-β, and SVM with the unweighted sum of the kernels (Average Kernel). To train the Average Kernel baseline, we use LIBSVM (Chang and Lin, 2001). The cost parameter is selected from the range $C \in \{0.1, 1, 10, 100, 1000\}$ for all the baseline methods. For all the binary classification algorithm, we use the 1-vs-All strategy for their multiple classes extension.

In the following we start by studying the convergence rate of OBSCURE and compare it with the original Pegasos algorithm (Shalev-Shwartz *et al.*, 2007; Shalev-Shwartz and Srebro, 2008). Then we study the behaviors of OBSCURE w.r.t different number of input kernels, as its behavior w.r.t different value of p is analogous to the OM-2 algorithm. Following that we show that OBSCURE achieves state-of-art performance on a challenging image classification task with 102 different classes. Finally we show its scalability w.r.t. the number of training samples.

Comparison of convergence rate with Pegasos algorithm

We have implemented an extended version of the original Pegasos algorithm for the MKL problem of (4.3). We first compare the running time performance between OBSCURE and Pegasos on the Oxford flowers dataset. Their generalization performance on the testing data (Figure 4.5(left)) as well as the value of the objective function (Figure 4.5(right)) are reported. In the same Figure, we also present the

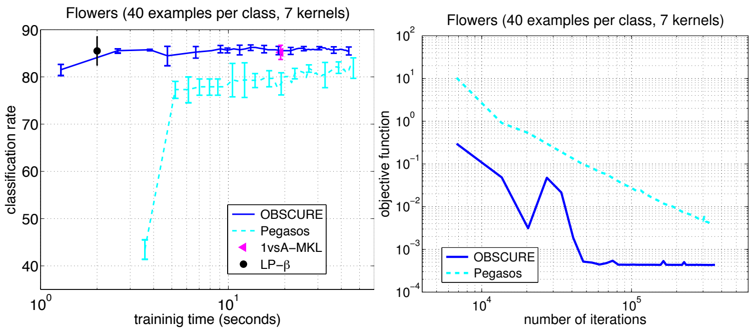


Figure 4.5. Comparison of running time performance on the Oxford flowers dataset between OBSCURE, Pegasos and SILP.

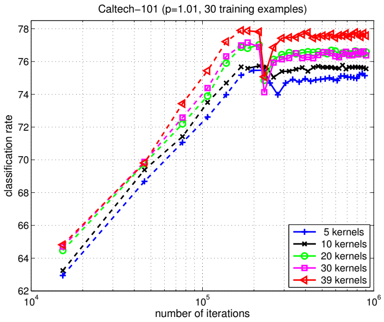


Figure 4.6. Behaviors of OBSCURE on the Caltech-101 w.r.t. different number of kernels randomly sampled from the 39 kernels.

results obtained using SILP. We see that OBSCURE converges much faster compared to Pegasos. This proves that, as stated in Theorem 4.2, OBSCURE has a better convergence rate than Pegasos, as well as faster running time than SILP. All the feature combination methods achieve similar results on this dataset.

Behavior w.r.t. the number of kernels

Figure 4.6 reports the behavior of OBSCURE on Caltech-101 for different numbers of input kernels. The dashed line in the figure corresponds to the results obtained by the first online stage of the OBSCURE algorithm. It shows that OBSCURE has a better converges rate when there are more kernels, as stated in Theorem 4.3. In other words, the algorithm achieves a given accuracy in less iterations when more kernels are given. It can also be observed that the online step of OBSCURE (OM-2) achieves a performance close to the optimal solution in a training time that is order of magnitudes faster (10^1 to 10^3), even if theoretically we cannot guarantee the online stage to reach the global optimum.

Multiclass Image Classification: Caltech-101

In this experiment, we use the Caltech-101 dataset with all the 39 kernels, and the results are shown in Figure 4.7. Figure 4.7 (left) reports the training time for different algorithms. Figure 4.7(right) reports the results obtained using different combination methods for varying size of training samples. The best results for OBSCURE were obtained when p is at the smallest value (1.01). This is probably because among these 39 kernels many were redundant or not discriminative enough. For example, the worst single kernel achieves only an accuracy of $13.5\% \pm 0.6$ when trained using 30 images per category, while the best

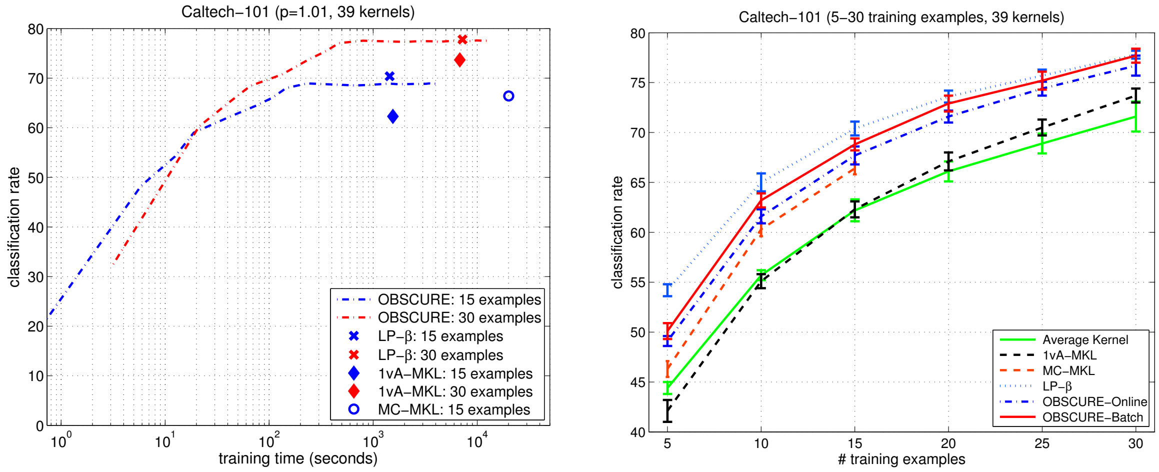


Figure 4.7. Performance comparison on the Caltech-101 dataset using different cue integration methods.

single kernel achieves 69.4%±0.4. Thus, sparser solutions are to be favored. The results again support our claim in Section 4.3.4 that the multiclass loss function is more suitable for this type of problem, as all the methods that use the multiclass loss outperform 1vA-MKL. OBSCURE achieve comparable performance as the state-of-art LP-β algorithm, especially when the number of training example is large (*e.g.,* 30 per class). Note that although our algorithm obtains a solution close to the sparse one, it will never reach a completely sparse solution. This may be one of the reasons for the gap in performance between OBSCURE and LP-β on this dataset. It can also be observed from Figure 4.7 (left) that OBSCURE reaches optimal solution much faster than the other cue combination algorithms which are implemented in C++. MC-MKL is computationally infeasible for 30 samples per category. Its significant gap from OBSCURE seems to indicate that it stops before converging to the optimal solution.

Scalability w.r.t. the number of training samples

In this section, we report the experiments on the MNIST dataset using varying sizes of training samples. Figure 4.8 shows the generalization performance on the test set achieved by OBSCURE over time, for various training size. We see that OBSCURE quickly converges to the best performance, moreover the convergence is faster when more training samples are used, as in Shalev-Shwartz and Srebro (2008). It also shows that the time to reach the optimum is approximately linear in the number of training samples. The performance of the SVM trained using the unweighted sum of the kernels and the best kernel are also plotted. Notice that in the figure we only show the results of up to 20,000 training samples for the

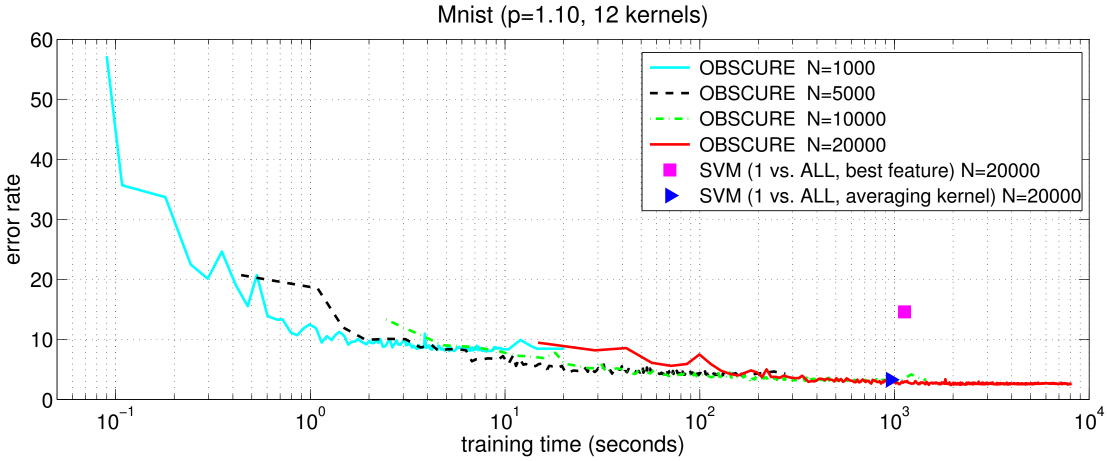


Figure 4.8. The generalization performance of OBSCURE on the MNIST dataset over different sizes of training samples.

sake of comparison, otherwise we could not cache all the 12 kernels in memory. However, by computing the kernel "*on the fly*" we are able to solve the MKL problem using the full 60,000 examples: OBSCURE obtains 1.95% error rate after 10 epochs, which is 0.45% lower compared to the results obtained by OBSCURE with 20,000 training samples after 500 epochs.

4.5 Sparse MKL: the UFO algorithm

The l_p norm MKL formulation allows to decide the level of sparsity of the solution through tuning p. Still this formulation never induces coefficients that are mathematically zero for any $p \neq 1$. It is also interesting to note that the convergence rate of the l_p MKL becomes slower when $p \to 1$. When p tends to 1, q goes to infinity and the strong convexity is lost, resulting in a slower convergence. In other words, it is more difficult and slower to find a sparse solution to the MKL problem. Under some circumstances, *e.g.,* when a large number of kernels are given and only few are informative, sparsity is still desired. In this section, we first propose a novel sparse MKL formulation which still preserve strong convexity property, then an efficient optimization algorithm to solve it.

While designing the algorithm, we look at the same time at the MKL problem from a learning and optimization points of view. Therefore, instead of designing a regularizer and then try to find an efficient method to minimize it, we design the regularizer while keeping the optimization process in mind. In other words, a perfect regularizer is useless if it is impractical to be used. We call this algorithm Ultra Fast Online Multi Kernel Learning, UFO-MKL. The UFO-MKL algorithm gives exact sparse solutions with a tunable level of sparsity, and a convergence rate bound that depends only logarithmically on the number of kernels used, and is independent of the sparsity required. Like the OM-2 algorithm and the OBSCURE algorithm, it is also independent on the particular convex loss function used.

In the rest of this section, we first present the sparse MKL formulation in Section 4.5.1. Then Section 4.5.2 and Section 4.5.3 present the theory and algorithm of UFO-MKL. Finally, Section 4.5.4 reports experiments on several different classification tasks.

4.5.1 Sparse MKL Formulation

Tomioka and Suzuki (2010) have proposed to use an elastic net form of regularization for MKL, that can be written as

$$h(\bar{\boldsymbol{w}}) := C(\frac{\lambda}{2}\|\bar{\boldsymbol{w}}\|_{2,2}^2 + (1-\lambda)\|\bar{\boldsymbol{w}}\|_{2,1}). \tag{4.7}$$

They have justified this form of regularization as a mean to control the degree of sparsity of the solution. In this way the solution has exact mathematical zeros, and the number of zeros can be tuned by changing λ. However, note that there is no particular reason to use the $(2,2)$ group norm, apart from having an easier optimization problem and a way to tune the level of sparsity. Similar considerations hold for the $(2,p)$ group norm used in the l_p norm MKL formulation. In fact, with the l_p norm MKL formulation, we have shown that in Section 4.3 and 4.4 it is possible to obtain a convergence bound of the order of $qF^{2/q}$, when a normalized kernel, *i.e.,* $\|\phi^j(\boldsymbol{x}_t, y_t)\|_2 \leq 1$, is considered. If $F \geq 3$ and $p = \frac{2\log F}{2\log F - 1}$, $qF^{2/q}$ becomes equal to $2e\log F$, and the rate of convergence will depend logarithmically on the number of kernels (see Corollary 4.1). Therefore, we propose the following regularizer

$$h(\bar{\boldsymbol{w}}) := \lambda/2\ \|\bar{\boldsymbol{w}}\|_{2,\frac{2\log F}{2\log F-1}}^2 + \alpha\|\bar{\boldsymbol{w}}\|_{2,1}\ , \tag{4.8}$$

with the precise aim of having the optimal convergence rate and an exact mathematical sparsity, tunable through a parameter. The first term of h gives us an easy to optimize problem, while the second one induces different levels of sparsity depending on α.

More in details, with this choice of p the regularization becomes similar to the entropic regularization. A similar method has been used in the context of sparse linear optimization in Shalev-Shwartz and Tewari

Algorithm 7 The UFO-MKL algorithm.

```
1: Input: $\alpha, \lambda, T$
2: Initialize: $\bar{\boldsymbol{w}}_1 = \boldsymbol{0}$, $\bar{\boldsymbol{\theta}}_1 = \boldsymbol{0}$, $q = 2\log F$
3: for $t = 1, 2, \ldots, T$ do
4:     Sample at random $(\boldsymbol{x}_t, y_t)$
5:     $\bar{\boldsymbol{z}}_t = \partial\ell\left(\bar{\boldsymbol{w}}_t, \boldsymbol{x}_t, y_t\right)$
6:     $\bar{\boldsymbol{\theta}}_{t+1} = \bar{\boldsymbol{\theta}}_t - \bar{\boldsymbol{z}}_t$
7:     $v_j = \left| \|\boldsymbol{\theta}^j_{t+1}\|_2 - \alpha t \right|_+, \forall j = 1, \cdots, F$
8:     $\boldsymbol{w}^j_{t+1} = \frac{v_j \boldsymbol{\theta}^j_{t+1}}{t\lambda \|\boldsymbol{\theta}^j_{t+1}\|_2} \left(\frac{v_j}{\|\boldsymbol{v}\|_q}\right)^{q-2}, \ \forall j = 1, \cdots, F$
9: end for
```

(2009). This motivates the choice of the first term in h. On the other hand, using only this term would result in a fixed regularization function, losing the possibility to adapt it to the problem. Hence we mix the $(2, 2\log F/(2\log F - 1))$ squared group norm with a $(2, 1)$ group norm, to be able to tune the level of sparsity.

Hence the optimization problem becomes

$$\min_{\bar{\boldsymbol{w}}} \ \lambda/2 \ \|\bar{\boldsymbol{w}}\|^2_{2, \frac{2\log F}{2\log F - 1}} + \alpha \|\bar{\boldsymbol{w}}\|_{2,1} + \frac{1}{N}\sum_{i=1}^{N} \ell\left(\bar{\boldsymbol{w}}, \boldsymbol{x}_i, y_i\right) \ . \tag{4.9}$$

Since the new regularizer (4.8) is $1/2\log F$-strongly convex for any value of α, to minimize (4.9) we could use stochastic gradient and mirror descent methods as the previous sections. Here we use the primal-dual framework for minimization of regularized loss functions (Shalev-Shwartz and Kakade, 2008), which is different from the Pegasos framework Shalev-Shwartz *et al.* (2007) used by the OBSCURE algorithm. In the primal-dual framework there is no rescaling of the hyperplane at each step, so each single iteration will be faster. Our method is also close to the dual averaging framework propose by Xiao (2010).

4.5.2 Algorithm

The pseudo code of the algorithm is illustrated in Algorithm 7. As the OM-2 algorithm and the OBSCURE algorithm, the algorithm maintains two set of weights, a primal one $\bar{\boldsymbol{w}}_t$ and a dual one $\bar{\boldsymbol{\theta}}_t$. At each step it takes a sample at random from the training set and update the dual vector $\bar{\boldsymbol{\theta}}_t$ with a subgradient descent step, where $\partial\ell\left(\bar{\boldsymbol{w}}_t, \boldsymbol{x}_t, y_t\right)$ is the subgradient w.r.t. $\bar{\boldsymbol{w}}_t$. Then the primal weight $\bar{\boldsymbol{w}}_t$ is calculated with lines 7-8. These two lines correspond to the gradient of the Fenchel dual of the elastic regularizer h. Line 7 has the effect to put to zeros the kernels that have a norm smaller than αt.

Therefore, it induces exact sparsity in the domain of the kernels.

4.5.3 Analysis

In this section we prove a theoretical guarantee for the converge rate of UFO-MKL to the optimal solution of (4.9). To prove the convergence, we could directly apply Theorem 2 in (Shalev-Shwartz and Kakade, 2008), that for completeness we restate here in a simpler form and with our notation.

Theorem 4.4. *(Shalev-Shwartz and Kakade, 2008) Let g be a β-strongly convex function w.r.t. the norm $\|\cdot\|$ over a set S and let $\|\cdot\|_*$ be its dual norm. Let $\ell_1, \ldots, \ell_T$ be an arbitrary sequence of convex loss functions, and R such that $\max_i \|\partial\ell_i(\boldsymbol{w}_i)\|_* \leq R$. Define $\boldsymbol{w}_t = \nabla g^*(-\frac{\eta}{t}\sum_{i=1}^{t-1}\partial\ell_i(\boldsymbol{w}_i))$ then, for any $\boldsymbol{u} \in \mathbb{S}$, and any $\eta > 0$ we have*

$$\frac{1}{T}\sum_{t=1}^{T}\left(\frac{g(\boldsymbol{w}_t)}{\eta} + \ell_t(\boldsymbol{w}_t)\right) - \frac{1}{T}\sum_{t=1}^{T}\left(\frac{g(\boldsymbol{u})}{\eta} + \ell_t(\boldsymbol{u})\right) \leq \eta\frac{R^2(1+\log T)}{2\beta T} \ .$$

This theorem introduces another way to minimize strongly convex regularized objective functions through stochastic gradient descent. It could be applied for example to minimize the standard SVM objective function. Note that the resulting convergence bound would be equal to the Pegasos' one. Also, there is no rescaling of the hyperplane at each step of Algorithm 7. This lack of rescaling makes each iteration of UFO-MKL faster than each iteration of OBSCURE (Algorithm 5), so that the total time needed to converge can be smaller. We will verify this experimentally in Section 4.5.4.

Theorem 4.5. *Denote by $f(\bar{\boldsymbol{w}}) = h(\bar{\boldsymbol{w}}) + \frac{1}{N}\sum_{i=1}^{N}\ell\left(\bar{\boldsymbol{w}}, \boldsymbol{x}_i, y_i\right)$ and by $\bar{\boldsymbol{w}}^*$ the solution that minimizes it. Suppose that the kernel are normalized $\|\phi^j(\boldsymbol{x}_t, \cdot)\|_2 \leq 1$, and the subgradient of the loss function ℓ satisfies $\|\boldsymbol{z}^j\|_2 \leq L\|\phi^j(\boldsymbol{x}, y')\|_2$, $\forall j = 1, \ldots, F,\ y' \in \mathbb{Y}$. Let $\delta \in (0,1)$, then with probability at least $1-\delta$ over the choices of the random samples we have that after T iterations of the UFO-MKL algorithm*

$$f(\bar{\boldsymbol{w}}_{T+1}) - f(\bar{\boldsymbol{w}}^*) \leq \frac{eL^2(1+\log T)\log F}{\lambda\delta T},$$

where e is the Euler's number.

To derive the UFO-MKL algorithm the only thing that is missing is to calculate $\nabla h^*(\bar{\boldsymbol{\theta}})$.

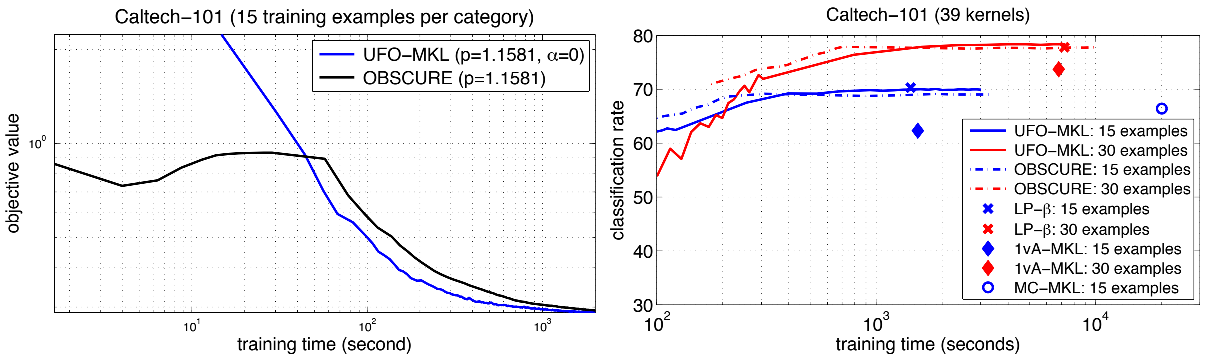


Figure 4.9. (left) Comparison of UFO-MKL to OBSCURE on convergence rate with the same value of p, and $\alpha = 0$ for UFO-MKL. (right) Performance comparison on Caltech-101 using different MKL algorithms.

Theorem 4.6. *Let*

$$\boldsymbol{v} = \left[\left|\|\boldsymbol{\theta}^1\|_2 - \alpha\right|_+, \cdots, \left|\|\boldsymbol{\theta}^F\|_2 - \alpha\right|_+\right],$$

then the component j of $\nabla h^*(\bar{\boldsymbol{\theta}})$ is equal to

$$\frac{\boldsymbol{\theta}^j}{\lambda\|\boldsymbol{\theta}^j\|_2} \frac{v_j^{q-1}}{\|\boldsymbol{v}\|_q^{q-2}}$$

4.5.4 Experiments

In this section, we compare UFO-MKL against the SILP algorithm and the Multiclass MKL (MC-MKL), as well as the OBSCURE algorithm, using the implementation in DOGMA. We consider the parameter $\alpha \in \{0.0001, 0.001, 0.0025, 0.005, 0.0075, 0.01, 0.02\}$, and use p=1.01 for the OBSCURE algorithm when sparsity is desired in the solution. The λ parameter has been chosen by cross validation as $1/(CN)$, where N is the number of training points, and C is from the set $\{1, 10, 10^2, 10^3\}$, and C=1000 yields the best results for all the algorithms. As the previous algorithms, we also use multiclass loss ℓ^{MC}. For multiple classes extension of the binary SILP algorithm, we use the 1-vs-All scheme.

Multiclass Image Classification: Caltech-101

In the experiments we use the same setup as described in the previous section 4.4.5. We start by comparing the convergence rates of UFO-MKL and OBSCURE. The training time of the OBSCURE algorithm is proportional to q/λ, where $1/p + 1/q = 1$. Therefore, when a sparse solution is needed, the algorithm becomes slow because q becomes big. For a fair comparison, we first set $q = 2 \log F$ in OBSCURE, and $\alpha = 0$ in UFO-MKL, so that their regularizers become exactly the same. Figure 4.9 (left) shows the value of the objective function as a function of the training time. OBSCURE is faster in the beginning because its first stage is an online algorithm, which quickly determines the region of the space where the optimal solution lives. UFO-MKL, after $\approx$ 1min of computation, converges faster than OBSCURE. We think that this is due to the simpler algorithm that does not require a scaling after each update, hence each iteration in UFO-MKL is faster. In practice, when the data is sparse, in another words, p is set to close to 1 for OBSCURE algorithm, the efficiency of UFO-MKL becomes more significant (see for example Figure 4.10).

The test results are reported in Figure 4.9 (right). The results support our claim that the multiclass loss function is more suitable for this type of problem, as all the methods that use the multiclass loss outperform SILP. MC-MKL is computationally infeasible for 30 samples per category, and its significant performance gap from OBSCURE and UFO-MKL seems to indicate that it stops before converging to the optimal solution. UFO-MKL also outperforms OBSCURE, probably because OBSCURE does not get a real sparse solution although it tends to be sparse. More importantly, the accuracy is comparable or better than the best results obtained in the literature by the LP-β algorithm.

Scalability w.r.t. the number of kernels

To test the scalability of UFO-MKL we tested it on the Oxford Flower data set, generating 1400 kernels. The task of the dataset is to classify 17 different flower categories. Each class has 80 images with predefined train and test splits. Precomputed distance matrices for 7 different features are available. For each precomputed matrix, we generate 200 kernels using $\exp(-\gamma^{-1} \cdot d)$ with 200 different γ values in the range between 0.01 and 100. Figure 4.10 (left) reports the results for varying number of kernels. Our algorithm outperforms all the other baseline in term of both accuracy and efficiency. UFO-MKL is 3-5 times faster compared to OBSCURE, which is again due to the fact that OBSCURE does not get a real sparse solution. It suggests that UFO-MKL is more suitable for feature selection tasks when a lot of kernels are available. Figure 4.10 (middle) shows the number of selected kernels and the accuracy obtained by varying values of α, using the same number of epochs. We see that a larger value of α, which corresponds to a sparser solution, leads to a slower running time. Figure 4.10 (right) reports the accuracy and the number of non-zero kernel weights of the last solution when the algorithm stops. It can be seen that when the model becomes over sparse (larger α) the performance starts dropping. However small values of α, which result in a denser model, do not correspond to higher accuracy, in fact many less discriminative kernels are included in the solution.

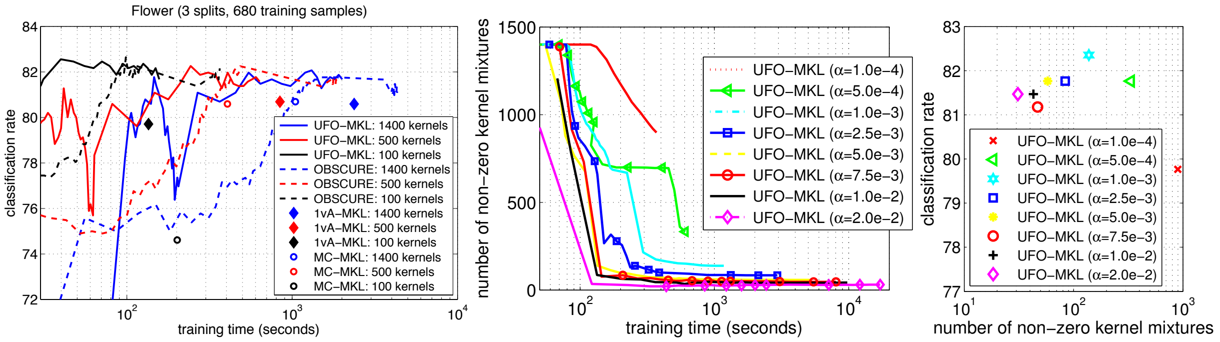


Figure 4.10. Running time performance of UFO-MKL and other baseline approaches w.r.t. different number of kernels and varying values of α.

4.6 Conclusion

In this chapter, we introduce several theoretically motivated algorithm for solving the MKL problem. The OM-2 algorithm and the OBSCURE algorithm use a new p-norm formulation of the MKL problem that allows us to tune the level of sparsity in order to obtain always nearly optimal performance. The UFO-MKL algorithm uses an elastic-net kind of regularization function which preserve strong convexity and have exact mathematical sparsity, tunable through a parameter. All the algorithms use stochastic gradient descent method and mirror descent method to solve the corresponding MKL formulations, which are very efficient and an order of magnitude faster compared to the other cue combination baseline approaches. Our approaches are general, so they can be used with any kind of convex losses, from binary losses to structured output prediction (Tsochantaridis *et al.*, 2004), and even to regression losses. Through experiments, we found that the negative results on the MKL algorithms, which do not improve their performance over simple averaging kernel baseline (Gehler and Nowozin, 2009b), are mainly due to their computational bottle neck which prevent them from using loss functions such as the multiclass loss which are more suitable for the tasks at hand. By using our method, MKL can still be an efficient machine learning tool for cue combination tasks.

This chapter concludes the first part of this thesis. We presented several efficient cue integration algorithms. All the algorithms presented in this thesis use the linear model as hypothesis class, and they achieve state-of-art performance and outperform other baselines using single or multiple cues, particularly the MKL algorithms. Our algorithms converge to the optimal solution of the MKL formulation efficiently. The MKL formulation employs similar objective function used in SVM. However, the large margin principle used in this method causes the scaling problem and initialization problem, which can strongly affect final solutions of learned kernels as well as performance (Chapelle *et al.*, 2002; Gai *et al.*, 2010). Moreover, only linear and non-negative combination of kernels were studied, while more powerful non-linear combinations which has not been studied much except in few papers (*e.g.,* Cortes *et al.* (2009a)) might lead to better results. Possible directions in the future are to develop new learning objective function which can lead to performance improvement, and to study the non-linear combinations of kernels.

Part II

From Cues Integration to Cues Exploitation

> *Differences in the world are only detectable because different senses perceive the same world events differently.*
>
> *Aristotle, De Anima (350 B.C.E)*

Chapter 5

Weakly Supervised Learning using Co-Occurring Cue(s)

In this chapter, we present a general learning framework to address the problem of training visual classifiers from images using natural accompanying captions without any explicit manual intervention. It is motivated by the fact that the captions (textual cue) weakly label the images, as they usually describe the images, but the words in the captions are not aligned with regions in the images. Moreover the captions usually contain many unrelated words, and many sentences in the captions may not describe the image. Under our new framework, each training image is represented as a bag of regions, associated with a set of candidate labeling vectors. Each labeling vector encodes the possible labels for the regions of the image. The set of all possible labeling vectors can be generated automatically from the caption using natural language processing techniques (Section 5.3). The use of labeling vectors provides a principled way to include diverse information from the captions, such as multiple types of words corresponding to different attributes of the same image region, labeling constraints derived from grammatical connections between words, uniqueness constraints, and spatial position indicators. Moreover, it can also be used to incorporate high-level domain knowledge useful for improving learning performance. We show that learning is possible under this weakly supervised setup (Section 5.4.1 and 5.4.2). Exploiting this property of the problem, we again use a large margin discriminative formulation to model the problem, and propose an efficient algorithm to solve it (Section 5.4.3 to 5.4.5). We apply the resulted framework and algorithm to artificial datasets (Section 5.5) as well as two real-world images and captions datasets, one on learning face classifiers using news images with captions (Section 5.6), another on modeling two type of words (name and verb) jointly to improve learning performance (Section 5.7). Contents presented in this Chapter is based on the following publication:

> Jie, L., Caputo, B., and Ferrari, V. (2009). Who's Doing What: Joint Modeling of Names and Verbs for Simultaneous Face and Pose Annotation. In *Advances in Neural Information Processing Systems 22*.
>
> Jie, L., and Orabona, F. (2010). Learning from Candidate Labeling Sets. In *Advances in Neural Information Processing Systems 23*.
>
> Jie, L., Orabona, F., Caputo, B., and Ferrari, V. (2011). Learning from Images with Captions Using the Maximum Margin Set Algorithm. *Idiap-RR-30-2011*.

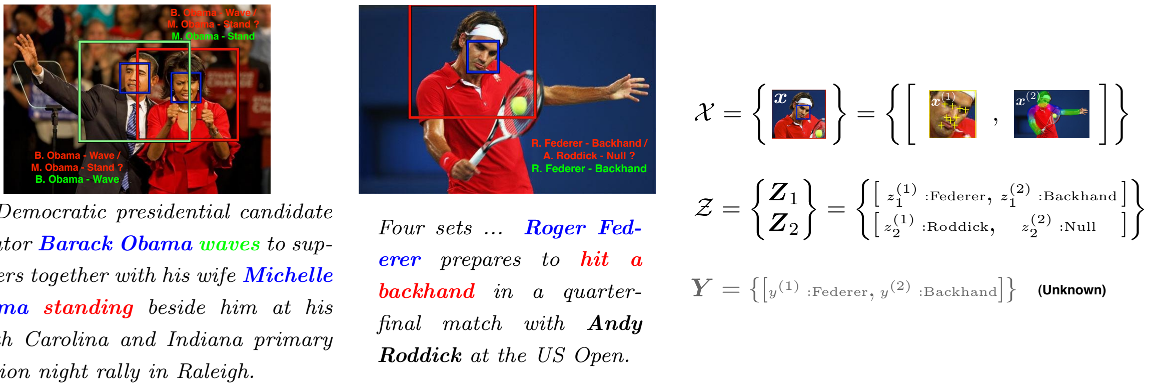
5.1 Introduction

A huge amount of images with accompanying text captions are available on the Internet. Websites selling various items such as houses and clothing provide photographs of their products along with concise descriptions. Online newspapers (*e.g.,* `news.yahoo.com`) have pictures illustrating events and comment them in the caption. These news websites are very popular because people are interested in other people, especially if they are famous (Figure 5.1). This motivates the recent interest in using captioned images for training visual classifiers. Exploiting the latent associations between images and the text cues can lead to a virtually infinite source of training annotations, without any explicit manual intervention. The learned model can then be used in a variety of Computer Vision applications, including face recognition, image search engines, and to annotate new images for which no caption is available.

There have been several works that study this problem on different applications and from different perspectives. Previous works have focused on associating names (Berg *et al.*, 2004b; Guillaumin *et al.*, 2010) in the captions to the faces of people in news images, on learning character naming systems from TV series using scripts (Everingham *et al.*, 2006) and screenplays (Cour *et al.*, 2009), on learning scene classification models from tagged photos (Barnard *et al.*, 2003; Grangier and Bengio, 2008; Wang and Mori, 2010), and on learning object recognition models from an online nature encyclopedia (Wang *et al.*, 2009). All these can be considered as weakly supervised learning problems, because each segment, face, or object in the image is only indirectly, ambiguously labeled by the words in the captions.

The above tasks are more challenging than standard supervised learning tasks due to the *correspondence ambiguity* problem: it is not known beforehand which part of the image corresponds to which part of the caption. Moreover, not everything mentioned in a natural text caption appears in the image, and, vice-versa, not everything in the image is mentioned by the caption. This is different from using tags which are guaranteed to describe the image, as in the Corel database (Barnard *et al.*, 2003). On the other hand, natural language descriptions contain rich semantic information about the relations between different image regions and labels. For example, in Figure 5.1 (left), knowing what "waves" (verb) means would reveal who of the two imaged persons is "Barak Obama" (subject). The other way around, knowing who is "Barak Obama" would deliver a visual example for the "waving" pose. This connection between the name and the verb can be exploited to constrain the labeling: if a region is labeled by the name Barak, then it must also be labeled by "waving". A labeling like "Barack-standing" is not valid given the caption. The caption sometimes enables to impose also other constraints. For instance, we know that Federer cannot appear twice in an image, so no two image regions can take the same label "Federer". As another example, captions sometimes contain spatial position indicators. In the example of Figure 5.2, "Chervynsky" cannot be a valid label for the person in the middle. Such constraints can be used to prune the space of possible labelings, which facilitates learning (Berg *et al.*, 2004b; Guillaumin *et al.*, 2010). To the best of our knowledge, all existing algorithms are designed to explicitly incorporate a particular type of constraint. This means the algorithm has to be redesigned in order to integrate a new type of constraint.

In this chapter, we propose a general, weakly supervised learning framework to model the problem of learning from images with captions. In this framework, each training image is represented as a bag of regions, and is associated with a set of *candidate labeling vectors*. Each candidate labeling vector encodes a possible labeling of all regions, with only one candidate labeling being fully correct. The set of candidate labeling vectors can be generated automatically from the captions using natural language processing (NLP) tools. This framework provides a unified way to include many types of constraints.



US Democratic presidential candidate Senator ***Barack Obama*** ***waves*** *to supporters together with his wife* ***Michelle Obama*** ***standing*** *beside him at his North Carolina and Indiana primary election night rally in Raleigh.*

Four sets ... ***Roger Federer*** *prepares to* ***hit a backhand*** *in a quarter-final match with* ***Andy Roddick*** *at the US Open.*

$$\mathcal{Z} = \begin{Bmatrix} \boldsymbol{Z}_1 \\ \boldsymbol{Z}_2 \end{Bmatrix} = \begin{Bmatrix} \left[\, z_1^{(1)} \text{ :Federer}, \; z_1^{(2)} \text{ :Backhand} \right] \\ \left[\, z_2^{(1)} \text{ :Roddick}, \; z_2^{(2)} \text{ :Null} \right] \end{Bmatrix}$$

$$\boldsymbol{Y} = \left\{ \left[y^{(1)} \text{ :Federer}, \; y^{(2)} \text{ :Backhand} \right] \right\}$$

Figure 5.1. (Left, Middle) Two examples of image-caption pairs for the "who is doing what" task. The face and upper body of the persons in the image are marked by bounding-boxes. We stress that a caption might contain names and/or verbs not visible in the image, and vice-versa. (Right) our candidate labeling set notation for the example in the middle. The image $\mathcal{X}$ contains one region $\boldsymbol{x}$, which has two attributes: the person name and the verb describing the action he is performing. The candidate labeling set $\mathcal{Z}$ contains two candidate labeling vectors $\boldsymbol{z}_1$ and $\boldsymbol{z}_2$. Each labeling vector encodes one label for every attribute of the region. Importantly, note how [Roddick, Backhand] is not a candidate, as Roddick is not the subject of the verb "hit a backhand" in the caption. The true labeling vector $\boldsymbol{Y}$ is unknown and must be recovered by the algorithm.

5.2 Related Work

Images and Captions/Tags. Learning visual classifiers from images with tags has been a very active line of research in recent years (Barnard *et al.*, 2003; Grangier and Bengio, 2008; Wang and Mori, 2010). These approaches must resolve the correspondence between image segments and tags, which are typically nouns (*e.g.,* tiger, grass, car). Because the tags are manually annotated to be descriptive for the image, algorithms can safely assume that nearly all tags should correspond to one or more image regions.

The problem of naming faces in images and videos using natural text sources has been particularly well studied (Berg *et al.*, 2004b; Cour *et al.*, 2009; Everingham *et al.*, 2006; Guillaumin *et al.*, 2010). These works exploit the fact that often the names of the persons in the image are mentioned in the caption. Therefore, a caption contains possible labels for the faces in the corresponding image. However, an imaged person might not be mentioned in the caption and vice-versa. Hence, the level of noise and ambiguity in natural captions is typically higher compared to image tags. Various kinds of task-specific knowledge has also been integrated to improve learning performance, such as that two faces in one image can not be associated with the same name (Berg *et al.*, 2004b), or exploiting the motion of the mouth and the gender of a person (Cour *et al.*, 2009).

Recently, a few works went beyond modeling a single type of word, and start to exploit the structure of sentences in the caption. Gupta and David (Gupta and Davis, 2008) model prepositions in addition to nouns (*e.g.,* 'bear in water', 'car on street'). This prunes down the space of possible labelings.

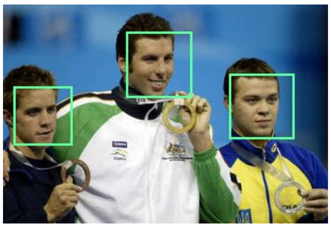
Related learning frameworks. Our problem is different from semi-supervised learning (Zhu, 2005), where the learner has access to a set of labeled examples as well as a set of unlabeled examples. Instead, it is closer to the ambiguously labeled learning or partially labeled learning setting (Cour *et al.*, 2009; Grandvalet, 2002; Hüllermeier and Beringe, 2006; Jin and Ghahramani, 2002), where each training example is associated with multiple labels, only one of which is correct. Many approaches to such problems use the EM algorithm to estimate model parameters and the correct labels (Grandvalet, 2002; Jin and Ghahramani, 2002). The recent work of Cour *et al.* (2009) is the most related to this chapter, as it proposes a convex learning formulation based on minimizing an ambiguous loss function. In this chapter, we generalize the ambiguous function to the multiple instances case, and use a non-convex learning formulation which achieves better performance than the convex learning formulation (sec. 5.4.4).

Our work is also related to multi-label learning (MLL) (Boutell *et al.*, 2004), where each example is assigned multiple labels, any subset of which can be correct. Other related lines of research are multi-instance learning (MIL) (Andrews *et al.*, 2003; Dietterich *et al.*, 1997), and multi-instance multi-label learning (MIML) (Zhang and Zhou, 2008; Zhou and Zhang, 2006). MIML extends the two-label MIL setup to multiple labels. In both setups, instances are grouped into bags. The labels of the individual instances are not given. Instead, labels are given to the bags. However, contrary to our framework, in MIML noisy labels are not allowed: all the given labels for a bag are correct. Moreover, current MIL and MIML algorithms usually rely on a 'key' instance in the bag (Andrews *et al.*, 2003) or they transform each bag into a single-instance representation (Zhou and Zhang, 2006). Instead, our algorithm makes an explicit effort to label every instance in a bag and to consider all of them during learning.

Latent Structure SVMs. Our algorithm is also related to Latent Structural SVMs (Felzenszwalb *et al.*, 2010; Yu and Joachims, 2009), where the correct labels are considered as latent variables. Wang and Mori (Wang and Mori, 2010) recently proposed a discriminative latent model for annotating scene images given object nouns as tags (*e.g.,* tiger, grass). They model the ground-truth region-to-annotation mapping and the overall scene label as latent variables.

5.3 Problem Definition

In this section, we define the problem of learning from images with captions, and establish the notation that will be used in the rest of the chapter. Figure 5.1 (right) gives an example of our setup. In Section 5.6



Australia's gold medalist ***Grant Hackett*** *(C), Ukraine's silver medalist* ***Igor Chervynsky*** *and USA's bronze medalist* ***Erik Vendt*** *show their medals following the 1500 metres freestyle race at the 10th World Swimming Championships in Barcelona July 27, 2003. Hackett clocked fourteen minutes 43.14 seconds.*

Figure 5.2. Example of an image-caption pair containing spatial indicators. The spatial indicator (C) indicates **Hackett** is the person in the middle, reducing the ambiguity in labels assignment.

and Section 5.7 we will give several examples on how to cast existing problems into our framework.

Input data. The input is a collection of N image and caption pairs $\{\mathcal{X}_i, \mathcal{C}_i\}_{i=1}^N$. An image $\mathcal{X}_i$ consists of M_i regions $\mathcal{X}_i = \{\boldsymbol{x}_{i,m}\}_{m=1}^{M_i}$, and $\boldsymbol{x}_{i,m} \in \mathbb{X}$. Each image has an associated caption $\mathcal{C}_i$, which implicitly provides partial labels for the image. Many real-world objects can belong to multiple concepts simultaneously. For example, an image region can bear several attributes: red (color), metal (texture) and car (object category). Quite often these attributes are correlated, so we argue that it is useful to model them together, using a label for each attribute. Without loss of generality, we assume that labels $\boldsymbol{Y}_i = \{\boldsymbol{y}_{i,m}\}_{m=1}^{M_i}$ exist for every image region, but they are unknown during training. We consider them as latent variables. The latent variables $\{\boldsymbol{y}_{i,m}\}_{m=1}^{M_i}$ encode the labels for each region in the images. Each $\boldsymbol{y}_{i,m}$ is either a set of labels $\{y_{i,m}^{(p)}\}_{p=1}^{P}$ or a single label $y_{i,m}$ (*i.e.*, $P = 1$), where P is the total number of attributes we model simultaneously. Each label $y_{i,m}^{(p)} \in \mathbb{Y}^{(p)} := \{1, 2, \cdots, K^{(p)}\}$ indicates a specific attribute of a region, and $K^{(p)}$ denotes the number of possible different labels for the attribute p.

Candidate Labeling Sets. Our goal is to learn from the input image-caption pairs a classification function $f : \boldsymbol{x} \to \boldsymbol{y}$ to classify regions of a new test image. The caption for the test image, when available, could still be used as an extra source of information to guide the prediction, but it is not required. Although the true labels of a training image are unknown, the accompanying caption usually describes the image. We assume that the labels of the regions only come from the caption. The learning algorithm should label with *null* any region whose true label is not mentioned in the caption. A label corresponds to a word, or to a few words with the same meaning (*e.g.*, "Barack Obama" and "President of the USA"). In the rest of the chapter we will only use the term "word". Based on these assumptions, we generate the set of all possible assignments of the words in a caption to the regions in the corresponding image i, which we call Candidate Labeling Set (CLS) $\mathcal{Z}_i$. We use L_i to denote the number of candidate assignments in

$\mathcal{Z}_i = \{\boldsymbol{Z}_{i,l}\}_{l=1}^{L_i}$, with each $\boldsymbol{Z}_{i,l} \in R^{P \times M_i}$. In other words, there are L_i different possible combinations of labels for the regions in the image i. Only one of these candidate assignments is the true labeling, while the others are only partially correct or even completely wrong. Note that this is *not* equivalent to simply associating L_i candidate labels independently to each region. Instead, our definition explicitly encodes the constrains between multiple regions and labels. To clarify this point, consider a simple example where we have two regions $\{\boldsymbol{x}_{i,1}, \boldsymbol{x}_{i,2}\}$ with two attributes each (color and object category). If it is known that they can only come from classes "red-car" or "blue-motor", and that no two regions can have the same label, then $\boldsymbol{z}_{i,1} = [\text{red-car}, \text{blue-motor}], \boldsymbol{z}_{i,2} = [\text{blue-motor}, \text{red-car}]$ will be the Candidate Labeling Vectors (CLVs) for this bag. Other possibilities such as $[\text{blue-car}, \text{red-motor}]$, $[\text{red-car}, \text{red-car}]$ are excluded. Another example could be an image with three regions, each of which could be labeled either "chair" or "elephant". However, we know there cannot be both chairs and elephants in the same image. Such a structure can be encoded in our CLSs, but not in simple independent label sets for each region.

Constraints between words. As the size of the regions and the number of words grow, the number of admissible labelings becomes intractable. To keep the problem tractable, we could first filter out uninteresting words such as interjections and conjunctions, and maintain a dictionary with only the words we want to model. Each of these words will correspond to a different label. However, the number of admissible labelings can still be very large after the filtering. Let W_i be the number of modeled words in the caption of an image i with M_i regions. In the most general case, this image has $L_i = W_i^{M_i}$ admissible labelings even when only one label can be assigned to each region. In this unconstrained scenario, the supervision information from the caption is very low for large W_i. Fortunately, captions frequently contain valuable context cues which we can extract using NLP tools (OpenNLP, 2010; Deschacht and Moens, 2009). These context cues can be translated into constraints to remove assignments from $\mathcal{Z}_i$. In addition, we can also reduce the size of $\mathcal{Z}_i$ by incorporating high-level domain knowledge. As L_i decreases when more constraints are added, the CLS $\mathcal{Z}_i$ becomes less ambiguous. This scenario allows us to design interesting learning algorithms, which we present in the next Section.

Our CLSs framework supports several types of useful constraints:

- [C1] *Word type matches region type.* For example, a name can only be associated with a face region, and/or a verb can only be associated with a person body region (where such regions are detected beforehand, *e.g.,* by an off-the-shelf face detector (Viola and Jones, 2004; Rodriguez, 2006) or an upper-body detector (Ferrari *et al.*, 2008b)). This type of constraint has been used in several works (Berg *et al.*, 2004b; Guillaumin *et al.*, 2008) (see Figure 5.1 for example).

- [C2] *Sentence structure.* Multiple words grammatically connected in the caption must be assigned to spatially related image regions or even to the same region. Several kinds of connections in the caption can be used to eliminate labelings from $\mathcal{Z}_i$ which violate the resulting constraints: (a) noun-adjective (Wang and Forsyth, 2009), *e.g.,* a "red car", where both "red" and "car" are attributes of the same region; (b) noun-preposition-noun (Gupta and Davis, 2008), *e.g.,* "sun in the sky" indicates that the two regions are close to each other, and the "sun" region is surrounded by the "sky" region. In general, this kind of connection conveys information about the spatial relationship between two regions. (c) name-verb, *e.g.,* "Roger Federer hits a backhand", "Roger Federer" is the subject of "hits backhand" and therefore point to the same person in the image. Therefore, any labeling that assigns "Roger Federer" to a certain face region, must assign "hits backhand" to the body region of the *same person*;

- [C3] *Uniqueness.* Some words can only appear once in the image. Two face regions in the same image cannot be associated to the same name (Berg *et al.*, 2004b; Guillaumin *et al.*, 2008).

- [C4] *Spatial indicators.* Captions sometimes contain spatial position indicators (Berg *et al.*, 2004b) such as "(L)" and "left". These suggest the relative spatial position of an image region w.r.t. the others. An example of this kind of connection is shown in Figure 5.2. The noun-preposition-noun structure discussed in [C2] can also be considered as a spatial indicator.

Based on these constraints, we can explicitly enumerate the set of admissible assignments $\mathcal{Z}_i$ from the caption $\mathcal{C}_i$ in several interesting problems. Hence, we replace the captions by the CLSs $\{\mathcal{Z}_i\}_{i=1}^N$. In a few other cases, memory limitations prevents us from explicitly storing all assignments $\mathcal{Z}_i$. In such a case we store the words and the constraints, which we can use to generate subsets of $\mathcal{Z}_i$ "on the fly" during learning.

In this setting, the training data are provided in the form $\{\mathcal{X}_i, \mathcal{Z}_i\}_{i=1}^N$. Each image $\mathcal{X}_i$ is associated with a set of CLVs $\mathcal{Z}_i$ (including one which is fully correct). Thus, our goal is to design a learning algorithm which learns classifiers from input data in this special form. Along the way to learning these classifiers, our algorithm also selects one CLV for each image, thus resolving the correspondence between image regions and words in the caption.

5.4 Learning from the Candidate Labeling Sets and the MMS Algorithm

In Section 5.3 we have discussed how to transform the problem of learning from image-caption pairs into the CLS problem. In this section, we first propose a large margin formulation of the CLS problem (Section 5.4.1 to 5.4.3), then we present an efficient algorithm to optimize the proposed formulation (Section 5.4.4 and 5.4.5).

Let $\mathcal{X}$ be the generic bag with M instances $\{\boldsymbol{x}_1, \ldots, \boldsymbol{x}_m, \ldots, \boldsymbol{x}_M\}$, $\mathcal{Z} = \{\boldsymbol{Z}_1, \ldots, \boldsymbol{Z}_l, \ldots, \boldsymbol{Z}_L\}$ the generic set of CLVs. Under this representation, $\mathcal{X}$ can be an image with M regions, and an instance $\boldsymbol{x}_m$ is a vector of appearance features describing the m-th region. An image region can either be a rectangular bounding box (*e.g.,* a face) found by an object detector (Viola and Jones, 2004), or an arbitrarily shaped segment found by an unsupervised segmentation algorithm (Felzenszwalb and Huttenlocher, 2004). Furthermore, let $\boldsymbol{Y} = \{\boldsymbol{y}_1, \ldots, \boldsymbol{y}_M\}, \boldsymbol{Z} = \{\boldsymbol{z}_1, \ldots, \boldsymbol{z}_M\}$ be two labeling vectors, where each element $\boldsymbol{z}, \boldsymbol{y} \in \mathbb{Y}^P$ is a label, with P denoting the number of attributes we model for each instance. We also assume a uniform prior over the CLVs in the CLSs, *i.e.,* $p(\boldsymbol{Z}_i) = p(\boldsymbol{Z}_j),\ \forall\ \boldsymbol{Z}_i, \boldsymbol{Z}_j \in \mathcal{Z}$. Later, we will discuss the possibility to extend these probabilities when the priors for each $\boldsymbol{Z}_i$ are known.

5.4.1 Prediction functions

Given the training data $\{\mathcal{X}_i, \mathcal{Z}_i\}_{i=1}^N$, we want to learn a linear score function $s_{\boldsymbol{w}}(\boldsymbol{x}, y^{(p)}) = \boldsymbol{w} \cdot \phi(\boldsymbol{x}, y^{(p)})$ which can work on individual instance and attribute as defined in Section 1.3. We also define the linear prediction function for an image region $\boldsymbol{x}$ as

$$f_{\boldsymbol{w}}(\boldsymbol{x}) = \operatorname*{arg\,max}_{y^{(p)} \in \mathbb{Y}^{(p)}} \sum_{p=1}^{P} s_{\boldsymbol{w}}(\boldsymbol{x}, y^{(p)}) = \operatorname*{arg\,max}_{\boldsymbol{y} \in \mathbb{Y}}\ \boldsymbol{w} \cdot \psi(\boldsymbol{x}, \boldsymbol{y})\ ,$$

where $\psi(\boldsymbol{x}, \boldsymbol{y}) = \sum_{p=1}^{P} \phi(\boldsymbol{x}, y^{(p)})$ is a joint feature mapping vector between the region $\boldsymbol{x}$ and the labeling vector $\boldsymbol{y}$. This definition includes the special case of training different hyperplanes, one for each class.

Indeed $\psi(\boldsymbol{x}, \boldsymbol{y})$ can be defined as

$$\psi(\boldsymbol{x}, \boldsymbol{y}) = [\, \overbrace{\mathbf{0},\ \cdots,\ \mathbf{0},\ \underbrace{\phi^{(1)}(\boldsymbol{x}),}_{y^{(1)}\text{-th}}\ \mathbf{0},\ \cdots,\ \mathbf{0},}^{K^{(1)}}\ \cdots,\ \overbrace{\mathbf{0},\ \cdots,\ \mathbf{0},\ \underbrace{\phi^{(p)}(\boldsymbol{x}),}_{y^{(p)}\text{-th}}\ \mathbf{0},\ \cdots,\ \mathbf{0},}^{K^{(p)}}\ \cdots] \ ,$$

where $\phi^{(p)}(\cdot)$ is a transformation that depends only on the data and the attribute p. In this case the classifier $\boldsymbol{w}$ is parameterized by $\sum_{p=1}^{P} K^{(p)}$ hyperplanes $\boldsymbol{w}_{y^{(p)}}$.

We can now define the score function for the image as $\mathbf{S}_{\boldsymbol{w}}(\mathcal{X}, \boldsymbol{Y})$, which intuitively is gathering from each region in $\mathcal{X}$ the confidence on the labels encoded in $\boldsymbol{Y}$. With the definitions above, we define the function $\mathbf{S}$ as

$$\mathbf{S}_{\boldsymbol{w}}(\mathcal{X}, \boldsymbol{Y}) = \sum_{m=1}^{M} \boldsymbol{s}_{\boldsymbol{w}}(\boldsymbol{x}_m, \boldsymbol{y}_m) = \sum_{m=1}^{M} \boldsymbol{w} \cdot \psi(\boldsymbol{x}_m, \boldsymbol{y}_m) = \boldsymbol{w} \cdot \Phi(\mathcal{X}, \boldsymbol{Y}) \ , \tag{5.1}$$

where we have $\Phi(\mathcal{X}, \boldsymbol{Y}) = \sum_{m=1}^{M} \psi(\boldsymbol{x}_m, \boldsymbol{y}_m)$. Given the CLS $\mathcal{Z}$, the predictions of the classifier are computed as $\mathbf{F}_{\boldsymbol{w}}(\mathcal{X}, \mathcal{Z}) = \arg\max_{\boldsymbol{Z} \in \mathcal{Z}} \ \mathbf{S}_{\boldsymbol{w}}(\mathcal{X}, \boldsymbol{Z})$.

Remark 5.1. *If the prior probabilities of the CLVs $\boldsymbol{z}_l \in \mathcal{Z}$ are also available, they can be incorporated into the score function by slightly modifying the feature mapping function in eq. (5.1) to $p(\boldsymbol{Z}_i) \cdot \Phi(\mathcal{X}, \boldsymbol{Z}_i)$, where each $p(\boldsymbol{Z}_i)$ is the prior probability for $\boldsymbol{Z}_i \in \mathcal{Z}$.*

5.4.2 Ambiguous loss functions

In the supervised learning setup, many loss functions have been proposed based on minimization of a convex upper bound of an arbitrary risk measurement function $\boldsymbol{\Delta} : \mathbb{Y} \times \mathbb{Y} \to \mathbb{R}$, which quantifies how much a predicted label differs from the true label. A classic loss function is the 0/1 loss:

$$\boldsymbol{\Delta}_{01}(\boldsymbol{Z}, \boldsymbol{Y}) = \sum_{m=1}^{M} \sum_{p=1}^{P} \Delta_{01}(z_m^{(p)}, y_m^{(p)}) = \sum_{m=1}^{M} \sum_{p=1}^{P} \mathbf{1}(z_m^{(p)} \neq y_m^{(p)}) \ ,$$

where $\mathbf{1}(\cdot)$ is the indicator function, $\boldsymbol{Y} = \{\boldsymbol{y}_1, \ldots, \boldsymbol{y}_M\}$ are the true labels of regions $\boldsymbol{x}_m$, and $\boldsymbol{Z}$ are the predicted labels. Hence, $\boldsymbol{\Delta}_{01}(\boldsymbol{Z}, \boldsymbol{Y})$ simply counts the number of mislabeled attributes over all regions.

However, in our setup the true labeling is unknown, and we only have access to the CLS $\mathcal{Z}$, knowing that the true labeling vector is in it. So we propose to use an ambiguous version of the loss $\boldsymbol{\Delta}_{01}$, as a proxy for it:

$$\boldsymbol{\Delta}_A(\boldsymbol{Z}, \mathcal{Z}) = \min_{\boldsymbol{Z}' \in \mathcal{Z}} \boldsymbol{\Delta}_{01}(\boldsymbol{Z}, \boldsymbol{Z}') \ .$$

This loss function underestimates the true loss, while our goal is to minimize the true loss. Nevertheless, we can prove a strong connection between the ambiguous loss $\boldsymbol{\Delta}_A(\boldsymbol{Z}, \mathcal{Z})$ and the true loss $\boldsymbol{\Delta}_{01}(\boldsymbol{Z}, \boldsymbol{Y})$.

The following proposition shows that, in expectation, the ambiguous loss upper bounds the true 0/1 loss up to a constant multiplicative factor. To prove this, we use the theorems (Proposition 3.1 to 3.3) stated in Cour *et al.* (2009) (for completeness we restate here the main proposition (Cour *et al.*, 2009, Proposition 3.1) with our notation), and define an *ambiguity degree* factor η for a region $\boldsymbol{x}_m$. The value of η corresponds to the maximum probability of a noise label (*i.e.*, $\forall y \in \mathbb{Y}^{(p)} \setminus y_m^{(p)}$) co-occurring with a true label $y_m^{(p)}$ in the CLS $\mathcal{Z}$, over all labels and examples generated by an unknown distribution P. We also define the ambiguous loss for a single instance $\boldsymbol{x}$ and an attribute y as $\Delta_A(z, \mathcal{Z}) = \min_{z' \in \mathcal{Z}} \Delta_{01}(z, z') = \mathbf{1}(z \notin \mathcal{Z})$, with $\mathcal{Z}$ being the set of candidate labels for this instance.

Proposition 5.1. *(Cour* et al.*, 2009, Proposition 3.1) For any classifier f and distribution P with the ambiguity degree factor $\eta < 1$, we have*

$$E_P \left[\Delta_A \left(f(x), \mathcal{Z}\right)\right] \leq E_P \left[\Delta_{01} \left(f(x), y\right)\right] \leq \frac{1}{1-\eta} E_P \left[\Delta_A \left(f(x), \mathcal{Z}\right)\right] \ .$$

Proposition 5.2. $E_P \left[\boldsymbol{\Delta}_{01}(\boldsymbol{Z}, \boldsymbol{Y})\right] \leq \frac{1}{1-\eta} E_P \left[\boldsymbol{\Delta}_A(\boldsymbol{Z}, \mathcal{Z})\right] \ .$

Remark 5.2. *The tightness of the above bound directly relates to the ambiguity degree η. When $\eta = 0$, we have only one labeling vector in every labeling set, i.e., $L_i = 1$. In this case, the problem becomes standard supervised learning, and the bound is tight. On the other extreme, when $\eta = 1$, which means a certain noise label always co-occurs with a true label $y^{(p)}$, it is impossible to distinguish them. One weakness of the stated bound is that it becomes very loose when there is a noise label that makes η very large, because that η equals the maximum probability of co-occurring among all the possible noise labels, although it only affects a few true labels. Nevertheless, using the extensions of Proposition 3.2 and Proposition 3.3 in Cour* et al. *(2009), it is possible to obtain label-specific bounds, which enable to retain good learning performance on the subset of labels with low label-specific ambiguity degrees.*

Hence, by minimizing the ambiguous loss we are actually minimizing an upper bound of the expected true loss. It is known that direct minimization of this loss is hard (Cristianini and Shawe-Taylor, 2000). Therefore, in the following we introduce another loss that upper bounds $\boldsymbol{\Delta}_A$ which can be minimized efficiently:

$$\ell_A\left(\mathcal{X}, \mathcal{Z}; \boldsymbol{w}\right) = \left| \max_{\bar{\boldsymbol{Z}} \notin \mathcal{Z}} \left(\boldsymbol{\Delta}_A(\bar{\boldsymbol{Z}}, \mathcal{Z}) + \mathbf{S}_{\boldsymbol{w}}(\mathcal{X}, \bar{\boldsymbol{Z}})\right) - \max_{\boldsymbol{Z} \in \mathcal{Z}} \mathbf{S}_{\boldsymbol{w}}(\mathcal{X}, \boldsymbol{Z}) \right|_+ . \tag{5.2}$$

The following proposition shows that ℓ_A upper bounds $\boldsymbol{\Delta}_A$.

Proposition 5.3. $\ell_A\left(\mathcal{X}, \mathcal{Z}; \boldsymbol{w}\right) \geq \boldsymbol{\Delta}_A\left(\mathcal{X}, \mathcal{Z}; \boldsymbol{w}\right)$.

5.4.3 Maximum Margin Set (MMS)

Using the square norm regularizer as in the SVM and the loss function (5.2), we have the following optimization problem:

$$\min_{\boldsymbol{w}} \quad \frac{\lambda}{2}\|\boldsymbol{w}\|_2^2 + \frac{1}{N}\sum_{i=1}^{N} \ell_A\left(\mathcal{X}_i, \mathcal{Z}_i; \boldsymbol{w}\right) . \tag{5.3}$$

This optimization problem is non-convex due to the second $\max(\cdot)$ inside the loss (5.2). To convexify this problem, one could approximate the second $\max(\cdot)$ with the average over all labeling vectors in $\mathcal{Z}_i$. Similar strategies have been used in analogous problems (Cour *et al.*, 2009; Zhang and Zhou, 2008). However, the approximation could be very loose if the number of labeling vectors is large. Fortunately, although the loss function is not convex, a good local minimum can be found using the constrained concave-convex procedure (CCCP) (Smola *et al.*, 2005; Yuille and Rangarajan, 2003).

5.4.4 Optimizing the MMS problem with CCCP

To optimize (5.2) using CCCP, we first rewrite it as

$$\begin{aligned} \min_{\boldsymbol{w}} \quad & \frac{\lambda}{2}\|\boldsymbol{w}\|_2^2 + \frac{1}{N}\sum_{i=1}^{N} \xi_i \\ \text{s.t.} \quad & \max_{\bar{\boldsymbol{Z}} \notin \mathcal{Z}_i} \left(\boldsymbol{\Delta}_A(\bar{\boldsymbol{Z}}, \mathcal{Z}_i) + \mathbf{S}_{\boldsymbol{w}}(\mathcal{X}_i, \bar{\boldsymbol{Z}})\right) - \max_{\boldsymbol{Z} \in \mathcal{Z}_i} \mathbf{S}_{\boldsymbol{w}}(\mathcal{X}_i, \boldsymbol{Z}) \leq \xi_i, \quad \xi_i \geq 0, \quad i = 1, \ \ldots, \ N \ . \end{aligned}$$

In this formulation, the objective function is convex, while the first set of constraints can be written as the difference of a convex function and a concave function. The CCCP solves the optimization problem using an iterative minimization process. At each round r, given an initial $\boldsymbol{w}^{(r)}$, the CCCP replaces the concave part of the constraints with its first-order Taylor expansion at $\boldsymbol{w}^{(r)}$, and then sets $\boldsymbol{w}^{(r+1)}$ to the solution of the relaxed constrained optimization problem. When this function is non-smooth, such as $\max_{\boldsymbol{z}\in\mathcal{Z}_i} \mathbf{S}_{\boldsymbol{w}}(\mathcal{X}_i, \boldsymbol{Z})$ in our formulation, the gradient in the Taylor expansion must be replaced by the subgradient[1]. Thus, at the r-th round, the CCCP replaces $\max_{\boldsymbol{Z}\in\mathcal{Z}_i} \mathbf{S}_{\boldsymbol{w}}(\mathcal{X}_i, \boldsymbol{Z})$ by

$$\max_{\boldsymbol{Z}\in\mathcal{Z}_i} \mathbf{S}_{\boldsymbol{w}^{(r)}}(\mathcal{X}_i, \boldsymbol{Z}) + (\boldsymbol{w} - \boldsymbol{w}^{(r)}) \cdot \partial \left(\max_{\boldsymbol{Z}\in\mathcal{Z}_i} \mathbf{S}_{\boldsymbol{w}}(\mathcal{X}_i, \boldsymbol{Z}) \right) . \tag{5.4}$$

The subgradient of a point-wise maximum function $g(\boldsymbol{x}) = \max_i g_i(\boldsymbol{x})$ is the convex hull of the union of subdifferentials of the subset of the functions $g_i(\boldsymbol{x})$ which equal $g(\boldsymbol{x})$ (Bertsekas, 2003). Defining by $\mathcal{C}_i^{(r)} = \{\boldsymbol{Z} \in \mathcal{Z}_i : \mathbf{S}_{\boldsymbol{w}^{(r)}}(\mathcal{X}_i, \boldsymbol{Z}) = \max_{\boldsymbol{z}'\in\mathcal{Z}_i} \mathbf{S}_{\boldsymbol{w}^{(r)}}(\mathcal{X}_i, \boldsymbol{Z})\}$, the subgradient of the function $\max_{\boldsymbol{Z}\in\mathcal{Z}_i} \mathbf{S}_{\boldsymbol{w}}(\mathcal{X}_i, \boldsymbol{Z})$ equals to $\sum_l \alpha_{i,l}^{(r)} \partial \mathbf{S}_{\boldsymbol{w}}(\mathcal{X}_i, \boldsymbol{Z}_{i,l}) = \sum_l \alpha_{i,l}^{(r)} \Phi(\mathcal{X}_i, \boldsymbol{Z}_{i,l})$, with $\sum_l \alpha_{i,l}^{(r)} = 1$, and $\alpha_{i,l}^{(r)} \geq 0$ if $\boldsymbol{Z}_{i,l} \in \mathcal{C}_i^{(r)}$ and $\alpha_{i,l}^{(r)} = 0$ otherwise. Hence we have

$$\sum_l \alpha_{i,l}^{(r)} \boldsymbol{w}^{(r)} \cdot \Phi(\mathcal{X}_i, \boldsymbol{Z}_{i,l}) = \max_{\boldsymbol{Z}\in\mathcal{Z}_i} \left(\boldsymbol{w}^{(r)} \cdot \Phi(\mathcal{X}_i, \boldsymbol{Z}) \right) \sum_{l:\boldsymbol{Z}_{i,l}\in\mathcal{C}_i^{(r)}} \alpha_{i,l}^{(r)} = \max_{\boldsymbol{Z}\in\mathcal{Z}_i} \left(\boldsymbol{w}^{(r)} \cdot \Phi(\mathcal{X}_i, \boldsymbol{Z}) \right) .$$

Combining this with (5.4), the constraints become

$$\max_{\bar{\boldsymbol{Z}}\notin\mathcal{Z}_i} \left(\boldsymbol{\Delta}_A(\bar{\boldsymbol{Z}}, \mathcal{Z}_i) + \boldsymbol{w} \cdot \Phi(\mathcal{X}_i, \bar{\boldsymbol{Z}}) \right) - \boldsymbol{w} \cdot \sum_{\boldsymbol{Z}_{i,l}\in\mathcal{C}_i^{(r)}} \alpha_{i,l}^{(r)} \Phi(\mathcal{X}_i, \boldsymbol{Z}_{i,l}) \leq \xi_i .$$

Hence the relaxed convex optimization program at the r-th round of the CCCP is equivalent to the problem

$$\min_{\boldsymbol{w}} \quad \frac{\lambda}{2}\|\boldsymbol{w}\|_2^2 + \frac{1}{N}\sum_{i=1}^{N} \ell_{\text{cccp}}^{(r)}(\mathcal{X}_i, \mathcal{Z}_i; \boldsymbol{w}) , \tag{5.5}$$

where

$$\ell_{\text{cccp}}^{(r)}(\mathcal{X}_i, \mathcal{Z}_i; \boldsymbol{w}) = \Big| \max_{\bar{\boldsymbol{Z}}\notin\mathcal{Z}_i} \left(\boldsymbol{\Delta}_A(\bar{\boldsymbol{Z}}, \mathcal{Z}_i) + \boldsymbol{w} \cdot \Phi(\mathcal{X}_i, \bar{\boldsymbol{Z}}) \right) - \boldsymbol{w} \cdot \sum_{\boldsymbol{Z}_{i,l}\in\mathcal{C}_i^{(r)}} \alpha_{i,l}^{(r)} \Phi(\mathcal{X}_i, \boldsymbol{Z}_{i,l}) \Big|_+ .$$

---

[1] Given a function g, its subgradient $\partial g(\boldsymbol{x})$ at $\boldsymbol{x}$ satisfies: $\forall \boldsymbol{u}, g(\boldsymbol{u}) - g(\boldsymbol{x}) \geq \partial g(\boldsymbol{x}) \cdot (\boldsymbol{u} - \boldsymbol{x})$. The set of all subgradients of g at $\boldsymbol{x}$ is called the subdifferential of g at $\boldsymbol{x}$.

Algorithm 8 The CCCP algorithm for solving MMS

```
initialize: $\boldsymbol{w}^{(1)} = \mathbf{0}$
repeat
    Set $\mathcal{C}_i^{(r)} = \{\boldsymbol{Z} \in \mathcal{Z}_i : \mathbf{S}_{\boldsymbol{w}^{(r)}}(\mathcal{X}_i, \boldsymbol{Z}) = \max_{\boldsymbol{Z}' \in \mathcal{Z}_i} \mathbf{S}_{\boldsymbol{w}^{(r)}}(\mathcal{X}_i, \boldsymbol{Z}')\}$
    Set $\boldsymbol{w}^{(r+1)}$ as the solution of the convex optimization problem (5.5) (Algorithm 9)
until convergence to a local minimum
output:$\boldsymbol{w}^{(r+1)}$
```

The procedure of the CCCP algorithm is outlined in Algorithm 8. It is guaranteed to decrease the objective function and it converges to a local minimum solution of problem (5.3) (Smola *et al.*, 2005; Yuille and Rangarajan, 2003).

We are free to choose the values of the $\alpha_{i,l}^{(r)}$ in the convex hull. Since the algorithm is susceptible to a local minima, its performance could possibly be sensitive to initialization. Here we choose to set $\alpha_{i,l}^{(r)} = 1/|\mathcal{C}_i^{(r)}|$ for $\forall \boldsymbol{z}_{i,l} \in \mathcal{C}_i^{(r)}$. With our choice of $\alpha_{i,l}^{(r)}$, in the first round of the CCCP when $\boldsymbol{w}$ is initialized at $\mathbf{0}$, the second $\max(\cdot)$ in (5.2) is approximated by the average over all the labeling vectors. In this way, the first round of the algorithm is similar to the convex relaxation methods in Cour *et al.* (2009); Zhang and Zhou (2008), but here the later iterations will improve the solution.

5.4.5 Solving the relaxed MMS optimization problem using the Pegasos framework

In order to solve the relaxed convex optimization problem (5.5) efficiently at each round of the CCCP, we have designed a stochastic subgradient descent algorithm, using the Pegasos framework (Shalev-Shwartz *et al.*, 2007). At each step the algorithm takes K random samples from the training set and calculates an estimate of the subgradient of the objective function using these samples. Then it performs a subgradient descent step with decreasing learning rate, followed by a projection of the solution into the space where the optimal solution lives (line 7). An upper bound on the radius of the ball in which the optimal hyperplane lives can be calculated by considering that

$$\frac{\lambda}{2}\|\boldsymbol{w}^*\|_2^2 \leq \min_{\boldsymbol{w}} \ \frac{\lambda}{2}\|\boldsymbol{w}\|_2^2 + \frac{1}{N}\sum_{i=1}^{N} \ell_{\text{cccp}}^{(r)}\left(\mathcal{X}_i, \mathcal{Z}_i; \boldsymbol{w}\right) \leq B \ ,$$

where $\boldsymbol{w}^*$ is the optimal solution of problem (5.5), with $B = \max_i(\ell_{\text{cccp}}^{(r)}(X_i, Z_i; \mathbf{0}))$, which equals the maximum number of regions in the image multiplied by the number of attributes P we model. The

Algorithm 9 Pegasos algorithm for solving the relaxed MMS problem

```
1: input: $\boldsymbol{w}_0$, $\{\mathcal{X}_i, \boldsymbol{Z}_i, \mathcal{C}_i^{(r)}\}_{i=1}^N$, $\lambda$, $T$, $K$, $B$
2: for $t = 1, 2, \ldots, T$ do
3:     Draw at random $A_t \subseteq \{1, \ldots, N\}$, with $|A_t| = K$
4:     Compute $\hat{\boldsymbol{Z}}_k = \arg\max_{\bar{\boldsymbol{Z}} \notin \mathcal{Z}_k} \big(\boldsymbol{\Delta}_A(\bar{\boldsymbol{Z}}, \mathcal{Z}_k) + \boldsymbol{w}_t \cdot \Phi(\mathcal{X}_k, \bar{\boldsymbol{Z}})\big) \qquad \forall k \in A_t$
5:     Set $A_t^+ = \{k \in A_t : \ell_{\text{cccp}}^{(r)}(\mathcal{X}_k, \mathcal{Z}_k; \boldsymbol{w}_t) > 0\}$
6:     Set $\boldsymbol{w}_{t+\frac{1}{2}} = (1 - \frac{1}{t})\boldsymbol{w}_t + \frac{1}{\lambda K t} \sum_{k \in A_t^+} \big(\sum_{\boldsymbol{Z} \in \mathcal{C}_i^{(r)}} \Phi(\mathcal{X}_k, \boldsymbol{Z})/|\mathcal{C}_i^{(r)}| - \Phi(\mathcal{X}_k, \hat{\boldsymbol{Z}}_k)\big)$
7:     $\boldsymbol{w}_{t+1} = \min\Big(1, \sqrt{2B/\lambda}/\|\boldsymbol{w}_{t+\frac{1}{2}}\|\Big)\, \boldsymbol{w}_{t+\frac{1}{2}}$
8: end for
9: output: $\boldsymbol{w}_{T+1}$
```

details of the Pegasos algorithm for solving problem (5.5) are given in Algorithm 9. Using the theorems in Shalev-Shwartz *et al.* (2007) it is easy to show that after $\widetilde{\mathcal{O}}\big(1/(\lambda\varepsilon)\big)$ iterations Algorithm 9 converges in expectation to a solution of accuracy ε.

Efficient implementation. At each iteration of Algorithm 9, step 4 searches for the most violating labeling vector $\hat{\boldsymbol{Z}}_k$, which is typically computationally expensive. Dynamic programming can be carried out to reduce the computational cost since the contribution of each instance is additive over different labels. In the general situation, the worst case complexity of a naive implementation which enumerates all the possible permutations is $\mathcal{O}(\prod_{m=1}^{M_i} K_{i,m})$, where $K_{i,m}$ is the number of unique possible labels for $\boldsymbol{x}_{i,m}$ in $\mathcal{Z}_i$ (usually $K_{i,m} \ll L_i$). However, the computation time can be further reduced by exploiting the structure of $\mathcal{Z}_i$. This complexity can be greatly reduced when there are special structures such as graphs and trees in the CLSs. See for example (Tsochantaridis *et al.*, 2005, Section 4) for a discussion on some specific problems and special cases. In Section 5.6.2, we will present an efficient inference algorithm specialized for solving the name association problem. In cases when computing an exact solution is too expensive, it is often possible to calculate an approximate solution of the same problem, and obtain good empirical results (Wang and Mori, 2010) with theoretical guarantees (Finley and Joachims, 2008).

5.5 Experiments on artificial data

In order to evaluate the proposed algorithm, we first perform experiments on several artificial datasets created from four widely used multiclass datasets taken from the LIBSVM (Chang and Lin, 2001) website (usps, letter, news20 and covtype).

The artificial training sets are created as follows: we first set at random pairs of classes as "correlated classes", and as "ambiguous classes", where the ambiguous classes can be different from the correlated classes. Following that, instances are grouped randomly into bags of fixed size B with probability at least P_c that two instances from correlated classes will appear in the same bag. Then L ambiguous labeling vectors are created for each bag, by modifying a few elements of the correct labeling vector. First, the number of elements to modify b is randomly chosen from $\{1, \ldots, B\}$. Then b instances are randomly chosen from the bag, and new labels are randomly chosen among a predefined ambiguous set. The ambiguous set contains the other correct labels from the same bag (except the true one) and a subset of the ambiguous pairs of all the correct labels from the bag. The probability of whether the ambiguous pair of a label is present equals P_a. For testing, we use the original test set, and each instance is considered separately.

Varying P_c, P_a, and L we generate datasets with different difficulty levels to evaluate the behaviour of the algorithms. For example, when $P_a > 0$, noisy labels are likely to be present in the labeling set. Meanwhile, P_c controls the ambiguity within a bag. If P_c is large, instances from two correlated classes are likely to be grouped into the same bag, thus it becomes more difficult to distinguish between them. The parameters P_c and P_a are chosen from $\{0, 0.25, 0.5\}$. For each difficulty level, we use 3 random training/test splits.

For our algorithm, we set the regularization parameter λ to $1/N$ in all of our experiments. We benchmark MMS against the following baselines:

SVM: we train a fully-supervised SVM classifier using the ground-truth labels by considering every instance separately. Its performance is an upper bound of the performance using candidate labeling sets. In all experiments, we use the LIBLINEAR (Fan *et al.*, 2008) package and test two different multiclass extensions, the 1-vs-All method with the hinge loss ℓ^{HL} (1vA-SVM) and the method by Crammer and Singer (Crammer and Singer, 2002) (MC-SVM) using the multiclass loss ℓ^{MC}.

CL-SVM: the Candidate Labeling SVM (CL-SVM) is a naive approach which transforms the ambiguously labeled data into a standard supervised representation by treating all possible labels of each instance as true labels. CL-SVM then learns K separate 1-vs-All SVM classifiers from the resulting dataset, where the negative examples for the y-th classifier are instances which do not have the corresponding label y in their candidate labeling set, in other words, it is not possible for these instances to come from class y. A similar baseline has been used in the two-class MIL literature (Bunescu and Mooney, 2007).

MIML: we also compared with two SVM-based MIML algorithms[2]: MIMLSVM (Zhou and Zhang, 2006) and M^{3}MIML (Zhang and Zhou, 2008). We trained the MIML algorithms by treating the labels in $\mathcal{Z}_i$ as a label for the bag. During the test phase, we consider each instance separately and predict the labels as: $y = \arg\max_{y \in \mathcal{Y}} F_{\text{miml}}(\boldsymbol{x}, y)$, where F_{miml} is the classifier learned during training, and $F_{\text{miml}}(\boldsymbol{x}, y)$ can be interpreted as the confidence of the classifier in assigning label y to instance $\boldsymbol{x}$. For a fair comparison, we use the linear kernel in all methods. The cost parameter for SVM algorithms is selected from the range $C \in \{0.1, 1, 10, 100, 1000\}$, and the best results are reported. The bias term is used in all algorithms.

In fig. 5.3, we plot the average classification accuracy. Several observations can be made. First, MMS achieves results close to the supervised SVM methods, and better than all other baselines. As MMS uses a similar multiclass loss as MC-SVM, it even outperforms 1vA-SVM when the loss has its advantage (*e.g.,* on the 'letter' dataset). For the 'covtype' dataset, the performance gap between MMS and SVM is more visible. It may be due to the fact that 'covtype' is a class unbalanced dataset, where the two largest classes (among seven) dominate the whole dataset (more than 85% of the total number of samples). Second, the change in performance of MMS is small when the size of the candidate labeling set grows. Moreover, when correlated instances and extra noisy labels are present in the dataset, the baseline methods' performance drops significantly, whereas MMS is less affected.

The CCCP algorithm usually converges in $3-5$ rounds, and the final performance is about $5\% - 40\%$ higher compared to the results obtained after the first round, especially when L is large. This behavior also proves that approximating the second $\max(\cdot)$ function in the loss function (5.2) with the average over all the possible labeling vectors can lead to poor performance.

5.6 Real case 1: Who is in the picture?

The first real-world problem we tackle is naming faces in news images accompanied by captions written by journalists (Berg *et al.*, 2004b; Guillaumin *et al.*, 2010). Thanks to recent developments in the computer vision and natural language processing fields, generic faces can be localized in the images using a face detector (Viola and Jones, 2004) and generic names can be localized in the captions using a named entity

---

[2] We used the original implementation at `http://lamda.nju.edu.cn/data.ashx#code`. We did not compare against MIMLBOOST (Zhou and Zhang, 2006), because it does not scale to all the experiments we conducted. Besides, MIMLSVM (Zhou and Zhang, 2006) does not scale to data with high dimensional feature vectors (*e.g.,* , news20 which has a 62,061-dimensions features). Running the MATLAB implementation of M^{3}MIML (Zhang and Zhou, 2008) on problems with more than a few thousand samples is computational infeasible. Thus, we will only report results using this two baseline methods on small size problems, where they can be finished in a reasonable amount of time.

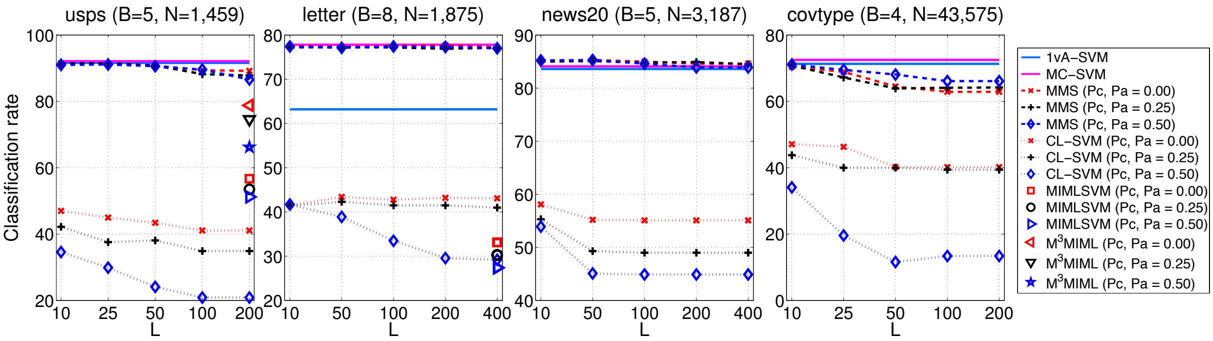


Figure 5.3. Classification performance of different weakly supervised learning and supervised learning algorithms on four artificial datasets.

detector (OpenNLP, 2010). Because the names of the most important persons in the image typically appear in the caption, we can attempt to *automatically* label the detected faces with their correct names. This enables to gather a large and realistic face dataset as well as learning face classifiers directly from news items, saving the effort of manually labeling the faces. The main challenge of this task is the *correspondence ambiguity*: there could be multiple faces in the image and/or multiple names in the caption, and not all the names in the caption appear in the image, and vice versa. Some of the face detections can be false positives, which adds to the ambiguity. The task of an algorithm is to resolve the correspondence ambiguity, *i.e.,* assign a name from the caption to each face in the image (or the *null* label if a person is not mentioned in the caption, or for false positive detections).

5.6.1 Modeling

Since in this problem we are only interested in learning face classifiers, the model will associate at most one label (a name) to each detected face region (no other attributes). In practice, there can be thousands of different names in real world datasets (*e.g.,* the whole of Yahoo! News Dataset). However, users are typically only interested in the top K most frequent names, or only in a limited number of celebrities. Moreover, because of the ambiguities mentioned above, we add a label called *null* (this also covers for those infrequent names we do not model). In total, there are $K+1$ classes, including K face classifiers to be learned.

For each detected face in an image we want to assign a name from the caption to it, or *null*. To format this problem into our framework, we use the following constraints to generate the CLSs:

- [A] a face can be assigned to exactly one name or to *null*;

- [B] a name can be assigned to at most one face;
- [C] a face can be assigned only to a name appearing in the caption of the corresponding image;
- [D] if spatial indicators, such as "left" and "(L)", exist, the labeling vectors in the CLSs should respect them.

In our setup, constraints [A], [B] and [C] apply to all news items, whereas constraint [D] applies only to a few items. When including constraints [A]-[C], the number of admissible candidate labeling vectors for an image-caption pair with M_i faces and W_i names is $L^i = \sum_{j=0}^{\min(M_i,W_i)} \binom{M_i}{j} \cdot \binom{W^i}{j}$ (see Figure 5.4 for an example with $M_i = W_i = 2$). In addition, when a spatial indicator is available, we remove the labeling vectors which do not comply with it. Take Figure 5.2 for example: we decide the relative position of faces by the horizontal coordinate of their center. With the spatial indicators, we know that the face in the middle can only be Grant Hackett. Then the face on the left can either be Igor Chervynsky or Erik Vendt (and the same for the face or the right). Their CLS can be generated as the two persons and two faces case. Different from previous methods, we do not allow the labeling vector which assigns all faces to *null*, because classifying every face as *null* would lead to the trivial solution with 0 loss.

5.6.2 Inference

As stated above, searching for $\hat{\boldsymbol{Z}}$ in line 4 of Algorithm 9 is the most expensive step of the training procedure. Here we propose an algorithm which can find $\hat{\boldsymbol{Z}}$ efficiently. We first compute the classification scores $s_{\boldsymbol{w}}(x_m, y)$ for every instances in the bag and every possible label $y \in \mathbb{Y}$. After the scores are computed, we use Algorithm 10 to find $\hat{\boldsymbol{Z}}$ in polynomial time with a bounded number of iterations.

The design of Algorithm 10 is motivated by the A* search algorithm (Cormen *et al.*, 2003). The algorithm first ranks the possible predictions y_m for each face $\boldsymbol{x}_m$ according to their scores $s_{\boldsymbol{w}}(\boldsymbol{x}_m, y_m) + \Delta_A(y, \mathcal{Z}(m))$, where $\mathcal{Z}(m)$ is the m-th row of $\mathcal{Z}$. Lines 9-14 of the algorithm guarantee that all elements in the heap H have a higher score S than any $S(\mathcal{X}, \boldsymbol{Z}) = \sum_{\boldsymbol{x}_m \in \mathcal{X}, z_m \in \boldsymbol{Z}} s_{\boldsymbol{w}}(\boldsymbol{x}_m, z_m)$ for any other arbitrary compositions of $\boldsymbol{Z}$ with $z_m \in \mathbb{Y}$, which not have been added into H. It is easy to verify that the algorithm terminates in at most $L + 1$ iterations, where L is the number of candidate labeling vectors in $\mathcal{Z}$. The worst case scenario is when the first L $\bar{\boldsymbol{Z}}$s returned by the heap (line 8) all belong to $\mathcal{Z}$. As typically $L \gg K$, the worst case complexity of searching for $\hat{\boldsymbol{Z}}$ using Algorithm 10 is $\mathcal{O}(L * M)$, where M is the

Algorithm 10 Efficient Algorithm for Searching for $\hat{\boldsymbol{Z}}$

```
input: $\mathcal{Z},\ s(x_m, y) + \Delta_A(y, \mathcal{Z}(m)),\ \forall\, m = 1,\dots,M,\ y \in \mathbb{Y}$
initialize: $H$ = new heap,  index variable $j_m = 1,\ \forall m$
Set $\boldsymbol{X}_m$ as a sorted array of $y$ in descending order, according to $s(x_m, y) + \Delta_A(y, \mathcal{Z}_m)$ ,  $\forall y \in \mathbb{Y}$
Set $\boldsymbol{Z} = [\boldsymbol{X}_1(j_1), \dots, \boldsymbol{X}_M(j_m)]$
Set $S = \sum_m s(x_m, \boldsymbol{Z}(m))$
Push $\mathcal{A} = \{\boldsymbol{j} = [j_1,\ \dots,\ j_M],\ \boldsymbol{Z},\ S\}$ into $H$
repeat
  Pop $\mathcal{A} = \{\boldsymbol{j}, \bar{\boldsymbol{Z}}, S\}$ with the highest score $S$ out of $H$
  for $m = 1, 2, \dots, M$ do
    Set $\boldsymbol{j}' = \boldsymbol{j}$, then $\boldsymbol{j}'(m)$++
    Set $\boldsymbol{Z}' = [\boldsymbol{X}_1(\boldsymbol{j}'(1)),\ \dots,\ \boldsymbol{X}_M(\boldsymbol{j}'(M))]$
    Set $S = \sum_{m'} s(x_{m'}, \boldsymbol{Z}'(m'))\}$
    Push $\mathcal{A}' = \{\boldsymbol{j}',\ \boldsymbol{Z}',\ S\}$ into $H$
  end for
until $\bar{\boldsymbol{Z}} \notin \mathcal{Z}$
output: $\hat{\boldsymbol{Z}} = \bar{\boldsymbol{Z}}$
```

number of regions in an image. Hence, Algorithm 10 can be used to obtain $\hat{\boldsymbol{Z}}$ efficiently when the value of L and B is not very large ($L \leq 10^4$ and $M \leq 10$), which is the case in this task.

In general, Algorithm 10 can not guarantee to find an exact solution $\hat{\boldsymbol{Z}}$ to the problem $\arg\max_{\bar{\boldsymbol{Z}} \notin \mathcal{Z}} U(\bar{\boldsymbol{Z}}) = \arg\max_{\bar{\boldsymbol{Z}} \notin \mathcal{Z}} \big(\boldsymbol{\Delta}_A(\bar{\boldsymbol{Z}}, \mathcal{Z} + \boldsymbol{w}_t \cdot \Phi(\mathcal{X}_k, \bar{\boldsymbol{Z}}))\big)$, denoted by $\hat{\boldsymbol{Z}}^*$, at every round. This is because the algorithm only considers $\boldsymbol{\Delta}_A(y, \mathcal{Z}(m)) = 1$ for those labels that do not appear in the $\mathcal{Z}(m)$. Let us consider a more concrete example: assume a bag $\{\boldsymbol{x}_1, \boldsymbol{x}_2\}$ with two labeling vectors $\boldsymbol{z}_1 = [1, 2]$ and $\boldsymbol{z}_2 = [2, 1]$. We also know that $s(\boldsymbol{x}_1, 1) = 3$, $s(\boldsymbol{x}_1, 2) = 2$, $s(\boldsymbol{x}_1, 3) = 1$, $s(\boldsymbol{x}_2, 1) = s(\boldsymbol{x}_2, 2) = 1$ and $s(\boldsymbol{x}_2, 3) = 0.99$. In this case, the solution obtained by Algorithm 10 is $\hat{\boldsymbol{Z}} = [1,\ 3]$, but $\hat{\boldsymbol{Z}}^* = [1, 1]$. Despite that, the algorithm will still obtain a $\hat{\boldsymbol{Z}}$ whose value of $U(\hat{\boldsymbol{Z}})$ is very close to $U(\hat{\boldsymbol{Z}}^*)$. However, in practice, the algorithm works very well, and the above special case rarely happens. Experiments on the same dataset show that Algorithm 10 almost always find $\hat{\boldsymbol{Z}}^*$, and that MMS executed with Algorithm 10 achieves the same performance as using an exponential time exact inference algorithm.

5.6.3 Experiments

We conducted experiments on the Labeled Yahoo! News dataset (see Appendix A for more details). We compare our results to the same baselines proposed in Section 5.5. In MIMLSVM, *null* faces are automatically considered as negative instances. In addition, we also compare with a baseline which does not consider the appearance of the faces:

RANDOM: randomly assign a name (or *null*) from the caption to each face in the corresponding image.

Table 5.1. Protocol I -Overall name assignment accuracy on the training set

Method	RANDOM	CL-SVM	MIMLSVM	MMS
Accuracy	73.1% ± 0.0	86.3% ± 0.1	89.62% ± 0.2	**91.86% ± 0.3**

Table 5.2. Protocol I - Overall face recognition accuracy on the test set

		Supervised Learning			Weakly supervised Learning		
Method	RANDOM	1vA-SVM	MC-SVM	MC-SVM[50%]	CL-SVM	MIMLSVM	MMS
Accuracy	66.0% ± 0.0	81.6% ± 0.6	87.2% ± 0.3	83.3% ± 0.2%	76.9% ± 0.2	74.7% ± 0.9	**85.7% ± 0.5**

Although there are more than 10000 different names in the captions, we will only consider those names which appear frequently enough. We consider two different protocols, detailed in the following sections:

Protocol I

In the first set of experiments, we use only constraints [A]-[C] to generate the CLS $\mathcal{Z}_i$ of each image-caption pair. We retain the 214 names occurring at least 20 times, and treat the other names as *null*. The experiments are performed over 5 different random train/test splits, sampling 80% of the items as training set and using the rest for testing. During splitting we also maintain the ratio between the number of samples from each class in the training and test set. Performance is measured by how many faces in the test set are correctly labeled with their name (or *null*). Moreover, we also compute name assignment performance on the training set by testing the final model on it.

Table 5.1 reports the name assignment performance on the training set. All the weakly supervised learning algorithms which consider face appearance outperform the RANDOM baseline, and MMS achieves the best result among all approaches. Table 5.2 summarizes the generalization performance on the test set. Several observations can be made. First, MMS achieves performance comparable to the fully-supervised SVM algorithms (1vA-SVM, MC-SVM), and it outperforms the other methods which train from ambiguously labeled data (*i.e.,* the captions). MMS even achieves an accuracy 4% higher than 1vA-SVM. This gain may be due to the fact that MMS uses a similar multiclass loss as MC-SVM, whose formulation is advantageous on this dataset. Moreover, we also present the result of MC-SVM trained on only half of the training data (MC-SVM[50%]), while evaluating on the same test set. The result shows that when MMS has more training data, it even outperforms the best fully supervised learning method we consider. This illustrates the promise of our method, as large amounts of image-caption pairs can be easily obtained from the internet, without manual labeling efforts.

Table 5.3. Protocol II -Overall name assignment accuracy on the training set

	Without POS		With POS	
Method	RANDOM	MMS	RANDOM	MMS
Accuracy	37.0% ± 0.0	85.3% ± 0.7	70.8% ± 0.0	**89.1% ± 0.7**

Table 5.4. Protocol II - Overall face recognition accuracy on the test set

Method	RANDOM	1vA-SVM	MC-SVM	MMS (Without POS)	MMS (With POS)
Accuracy	39.5% ± 0.0	82.2% ± 0.2	87.3% ± 0.1	83.5% ± 0.4	**84.9% ± 0.5**

Protocol II

Here we consider all four constraints including the spatial indicators [D] (Section 5.6.1). Only 3105 image-caption pairs contain any spatial indicator. We use all of them as part of the training set. In addition, we also randomly sample 6895 image-caption pairs from the dataset, resulting in a training set of 10000 image-caption pairs. All other image-caption pairs form the test set. We retain the 460 names occurring at least 3 times, and treat the other names as *null.*

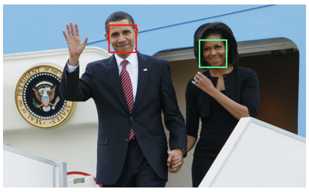
The results are reported in table 5.3 and 5.4. Our MMS algorithm can take advantage of the spatial indicators, and improve performance for both name association on the training set (+3.8%) and face recognition on the test set (+1.4%).

5.7 Real case 2: Who is doing what?

The second real-world problem we tackle is finding out "who's doing what", *i.e.,* associating names and action verbs in the captions to the faces and body poses of the persons in the images. In addition, the algorithm should also learn visual appearance models for the face and pose classes jointly. This task generalizes the work described in the previous section by considering the subject-verb language construct and by modeling names and verbs jointly. In our previous work (Jie *et al.*, 2009b), we have shown that the correspondence ambiguity is reduced by jointly modeling face and pose together using a generative model. In this section, we show that our MMS technique can be used to model the same problem, and achieves better performance than (Jie *et al.*, 2009b).

5.7.1 Modeling

In this task, the corpus of news items contains still images of persons performing actions. Each image is annotated with a caption describing "who's doing what" in the image (Figure 5.1). The interesting



President ***Barack Obama*** *and first lady* ***Michelle Obama*** *wave from the steps of Air Force One as they arrive in Prague, Czech Republic.*

$$\begin{array}{c|c|c|c|c|c|c|l} & \boldsymbol{Z}_1 & \boldsymbol{Z}_2 & \boldsymbol{Z}_3 & \boldsymbol{Z}_4 & \boldsymbol{Z}_5 & \boldsymbol{Z}_6 & \\ \mathcal{Z}: & \mathrm{n}_a & \mathrm{n}_a & \circ & \mathrm{n}_b & \circ & \mathrm{n}_b & \leftarrow \text{face}_1 \\ & \mathrm{n}_b & \circ & \mathrm{n}_b & \mathrm{n}_a & \mathrm{n}_a & \circ & \leftarrow \text{face}_2 \end{array}$$

Figure 5.4. (Left): An example image with its associated caption. There are two detected faces face_1 and face_2 and two names Barack Obama (n_a) and Michelle Obama (n_b) from the caption. (Right): The CLS for this image-captions pair. The labeling vectors are generated using the constrain [A], [B] and [C], where the *null* class is denoted as $\circ$.

regions in the images are persons. A person corresponds to a face and upper-body (including false positive detections), which can be detected with available software. A face and an upper-body are considered to belong to the same person if the face lies near the center of the upper-body bounding-box. One could use a named entity detector (OpenNLP, 2010) and a language parser (Deschacht and Moens, 2009) to extract a list of name-verb pairs from each caption, to represent the connection between a subject and its verb in a sentence. If a name is not connected to any verb, the pair is name-null. Our system models two types of words jointly, and the goals are to: (i) associate the persons in the images to the name-verb pairs in the captions, and (ii) learn a visual appearance model for each name and each verb, corresponding to face and pose classes. These can be used for recognition on new images with or without caption.

The candidate labels for a detected person are the name-verb pairs in the caption. One label assigns a name to a face, and its connected verb from the caption to the body pose of the same person in the image. Hence, name and verb are seen as two attributes of the same image region (a person). Therefore, during learning, to find the best possible $\boldsymbol{Z} \in \mathcal{Z}$ for the image, the names and the verbs are considered jointly in making decisions. The chosen name-verb pair is the one which has the highest confidence score over both attributes. For generating constraints, we assume again the uniqueness of each person and her name and apply constraints analog to Section 5.6.1 for generating the CLVs (except [D], as spatial indicators are not available in the dataset we perform experiments on). More precisely, the *face* and *name* in constraints [A]-[C] are replaced by *person* and *name-verb*. In this way, each person region is associated with two attributes: name and verb, which must come from the same pair detected from the caption. Figure 5.5 illustrates how the CLVs are generated on an example with two persons in the image and two name-verb pairs in the caption (news item in Figure 5.1 (left)).

5.7.2 Inference

We use Algorithm 10 to compute $\hat{\boldsymbol{Z}}$.

$$\begin{array}{cc}
 & \begin{array}{cccccc} \boldsymbol{Z}_1 & \boldsymbol{Z}_2 & \boldsymbol{Z}_3 & \boldsymbol{Z}_4 & \boldsymbol{Z}_5 & \boldsymbol{Z}_6 \end{array} \\
\mathcal{Z}: & \left[\begin{array}{c|c|c|c|c|c}
\mathrm{n}_a & \mathrm{n}_a & \circ & \mathrm{n}_b & \circ & \mathrm{n}_b \\
\mathrm{v}_a & \mathrm{v}_a & \circ & \mathrm{v}_b & \circ & \mathrm{v}_b \\
\mathrm{n}_b & \circ & \mathrm{n}_b & \mathrm{n}_a & \mathrm{n}_a & \circ \\
\mathrm{v}_b & \circ & \mathrm{v}_b & \mathrm{v}_a & \mathrm{v}_a & \circ
\end{array}\right]
\begin{array}{l}
\leftarrow \text{person}_1\text{-face} \\
\leftarrow \text{person}_1\text{-pose} \\
\leftarrow \text{person}_2\text{-face} \\
\leftarrow \text{person}_2\text{-pose}
\end{array}
\end{array}$$

Figure 5.5. CLVs generation for the new item in Figure 5.1 (left). There are two detected persons, person_1 and person_2, and two name-verb pairs, Barack Obama-Waving (n_a-v_a) and Michelle Obama-Standing (n_b-v_b). The CLVs are generated using constraints [A]-[C] as in Section 5.6.1. Labels such as Barack Obama-Standing is not allowed, as Barack is not the subject of the verb "standing" in the caption.

5.7.3 Experiments

We conducted experiments on the Idiap/ETHZ Faces and Poses dataset which contains images of person(s) performing certain actions. For each person in the image, we extract a face descriptor and three types of body pose descriptor. We describe the face with the method of Everingham *et al.* (2006), which detects nine distinctive feature points within the face bounding box. Each point is represented by the pixels in an elliptical region around it, normalized for local photometric invariance. For describing the body poses, we use the features of Ferrari *et al.* (2008a). A pose E consists of a distribution over the position (spatial and orientation) for each of 6 body parts (head, torso, upper/lower left/right arms) output by the estimator of Eichner and Ferrari (2009). Three low-dimensional descriptors are derived from E (*e.g.*, the relative position between pairs of body parts). We use non-linear kernels for both face and pose descriptors, in the form $k(\boldsymbol{x}, \boldsymbol{x}') = \exp\left(-\gamma^{-1} d(\boldsymbol{x}, \boldsymbol{x}')\right)$, with d the distance between two descriptors, and γ selected by cross-validation. We measure the distance between two face descriptors $\boldsymbol{x}, \boldsymbol{x}'$ using $d_{\text{face}}(\boldsymbol{x}, \boldsymbol{x}') = 1 - \boldsymbol{x}^T \boldsymbol{x}' / (\|\boldsymbol{x}\| \|\boldsymbol{x}'\|)$. In (Ferrari *et al.*, 2008a), different similarity measures are proposed for each type of descriptor. We normalize the range of each similarity to $[0, 1]$, and denote their average as $s_{\text{pose}}(\boldsymbol{x}, \boldsymbol{x}')$. The final distance between two poses is $d_{\text{pose}}(\boldsymbol{x}, \boldsymbol{x}') = 1 - s_{\text{pose}}(\boldsymbol{x}, \boldsymbol{x}')$. The face and pose kernel matrices are computed in advance to speed up the learning. It is easy to verify that they are all Mercer kernels. In this experiment, we use the available ground-truth name-verb pairs from the captions directly (instead of running an NLP tool) for generating the CLSs.

We compare the results of MMS against (i) a simplified version of the constrained mixture model "GMM" (Berg *et al.*, 2004b, Section 2.3) which does not incorporate a language model of the caption; (ii) the distance-based generative model "DIST" (Jie *et al.*, 2009b); (iii) a "RANDOM" baseline which randomly assigns a name-verb pair from the captions to each region in the corresponding image. We did

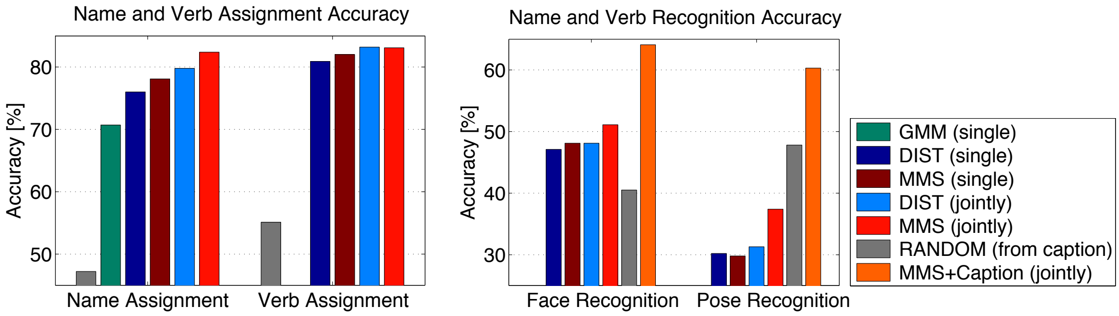


Figure 5.6. (Left) Name and verb assignment accuracy on 1600 training images from the Idiap/ETHZ face+pose database. 'Single' stands for modeling one attribute in its corresponding task. 'Jointly' stands for models using two attributes jointly. (Right) Name and verb recognition accuracy on the test images. All methods but 'MMS+Caption' and 'RANDOM (from caption)' only input the images without captions. The 'caption" methods input a test image-caption pair.

not compare to the other weakly-supervised learning baselines used in Section 5.5 & 5.6 because they do not support multiple attributes. As in the protocol of Jie *et al.* (2009b), we use 1600 items for training and 103 for testing.

Figure 5.7.3 (left) reports the name and verb assignment performance on the training set, while Figure 5.7.3 (right) reports the recognition results on the test images. We observe that: (i) DIST using only face information outperforms GMM for name-to-face assignment on the training set. This validates the quality of the distance-based appearance model of Jie *et al.* (2009b). We reuse it also in our MMS framework. (ii) MMS outperforms DIST on both the training and the set set. (iii) the joint "face and pose" model outperforms models using face or pose information alone, demonstrating that modeling both attributes jointly reduces the correspondence ambiguity on the training set and leads to appearance models which perform better on the test set. This phenomenon holds for both DIST and MMS. (iv) on the test set, MMS gets further performance gains when given captions. In this case the problem is easier because the correct label for a person is one of the few appearing in the caption.

5.8 Conclusion

In this chapter, we present an example on using text cue which accompanies the images to supervise the learning of visual classifiers without manual intervention. We show that such type of learning problem can be casted into a new weakly supervised learning framework. Compared to the weakly supervised learning algorithm proposed by Cour *et al.* (2009), our framework provides a principled way to encode different constraints widely used in many tasks, as a list of possible labelings which can be generated automatically from the image-caption pair. We also propose a large margin discriminative learning formulation and an efficient optimization algorithm to train the classifier effectively. We solve the non-convex objective function using the CCCP algorithm, which results in a tighter convex relaxation compared to the relaxation technique used in Cour *et al.* (2009); Zhang and Zhou (2008). We demonstrate on two real-world tasks that the proposed method can learn face classifiers from images with captions, and can learn multiple attributes jointly (names and verbs).

The MMS algorithm can be extended to solve other related problem. For example, in the multiple annotators scenario, where each data is associated with the labels given by independently hired annotators. The annotators can disagree on the data and the aim is to recover the true label of each sample. The use of this algorithm does not have to be limited to data which is naturally grouped in multi-instance bags. It could be also possible to group separate instances into bags and solve the learning problem using MMS, when there are labeling constraints between these instances (*e.g.,* a clustering problem with linkage constraints (Shental *et al.*, 2003)).

Chapter 6

Knowledge Transfer Across Different Cues

In this chapter, we present a multiclass transfer learning algorithm that allows to take advantage of priors built over different features and with different learning methods than the one used for learning the new task. The algorithm exploits learned models from potentially correlated tasks. These prior models are considered as experts, and the system transfers their outputs to the new incoming samples as additional information (Section 6.2). We cast the learning problem within the Multi Kernel Learning framework. The resulting formulation solves efficiently a joint optimization problem that determines from where and how much to transfer, with a principled multiclass formulation (Section 6.3). We call the proposed method Multi Kernel Transfer Learning (MKTL). Experiments show that the proposed approach can boost the performance when very few labeled data are available, while it still improves the performance when the algorithm has seen a reasonable amount of samples (Section 6.5). The content presented in this Chapter is based on the following publication:

> Jie, L., Tommasi, T., and Caputo, B., (2011). Multiclass Transfer Learning from Unconstrained Priors. In *Proceedings of the 13th International Conference on Computer Vision.*

6.1 Introduction

The artificial intelligent and machine learning community has shown a growing interest in transfer learning algorithms in the last few years. Indeed, this type of algorithms allows to exploit prior knowledge when learning a new class, which reduces the need for annotated training data. As the frontiers in object categorization move from systems able to categorize 10^2 objects (*e.g.,* Caltech-101 (Fei-Fei *et al.*, 2004) and Caltech-256 (Griffin *et al.*, 2007)) to systems aiming to recognize 10^4 categories (*e.g.,* ImageNet (Deng *et al.*, 2009)), there is a growing demand for techniques able to learn robust categorization models from few labeled samples.

Transfer learning has been studied in multiple domains and under various perspectives. Many works address the issue of what to transfer (samples (Bickel *et al.*, 2007), feature representation (Chang and Sridhar, 2008; Quattoni *et al.*, 2008), model parameters (Fei-Fei *et al.*, 2004; Stark *et al.*, 2009; Tommasi *et al.*, 2010)), some focus on how to transfer (generative approaches (Fei-Fei *et al.*, 2004; Stark *et al.*, 2009), boosting (Yao and Doretto, 2010), KNN (Saenko *et al.*, 2010) and SVM (Duan *et al.*, 2009; Tommasi *et al.*, 2010)), while others concentrate on how to avoid negative transfer, evaluating when and how much to transfer (different source selection approaches (Tommasi *et al.*, 2010) or methods to measure the task relatedness (Eaton *et al.*, 2008)). Some knowledge transfer strategies propose to exploit sets of unlabeled target samples (Quattoni *et al.*, 2008; Raina *et al.*, 2007) or alternative sources of extra information as attributes (Farhadi *et al.*, 2009; Lampert *et al.*, 2009).

As diverse as these approaches are, they all assume a strong control over the priors, whether in the form of constraining how the prior models are built (Fei-Fei *et al.*, 2004; Tommasi *et al.*, 2010), or in the way of preserving the priors training samples (Dai *et al.*, 2009; Daumé III, 2007), or in the form of imposing the same feature representation for all priors and for the new target class (Daumé III, 2007; Tommasi *et al.*, 2010). These constraints become particularly strict when the target problem is multiclass (Rohrbach *et al.*, 2010; Tang *et al.*, 2010).

Our contribution is a multiclass transfer learning algorithm from unconstrained priors. We assume to have no control on the features from which prior models are learned, nor on the learning methods used to build the corresponding classifiers. This is achieved by using the prior knowledge as experts evaluating the new incoming data and transferring their confidence outputs. These outputs are used to augment the feature space of the new target data. The learning process is defined solving an optimization problem

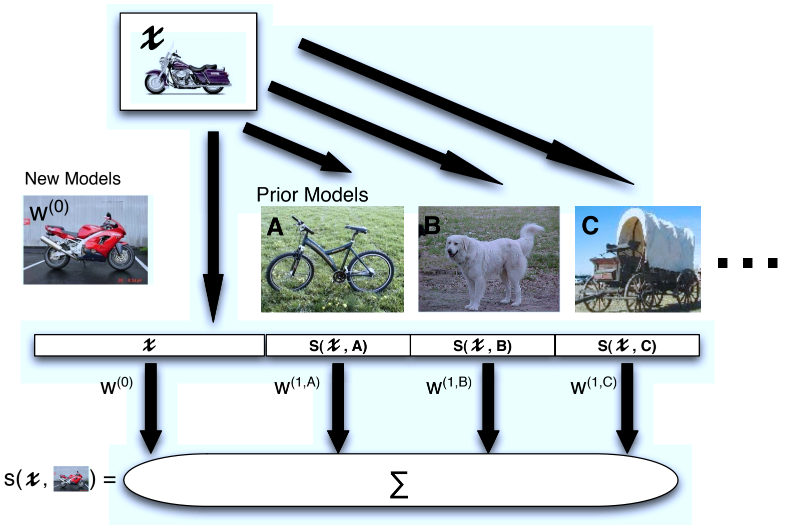


Figure 6.1. A graphical representation of how to use the outputs from the prior models as auxiliary features when computing the score of a new class.

which considers both from where and how much to transfer using a principled multiclass formulation. We model our learning algorithm using the structural risk minimization principle, with a group norm regularization term which allows to tune the level of sparsity in the domain of the prior models. We show that it is possible to cast the problem within the Multi Kernel Learning framework, and to solve it efficiently with off-the-shelf MKL solvers. We use the OBSCURE algorithm in Section 4.4 to solve the problem in the primal, resulting in a computationally efficient method that scales well w.r.t. the number of priors.

6.2 Problem Definition

This section introduces formally the transfer learning framework used in this chapter.

Prior Knowledge. Consider the scenario where we have already known $K (K \geq 2)$ classes, modeled via a classification function f_{p} of the form: $f_{\mathrm{p}}(\boldsymbol{x}) = \arg\max_{z \in \mathbb{Z}} \; s_{\mathrm{p}}(\boldsymbol{x}, z)$, where $\mathbb{Z} = \{1, \ldots, K\}$, and $s_{\mathrm{p}}(\boldsymbol{x}, z)$ is the score function of predicting the instance $\boldsymbol{x}$ as the z-th prior class. In the case of binary classification, the classification function can be further simplified as $f(x) = \operatorname{sign}(s(\boldsymbol{x}))$, with $\mathbb{Z} = \{-1, +1\}$. In the rest of the chapter, we only describe our model for the multiclass situation, as its modification to the binary case is straightforward.

The Transfer Learning Framework. We are interested in the task of learning a classifier for K' categories, different from the K categories already known. Given the new training set $\{\boldsymbol{x}_i, y_i\}_{i=1}^{N}$, traditional supervised learning methods, *e.g.,* SVM, minimize an upper bound of the generalization error, without taking advantage of the existing models f_{p}. However, when the number of training samples is small, this upper bound may become very loose and the learned model becomes unreliable. One way to improve performance is to exploit existing priors. Here, we propose to incorporate the predictions of prior knowledge models with the training samples as auxiliary features. In addition to the training sample $\boldsymbol{x}_i$, we also gather the scores $s_{\mathrm{p}}(\boldsymbol{x}_i, z)$, $z = 1, \ldots, K$, predicted by the prior models. In this thesis, we focus on the standard linear model. Therefore, when learning a new category the score function is:

$$s(\boldsymbol{x}, y) = \bar{\boldsymbol{w}} \cdot \bar{\phi}(\boldsymbol{x}, y) = \boldsymbol{w}^{(0)} \cdot \phi^{(0)}(\boldsymbol{x}, y) + \sum_{z=1}^{z=K} \boldsymbol{w}^{(y,z)} \cdot \phi^{(y,z)} \left(s_{\mathrm{p}} \left(\boldsymbol{x}, z \right), y \right) \tag{6.1}$$

where $\boldsymbol{w}^{(\cdot)}$ is a hyperplane, $\phi^{(\cdot)}(\cdot, \cdot)$ is the joint feature mapping function. We use the index 0 to indicate the feature mapping function $\phi^{(0)}(\boldsymbol{x}, y)$ for the original input features $\boldsymbol{x}$ and their corresponding model parameters $\boldsymbol{w}^{(0)}$. The indices (y, z) correspond to the feature mapping of $s_{\mathrm{p}}(\boldsymbol{x}, z)$ to the y-th new class, where $y = 1, \ldots, K'$. In other words, given the score $s_{\mathrm{p}}(\boldsymbol{x}, z)$ produced by the prior model, $\boldsymbol{w}^{(y,z)}$ represents the contribution of the z-th prior model in predicting that $\boldsymbol{x}$ belongs to class y. Intuitively, if prior knowledge of a bicycle gives a high score to images of a motorbike, this information may also be useful in the score function of motorbikes, since the two classes share common visual properties. Therefore, we might expect that the model will give to this prior knowledge a higher weight. On the contrary, we expect lower weights for classes which are not very relevant, such as dogs. Figure 6.1 illustrates the approach when computing the score for one class. Again, the predicted label is the class achieving the highest score.

Ideally, we would like to build the auxiliary feature representation using all the prior knowledge we have, and let the learning algorithm decide automatically from where to transfer and how much to transfer. Nevertheless, from a machine learning point of view, the more priors are considered, the higher is the risk for overfitting, especially when the number of training samples is limited. Moreover, among the K prior models, we expect only few of them to be relevant w.r.t. a specific new class, while the rest can even add noise to the problem producing negative transfer. Both factors need to be taken in consideration when designing the learning algorithm.

Learning the Objective Function. The supervised learning optimization problem here is to find the modeling parameter $\bar{\boldsymbol{w}}$ that minimizes the structural risk introduced in Section 1.3:

$$\min_{\bar{\boldsymbol{w}}} \ \lambda h(\bar{\boldsymbol{w}}) + \sum_{i=1}^{N} \ell\left(\bar{\boldsymbol{w}}, \boldsymbol{x}_i, y_i\right) \ . \tag{6.2}$$

As stated above, we would like to encourage sparsity on the level of prior models, such that out of all the models, only a few of them are actually taking part in the scoring function. For this purpose we select the squared $(2,p)$ group norm as our regularizer,

$$h(\bar{\boldsymbol{w}}) = \frac{1}{2}\|\bar{\boldsymbol{w}}\|_{2,p}^2 = \frac{1}{2}\left\|\left[\|\boldsymbol{w}^{(0)}\|_2, \ \|\boldsymbol{w}^{(1,1)}\|_2, \ \cdots, \ \|\boldsymbol{w}^{(y,z)}\|_2 \ \cdots, \ \|\boldsymbol{w}^{(K',K)}\|_2\right]\right\|_p^2,$$

with $p \in (1,2]$. Each $\boldsymbol{w}^{(y,z)}$ forms its own group, and minimizing $h(\bar{\boldsymbol{w}})$ corresponds to minimize the norm of each $\boldsymbol{w}^{(\cdot)}$ jointly.

The learning formulation is flexible, and we can use any convex Lipschitz loss function. In this chapter we will again consider only the hinge loss ℓ^{HL} and the multiclass loss ℓ^{MC}.

6.3 Multiple Kernel Transfer Learning

The original learning problem (6.2) can be converted into an l_p-norm MKL (problem (4.3), Section 4.2), which can be solved with an l_p-norm MKL algorithm such as OM-2 (Section 4.3), OBSCURE (Section 4.4) or the SHOGUN library (Sonnenburg *et al.*, 2006).

To transform problem (6.2), we first set

$$\bar{\boldsymbol{w}} \ = \ [\ \boldsymbol{w}^{(0)}, \ \boldsymbol{w}^{(1,1)}, \ \cdots, \ \boldsymbol{w}^{(y,z)}, \ \cdots, \ \boldsymbol{w}^{(K',K)} \ ] \ ,$$

and

$$\bar{\boldsymbol{\phi}}(\boldsymbol{x}, y) \ = \ [\ \phi^{(0)}, \ \phi^{(1,1)}(s_{\mathrm{p}}(\boldsymbol{x},1), y), \ \cdots, \ \phi^{(y,z)}((s_{\mathrm{p}}(\boldsymbol{x},z), y), \ \cdots, \ \phi^{(K',K)}(s_{\mathrm{p}}(\boldsymbol{x},K), y) \ ] \ .$$

Therefore, in total, we will have $(K \times K' + 1)$ feature mapping functions $\phi^{(\cdot)}(\cdot,\cdot)$, and the same number of kernels $K^j((\boldsymbol{x},y),(\boldsymbol{x}',y')) = \phi^j(\boldsymbol{x},y)\cdot\phi^j(\boldsymbol{x}',y')$. This definition includes the particular case of training

K' different hyperplanes, one for each new class. In fact, we have that $\phi^{(0)}(x, y)$ is equal to

$$\phi^{(0)}(\boldsymbol{x}, y) = [\mathbf{0}, \cdots, \mathbf{0}, \underbrace{\psi^{(0)}(\boldsymbol{x})}_{y}, \mathbf{0}, \cdots, \mathbf{0}],$$

where $\psi^{(0)}(\cdot)$ is a transformation that depends only on the data. Similarly, $\boldsymbol{w}^{(0)}$ will be composed by K' blocks, with each block corresponding to the hyperplane for each class. The feature mapping function for the z-th prior model output can now be written as:

$$\phi^{(y',z)}(\boldsymbol{x}, y) = \begin{cases} [\mathbf{0}, \cdots, \underbrace{\psi(s_{\mathrm{p}}(\boldsymbol{x}, z))}_{y}, \cdots, \mathbf{0}] \ , & \text{if } y = y' \\ \qquad\qquad \mathbf{0} \qquad\qquad\quad , & \text{otherwise} \end{cases} .$$

Again, $\boldsymbol{w}^{(y',z)}$ will be composed by K' blocks. However, with this construction, all the blocks of $\boldsymbol{w}^{(y',z)}$ are $\mathbf{0}$ except for the y'-th block. Hence, $\boldsymbol{w}^{(y',z)}$ only appears in the score functions $s(\boldsymbol{x}, y')$ predicting if $\boldsymbol{x}$ belongs to the class y'.

We solve the MKTL problem using our OBSCURE framework. Since OBSCURE has a faster convergence rate as the number of kernels grows, it somehow mitigates the problem that the number of kernels grows linearly with the number of priors. Moreover, since OBSCURE minimizes the primal objective function directly, it makes the learning algorithm more memory and computationally efficient, when we can write the explicit form of feature mapping $\psi(\boldsymbol{x})$ (*e.g.,* a linear kernel or polynomial kernel with a low degree).

In our experiments, we will only consider the identity function $\psi(\boldsymbol{x}) = \boldsymbol{x}$ (i.e. linear kernel) for the scores of prior models. Therefore, the algorithm does not need to use kernel caching for the extra $(K \times K')$ kernels coming from the prior knowledge. Similarly, the algorithm could also store $\boldsymbol{w}^{(y,z)}$ directly in its primal representation. Hence, compared to the original supervised learning problem without prior knowledge, the algorithm will use $\mathcal{O}(K \times K')$ extra memory space, and additional computational complexity at each iteration is also $\mathcal{O}(K \times K')$. In the experiments we modified our OBSCURE algorithm to incorporate the auxiliary prior features and learn them efficiently, using both a binary and a multiclass loss function. For the binary version, we also modified the algorithm to obtain a weighted version for unbalance data (Chang and Lin, 2001), which considers a different value of C for positive and negative examples.

The value of the parameter p is usually defined through cross-validation, and its optimal value depends on the sparseness of the data. According to the Theorem 4.2 , it is also possible to set p equal to $\frac{2\log F}{2\log F-1}$ to get a convergence rate that depends logarithmically on the total number of kernels, which is denoted by F. With this setup of p, we have only one free parameter λ.

6.4 Comparison with Existing Methods

In this section we briefly discuss other related existing approaches, emphasizing the connections and differences between them and our method.

Using model outputs as auxiliary features. The idea of using the output of other classifiers as basic feature representation has been well-explored in various AI domains. It recently gained popularity in the computer vision community, thanks to a large amount of annotated object image datasets that become available on the web. Several papers demonstrated that the outputs of object detectors (Li *et al.*, 2010), visual attributes (Farhadi *et al.*, 2009; Lampert *et al.*, 2009) and semantic visual concept (Torresani *et al.*, 2010; Vogel and Schiele, 2008) can be used to define a good feature representation and to improve recognition performance. Our transfer learning approach follows this line of thoughts. The novelty lies in using the outputs of object classifiers as additional feature representations combined with sample features from the new target class. This makes it possible to exploit these ideas within the transfer learning framework. Moreover, we differ from these methods, as we use prediction outputs from models of similar object categories (*e.g.,* , when transfers from bicycle to motorbike). This is in contrast with, for instance Object Bank (Li *et al.*, 2010) where the output of semantic part detectors (*e.g.,* , sky, tree) are used.

Most works (Farhadi *et al.*, 2009; Lampert *et al.*, 2009; Torresani *et al.*, 2010; Vogel and Schiele, 2008) use features computed from the entire image. Notably different, Object Bank (Li *et al.*, 2010) uses a localized representation where features are extracted at different spatial pyramid levels. This is more suited for representing cluttered images composed of many objects, such as natural scenes. Although in our experiments we also use outputs computed from the entire images, the algorithm we propose can handle various multi-dimensional representations, *e.g.,* representations like Object Bank. Furthermore, MKTL takes advantage of the MKL machinery, which allows to group freely information from different unconstrained sources including the new training data into different kernels.

Finally, MKTL has a principled multiclass formulation. Each class learns from which auxiliary features to transfer in a joint optimization problem. This multiclass formulation could be generalized to other similar problems, such as those described above. It also allows to define different kernels for the new and the prior knowledge.

Multi Model Knowledge Transfer (Multi-KT) (Tommasi *et al.*, 2010). A transfer learning algorithm close to ours is Multi-KT, which modified the l_2 square norm regularizer in the classical Least-Square-SVM objective function, constraining the new hyperplane $\boldsymbol{w}$ to be close to some of the hyperplanes $\boldsymbol{u}^j$ of the F prior models. Its regularization term can be written as $\|\boldsymbol{w} - \sum_{j=1}^{F} \beta^j \boldsymbol{u}^j\|^2$, where β^j is a parameter to be learned which defines the reliability of known models for the new learning problem, subject to the constraint that $\|\boldsymbol{\beta}\|_2 \leq 1$. The algorithm is binary, and its final decision function for a given sample $\boldsymbol{x}$ can be written as:

$$s(\boldsymbol{x}) = \boldsymbol{w} \cdot \phi(\boldsymbol{x}) + \sum_{j=1}^{F} \beta^j \boldsymbol{u}^j \cdot \phi(\boldsymbol{x}).$$

This is very similar to the binary version of the score function defined in (6.1). However, Multi-KT is solved based on two separate optimization problems, while our algorithm finds both the best hyperplane's parameter and the weights to be assigned to each prior knowledge model in a joint optimization process. Moreover, Multi-KT requires that each prior model $\boldsymbol{u}$ is constructed using the same type of classifier of the new model. All the models (priors and new) must also use the same type of feature descriptors. On the other hand, MKTL has neither of these constrains. It is capable of *heterogeneous transfer from unconstrained priors*: we can freely combine different learning methods and different features to boost performance. Finally, Multi-KT can not be extended to principle multiclass formulation using the multiple class loss function ℓ^{MC},

6.5 Experiments

We present here three sets of experiments designed for studying the behavior of MKTL: (a) in the object category detection scenario, with priors and new model learned using the same features and learning methods (Section 6.5.1); (b) in the multiclass object categorization scenario, with limited priors and few annotated samples for the target class, where priors and new model are learned using different algorithms and features (Section 6.5.2), and (c) where the problem is again multiclass, but scaling w.r.t. the number of available priors and w.r.t. the number of labelled samples for the new classes. In all our experiments, the regularization parameter is set as $\lambda = \frac{1}{CN}$, with C selected from the set $\{0.1, 1, 10, 100, 1000\}$, and the parameter p is chosen from the set $\{1.01, 1.05, 1.10, 1.15, 1.20, 1.25, 1.30, 1.40, 1.50\}$.

We compare MKTL against the following baselines:

- [**No-Transfer**] It corresponds to the standard supervised learning task without considering prior knowledge. We train SVM classifiers using the 1-vs-All scheme for the multiclass extension. Ideally, the performance of a transfer learning algorithm should not be worse than this baseline, to avoid negative transfer that might hurt performance.

- [**Prior-Features**] We also test the performance when using only the outputs of prior models as feature descriptors. We concatenated the outputs of the prior models into a vector representation, then use a linear SVM classifier to test their performance. This idea is similar to the classemes feature proposed in (Torresani *et al.*, 2010). This baseline will help us understand the role of the prior models in the performance. For example, if the performance of all the prior models is very low compared to No-Transfer, we may expect to see an improvement in performance relatively small compared to standard supervised learning, and vice versa. This kind of baseline has often been ignored in previous transfer learning literature. Here we argue that it should be considered as an obligatory competitor, since sometimes using the prior models alone could lead to higher accuracy.

- [**Multi-KT**] We also compared against the Multi-KT transfer learning algorithm. This method assumes that all the prior models and the new model use the same type of feature descriptors and learning method. Thus, for Multi-KT, we did train all our prior models with the same feature descriptors and kernel parameters using SVM classifier. Since this algorithm has been presented only in a binary version, we implemented the multiclass extension using the 1-vs-All scheme.

- [**Average-TL**] MKTL learns the weights to combine the outputs of each prior models with the new knowledge representation. Thus, a natural baseline is to consider the information coming from the priors and the new knowledge as equally relevant. This is equivalent to train a SVM classifier using the average of all the available kernels. This method often performs as good as many MKL algorithms (Gehler and Nowozin, 2009b).

For all the baseline methods, we use the LIBSVM (Chang and Lin, 2001) package for training and testing the SVM classifier. The regularization parameter C is selected from the same range as MKTL, and the best results are reported. For No-Transfer and Average-TL, we use the RBF kernel.

6.5.1 Binary Transfer Learning

We consider the same binary experimental setup proposed in (Tommasi *et al.*, 2010)[Section 5.3] on a subset of 30 classes plus the background class extracted from the Caltech-256 database (Griffin *et al.*, 2007). Here we just repeat briefly the experimental procedure, for a detailed description of the setup we refer the readers to the original paper. The task is to recognize if a test image belongs to the target object class or not (*i.e.,* belonging to a pre-defined background class). In turns, a small number of labelled training examples are available for a target object class and all the 29 remaining classes are used for training the prior models. We use the same four image descriptors as (Tommasi *et al.*, 2010) and combine the features through concatenation. In the experiments, the number of negative examples are far larger than the number of positive examples in the training data, leading to an unbalanced data problem. This is very common in the object category detection scenario, and a popular solution to it is to give different importance weights to the positive and negative examples (Tommasi *et al.*, 2010). We modified our algorithm for this purpose. Here the weights are defined to be $w^+ = N^-/N^+$ and $w^- = 1$, where N^+ and N^- are the number of positive and negative samples. Both the normal ($w^+ = w^- = 1$) and weighted MKTL are considered in our experiments.

The average results of all the 30 categories as well as the average results for each class are shown in Figure 6.2. It can be observed that all the transfer learning methods outperform the No-Transfer approach for different numbers of training samples. Weighted MKTL achieves better performance compared to Multi-KT except for the cases with only 3 positive sample. MKTL without weights is slightly worse at the beginning, but it beats Multi-KT when the number of positive training sample reaches 15. We expect prior models to achieve high accuracy on the target task as both the prior and the target problem consist in distinguishing different objects from a common background class. It is surprising to find that using Prior-Features alone outperforms Multi-KT when the number of positive samples grows, which seems to suggest that Multi-KT is not able to combine the prior models and the new knowledge as desired (in oder to minimize the error) when the prior models are very strong. On the other hand MKTL guarantees a performance at least as good as what has been transferred. It is also interesting to look into the results

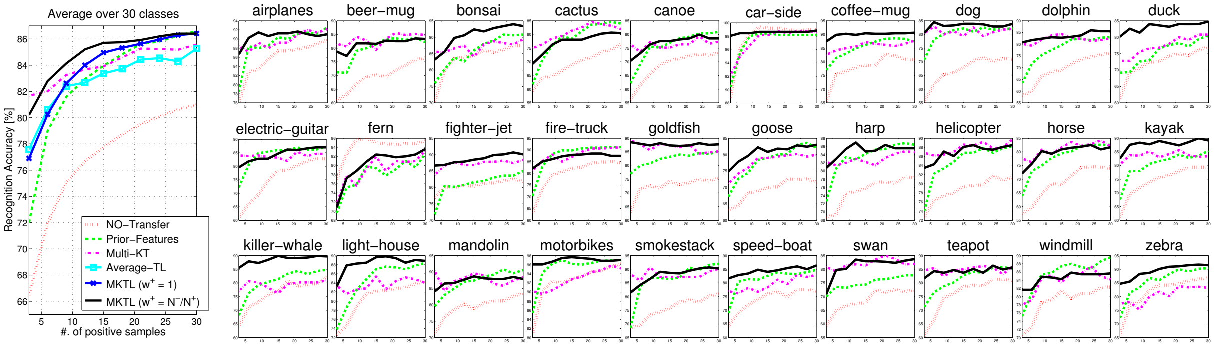


Figure 6.2. Results obtained on the object category detection scenario, when learning 1 out of 30 categories with the rest categories as prior models. Classification performance is shown as a function of the number of object training images. For each class, we repeat the experiment 5 times using different random permutations. Class by class results are shown on the right. For the sake of clarity, we only plot the results of “No-Transfer”, “Prior-Feature”, “Multi-KT” and “MKTL ($w^+ = N^-/N^+$)” on these figures.

obtained from each single class. Killer-whale and duck seem to exploit at the best the priors, while fern is the only case where all the transfer learning methods fail to avoid negative transfer. In most of the classes we observe that MKTL is better (or at least equal) than using Prior-Features alone.

6.5.2 Multiclass Transfer Learning

We perform multiclass classification experiments on two different datasets: subsets of the Caltech-256 and the Animals with Attributes (AwA) dataset. Precomputed features are available for both the databases.

For the experiments on the Caltech-256 dataset, we consider 9 new classes (bonsai, sunflower, mushroom, horse, skunk, gorilla, motorbikes, snowmobile, segway), and we randomly extract a maximum of 30 samples per class for training and 50 samples for testing. Twenty-three classes are considered as possible prior knowledge sources, which can be divided into four groups, plants (palm-tree, cactus, fern, hibiscus), animals (bat, bear, leopards, zebra, dolphin, killer-whale), vehicles (mountain-bike, fire-truck, car-side, bulldozer) and mix (grapes, tomato, camel, dog, raccoon, chimp, school-bus, touring-bike, covered-wagon), and we use different feature descriptors [1] for each group. For the first two groups, we concatenate the feature descriptors together, and train the prior models with Multiclass AdaBoost (Schapire and Singer, 1999). Then, for vehicles and mix group, we compute RBF kernels for each feature descriptor, and train SVM using the average of the RBF kernels with 1-vs-All extension. In the end we

---

[1] Plants & Mix: SIFT (Lowe, 2004) and LBP (Ojala *et al.*, 2002); Vehicles: SIFT; Animals: REGCOV (Tuzel *et al.*, 2007), SIFT and V1S+ (Pinto *et al.*, 2008). Since Multi-KT is limited to use only one type of feature descriptor, we use PHOG (Bosch *et al.*, 2007) features for all the groups.

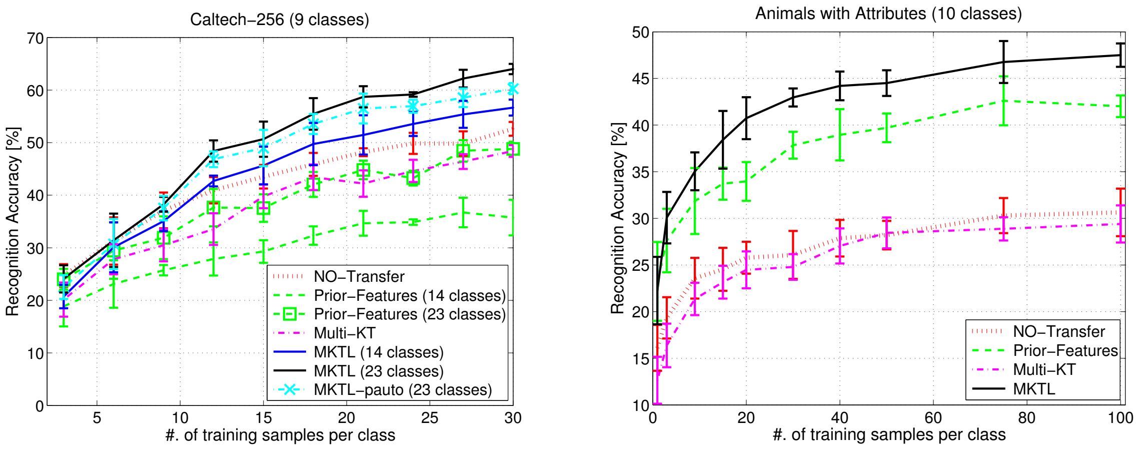


Figure 6.3. Results obtained on the multiclass object categorization scenario. Classification performance is shown as a function of the number of object training images. (left) average results obtained using subset of the the Caltech-256 dataset; (right) average results obtained using the AwA dataset. For both datasets, each experimental setup is repeated for 10 times, and their standard deviations are also reported.

use a RBF kernel for the new training images described with PHOG (Bosch *et al.*, 2007) features. The γ parameters of the RBF kernels were fixed to the mean of the pairwise distances among the samples as done in (Gehler and Nowozin, 2009b; Lampert *et al.*, 2009). Our choice of features descriptors and prior models are arbitrary, as we want to show that the prior models could be constructed using various features descriptors and learning algorithms. For comparison, we first consider transfer learning from the first 14 classes (from palm-tree to bulldozer). Then we progressively add the remaining 9 classes (from grapes to covered-wagon) to the prior models. Meanwhile, we also experiment with $p = \frac{2 \log K}{2 \log K - 1}$ to test if it is possible to set the parameter p automatically (MKTL-p_{auto}). These results are reported in Figure 6.3(left).

We performed similar experiments on the AwA dataset. We consider the same 10 test classes in (Lampert *et al.*, 2009) as new classes to learn, randomly extracting a maximum of 100 samples from each class for training and 50 samples for test. The remaining 40 classes are used to build prior knowledge sources. We use the average of two RBF kernels computed using color histogram and SURF features (Bay *et al.*, 2008) for describing all the prior classes, and train these models using SVM with 1-vs-All extension. Again, we use PHOG (Bosch *et al.*, 2007) feature with a RBF kernel for describing the new training images, and the γ parameters are computed using the same method discussed above. These results are reported in Figure 6.3(right).

MKTL clearly shows a gain in performance. It can be observed that MKTL achieves better results compared to No-Transfer, and other baseline methods, especially when the number of training samples grows (Figure 6.3(left & right), after receiving 5-10 training examples per class), and more prior models are used (Figure 6.3 (left), 23 priors compared to 14 priors). Here the expected *higher start* effect (Rosenstein *et al.*, 2005) with few training samples is not as significant as in the binary case. It suggests that the multiclass problem is substantially more difficult compared to the binary object categorization task. Thus, we could expect that we need more samples for each class in order to learn the tasks. Moreover, although the performance of Prior-Features alone is relatively low, MKTL still achieves significant improvement in performance by combining the prior outputs with the new knowledge. We also see that the improvement is consistent even after receiving 100 training samples per class (Figure 6.3 (right)). This demonstrates the *higher asymptote* advantage for knowledge transfer (Rosenstein *et al.*, 2005). This advantage is theoretically guaranteed by the fact that the knowledge transfer problem is solved in a higher dimensional feature space than the original No-Transfer. The same performance can not be expected for Multi-KT: when the number of training samples grows, the regularization term $\|\boldsymbol{w}-\sum_{j=1}^{F}\beta^j\boldsymbol{u}^j\|^2$ looses its relevance and the problem reduces to learning from scratch.

The results for MKTL using the automatic setup of the p parameter is comparable to the results we obtained with cross validation on p. This suggests a possible way to eliminate one free parameter in practice when validation data are not available. We also tested Multi-KT on both datasets using the 1-vs-All extension. In this case, Multi-KT does not improve over the No-Transfer baseline. One possible explanation may be that the 1-vs-All scheme may induce confusion when combining the binary results over multiple classes, as the special optimization scheme used in Multi-KT does not guarantee that the output for each binary classification problem will be in a similar range. It is also worth mentioning that our learning algorithm is very efficient and takes less than 1 minute to finish, on the AwA dataset with 100 training sample per categories and 40 prior models.

6.6 Conclusions

In this chapter, we present a multiclass transfer learning algorithm for learning object categories from few examples. The algorithm uses the output of pre-trained models as auxiliary feature inputs, and uses a learning based approach to automatically decide from which prior models to transfer and how much to transfer. The proposed approach has no constraint on the pre-trained prior models and their features representation, as they can be built from different types of learning methods and using different types of feature representations. Furthermore, our algorithm uses a principled multiclass formulation and solves the multiclass problem in a joint optimization process. The optimization algorithm is modified from the l_p-norm MKL framework which solves the optimization problem in the primal. It thus scales well w.r.t. the number of prior models. Experiments show that our algorithm outperforms all the baseline methods, and is able to boost the performance when more relevant priors are given. Thanks to the principled multiclass formulation, the performance gain is more significant for multiclass scenarios, where the tasks are substantially more difficult than the more studied binary case.

This chapter concludes the second part of this thesis. We presented two examples on exploiting cue(s) which occurs together with the target to facilitate the learning. As discussed in Chapter 1, unlike the cue integration problem where a general framework exists, the cues exploitation problem is usually task specifical. It usually requires to have domain knowledge and understand the structure of the data. On the other hand, this type of accompanying information usually exists in many problems, therefore it could be an interesting and challenging problem to investigate how these methods generalize over different practical applications in the future.

Chapter 7

Summary and Future Direction

In this chapter we are going to summarize what we have presented in this thesis, and sketches possible future direction of research.

7.1 Discussion

In this thesis we studied the problem of learning from multiple cue inputs, under two different contexts: Cues Integration and Cues Exploitation. In the first part, we showed that integrating multiple cues can improve the classification accuracy significantly, especially when cues are independent and come from multiple sensory inputs. Our algorithms are efficient, and are capable of online updating of their internal representations when new examples arrive. We elaborated some theories for these algorithms, which provide performance guarantees for the online algorithms, and convergence bounds for the batch algorithms. In the second part, we showed that it is possible to exploit information from some cues, and improve the performance of learning other cues under a less supervised setting. To illustrate the ideas, we presented two different examples, where we learn image classifiers from images using textual cue co-occurring with the images, and transfer trained models of correlated tasks to the new tasks to boost the performance.

7.2 Future Directions

We would like to explore the following possibilities in the future.

7.2.1 Multiple Kernel Active & Semi-supervised Learning

MKL has shown its promise by achieving state-of-art performance in multiclass image classification task under standard supervised learning setup. For learning problems, apart from training efficiency and scalability, another important issue is the lack of labeled training data. Active learning (see Dasgupta and Langford (2009) for a review) is defined by contrast to the passive model of supervised learning where all the labels for examples are obtained without reference to the learning algorithm. In active learning, unlabeled examples are provided to the learner, and the learner can choose interactively which data points to label. The idea is that interaction can substantially reduce the number of required labels. In semi-supervised learning (see Zhu (2005) for a complete review) for classification, the system try to incorporate unlabeled examples to improve the performance of models trained with labeled data alone. Previous studies in both directions were mainly conducted for single cue. It is thus quite natural to ask if we could take advantage of using multiple cue inputs in the active learning and semi-supervised learning framework to improve the performance as well as reduce further the number of required labels.

7.2.2 Approximate Inference for Training MMS

As mentioned in Chapter 5, the most expensive step in training MMS is solving the $\arg\max$ in step 4 of Algorithm 9 for the stochastic subgradient descent method. For complex problems with a large number of instances and exponential size candidate labeling vectors, the inference is usually intractable. It is possible to extend our framework using approximate inference to solve other related tasks. For example, a popular task (Barnard *et al.*, 2003; Wang and Mori, 2010), which is to learn a scene classifier for image annotation from segmented images with unaligned object-level textual annotations (tags), can also be casted into our framework. In the image annotation task, the size of candidate labeling set of an image depends exponentially on the size of image segmentations and their tags. Therefore, it can be very memory and computational expensive to enumerate all the possible assignments as well as to infer $\hat{\boldsymbol{Z}}$. In this case, we could use some approximate approaches to speed up the learning. One interesting recent study on image annotation by Wang and Mori (2010) models the problem using the latent SVM framework, and formulates its inference step as a Linear Programming (LP) problem with constraints on tags assignment. It is possible to use a similar LP formulation in our algorithm to perform the inference step approximately without enumerating all the possible assignments in a large set.

Appendix A

Datasets and Features

Here we briefly introduce the datasets used in this thesis, and we also describe the used feature descriptors as well as how their kernel matrices are computed when using kernel approaches.

The KTH-IDOL2 dataset (Pronobis *et al.*, 2010) contains 24 image sequences acquired using a perspective camera mounted on two mobile robot platforms. The sequences were captured with two robots (Dumbo and Minni) moving in an indoor laboratory environment consisting of five different rooms; they were acquired under various weather and illumination conditions (sunny, cloudy, and night) and across a time span of six months. This dataset is ideal for testing online learning algorithms: the algorithm has to incrementally update the model, so to adapt to the variations captured in the dataset. During the acquisition, three types of inputs were recorded: images, laser scans data and odometry data. From the images, we extract three types of image descriptors, namely, Composed Receptive Field Histogram (CRFH) (Linde and Lindeberg, 2005) (Gaussian derivative along x & y direction), Bag-of-Words (BOW) (Sivic and Zisserman, 2003) using SIFT descriptor (Lowe, 2004) with 300 visual words, and RGB color histogram. We also use a simple geometric feature from the Laser Scan sensor (Mozos *et al.*, 2005). In total, we have four different types of features, which correspond to four kernels (the χ^2 kernel is used for the image features, and the RBF kernel for the laser feature).

The ETH-80 dataset (Leibe and Schiele, 2003) consists of 80 objects from eight different categories (apple, tomato, pear, toy-cows, toy-horses, toy-dogs, toy-cars and cups). Each category contains

10 objects with 41 views per object, spaced equally over the viewing hemisphere, for a total of 3,280 images. We use four image descriptors: one color feature – RGB color histogram, two texture descriptors – CRFH (Linde and Lindeberg, 2005) with two different kinds of filters (Gaussian derivative L_xL_y and gradient direction ($DirC$), and a global shape feature – centered masks (Leibe and Schiele, 2003).

The Oxford Flower dataset (Nilsback and Zisserman, 2006) contains 17 different categories of flowers (available at `www.robots.ox.ac.uk/~vgg/research/flowers/`). Each class has 80 images with three predefined splits (train, validation and test). The authors also provide 7 precomputed distance matrices. These distance matrices are transformed into kernel using $\exp(-\gamma^{-1}d)$, where γ is the average pairwise distance and d is the distance between two examples. It results in 7 different kernels.

The Pendigits dataset (Gönen and Alpaydin, 2010) is on pen-based digit recognition (multiclass classification with 10 classes) and contains four different feature representations (available at `http://mkl.ucsd.edu/dataset/pendigits`). The dataset is splitted into independent training and test sets with 7,494 samples for training and 3,498 samples for testing. We have generated 4 kernel matrices, one matrix for each feature, using an RBF kernel, $\exp(-\gamma^{-1}\|\boldsymbol{x}_i - \boldsymbol{x}_j\|^2)$. For each feature, γ is equal to the average of the squared pairwise distances between the examples.

The Caltech-101 dataset (Fei-Fei *et al.*, 2004) is a standard benchmark dataset for object categorization. It contains 101 different object classes plus one background class. For each class, it has at least 31 images. In our experiments, we used the pre-computed features and kernels of Gehler and Nowozin (2009b) which the authors made available on their website: `http://www.vision.ee.ethz.ch/~pgehler/projects/iccv09/`, with the same training and test splits. This allows us to compare against them directly. Following that, we report results using all 102 classes of the Caltech-101 dataset using five splits. There are five different image descriptors, namely, PHOG Shape Descriptors (PHOG) (Bosch *et al.*, 2007), Appearance Descriptors (App) (Lowe, 2004), Region Covariance (RECOV) (Tuzel *et al.*, 2007), Local Binary Patterns (LBP) (Ojala *et al.*, 2002) and V1S+ (Pinto *et al.*, 2008). All of them but the V1S+ feature were computed in a spatial pyramid as proposed in Lazebnik *et al.* (2006), using several different setups of parameters. This generates several kernels (PHOG, 8 kernels; App, 16 kernels; RECOV, 3 kernels; LBP 3 kernels; V1S+, 1 kernels). We also consider a subwindow kernel, as proposed in Gehler and Nowozin (2009a). In addition to the 32 kernels, the products of the pyramid levels for each feature results in other 7 kernels, for a total of 39 different kernels.

The Caltech-256 dataset (Griffin *et al.*, 2007) is an extension of Caltech-101, which is collected in a similar manner with several improvements. It contains 256 different object classes plus one background class. The minimum number of images in any category is increased to 80. Precompute features have been made available by Gehler and Nowozin (2009b) on their website `http://www.vision.ee.ethz.ch/~pgehler/projects/iccv09/`. They are PHOG, App, RECOV, LBP and V1S+, which have also been used for representing Caltech-101 in the same work.

The Animals with Attributes (AwA) Dataset (Lampert *et al.*, 2009) consists of 30,475 images of 50 animals classes with six pre-extracted feature representations (Color Histogram, PHOG, Local Self-Similarity Histograms (Shechtman and Irani, 2007), SIFT, rgSIFT (van de Sande *et al.*, 2008) and SUFR (Bay *et al.*, 2008)) for each image. The dataset was originally provided as a platform to benchmark transfer-learning algorithms, in particular attribute based classification (Lampert *et al.*, 2009). However, it can also be used to study image categorization algorithms.

The MNIST dataset (LeCun *et al.*, 1998) is a handwritten digits dataset. It has a training set of 60,000 gray-scale 28x28 pixel digit images for training and 10,000 images for testing. We cut the original digit image into four square blocks (14×14) and obtained an input vector from each block. We used three kernels on each block: a linear kernel, a polynomial kernel and a RBF kernel, resulting in 12 kernels.

The Labelled Yahoo! New Database (Berg *et al.*, 2004a; Guillaumin *et al.*, 2010) was collected by Berg *et al.* (2004a) from the Yahoo! News website (`http://news.yahoo.com/`) in a span of two years. It consists of news images and their captions describing the events appearing in the images. Guillaumin et al. (Guillaumin *et al.*, 2010) provide ground-truth annotations of the dataset and precomputed feature descriptors of the faced detected by (Viola and Jones, 2004) (available at `http://lear.inrialpes.fr/data/`). The resulted dataset contains 20,071 images and 31,147 detected faces. Among them, 3,105 image-caption pairs contain at least one spatial indicator, such as “(L)”, “(R)” or “(C)”, which suggests the relative position of the face in the image. There are more than 10,000 different names in the captions. The maximum number of detected faces in an image is 15, and the maximum number of names in a caption is 9. The descriptors are 128-D SIFT (Lowe, 2004) at 3 scales extracted at 13 landmark points localized by (Everingham *et al.*, 2006), resulting in a 4,992-D descriptor. In our experiments, we only use the first 1664-D of the descriptor at scale 1.

The Idiap/ETHZ Faces and Poses dataset (Jie *et al.*, 2009b) was created by us for the purpose of studying automatic face and pose annotation (available at `http://www.vision.ee.ethz.ch/~calvin/faces+poses/`). It contains 1,703 image-caption pairs collected by querying Google Images using keywords generated by combining different names (sport stars and politicians) and verbs (from sports and social interactions). An example query is "Barack Obama" + "shake hands". A caption contain the names of some of the persons in the corresponding image, and verbs indicating what they are doing. The captions are derived from the snippet of text returned by Google Images and typically mention the action of at least one person in the image, but also contain names/verbs not appearing in the image. In addition to the image-caption pairs, ground-truth associations between names and verbs in the captions, ground-truth lists of which names from the caption appear in the images, ground-truth locations of the persons in the images, name-verb pairs extracted automatically from the captions using (OpenNLP, 2010; Deschacht and Moens, 2009), as well as face and upper-body bounding-boxes detected by the methods of Ferrari *et al.* (2008b); Rodriguez (2006) were also released with the dataset, which can be used directly in the experiments.

Appendix B

Miscellaneous Proofs

Proof of Theorem 3.1

Proof: First we have that the cumulative loss of the learner at the second layer, for any sequence of scores, $[s^1(\boldsymbol{x}_1, 1), \cdots, s^j(\boldsymbol{x}_1, y), \cdots, s^F(\boldsymbol{x}_1, K)]$, $\cdots$, $[s^1(\boldsymbol{x}_T, 1), \cdots, s^j(\boldsymbol{x}_T, y), \cdots, s^F(\boldsymbol{x}_T, K)]$, generated by the classifiers f^j from the first layer, are bounded by

$$\sum_{t=1}^{T} g'(\boldsymbol{w}_t) \leq \sum_{t=1}^{T} g'(\boldsymbol{u}) + r'(\boldsymbol{u}, T) \ , \tag{B.1}$$

where $g'(\boldsymbol{u}) = \ell(\boldsymbol{u}, [s^1(\boldsymbol{x}_t, 1), \cdots, s^j(\boldsymbol{x}_t, y), \cdots, s^F(\boldsymbol{x}_t, K)], y_t)$. Suppose that the j-th classifier have the minimum loss in the first layer. Since the above bound holds for any $\boldsymbol{u'} \in \mathbb{R}^{F*K}$, we could simply set $\boldsymbol{u'}$ to be the a full zero vector, except for the outputs from the j-th classifier, $s^j(\boldsymbol{x}_t, y), \forall y = 1, \cdots, K$, which we set their value to one. In this case, since the first layer and the second layer use the same type of loss function, we will have

$$\sum_{t=1}^{T} g'(\boldsymbol{u}) = \sum_{t=1}^{T} g^j(\boldsymbol{v}_t^j) \tag{B.2}$$

Plug (B.2) into (B.1) we proof the theorem. □

Proof of Lemma 4.1

Proof: The proof is based on an adaptation of a result from Kakade *et al.* (2009) for the FTRL algorithm. We proceed by bounding the quantity $\bar{\boldsymbol{\theta}}_{T+1} \cdot \bar{\boldsymbol{u}}$ from below and from above. From (Kakade *et al.*, 2009, Corollary 19), we know that $h^*(\bar{\boldsymbol{\theta}}) = \frac{1}{2q}\|\bar{\boldsymbol{\theta}}\|_{2,q}^2$ is 1-smooth w.r.t. $\|\cdot\|_{2,q}$. Moreover, line 9 in the algorithm's pseudo-code implies that $\bar{\boldsymbol{w}}_t = \nabla h^*(\bar{\boldsymbol{\theta}}_t) = \nabla h^*\big(-\sum_{i=1}^{t-1}\eta_i \bar{\boldsymbol{z}}_i\big)$. Hence, we obtain

$$\|\bar{\boldsymbol{\theta}}_{T+1}\|_{2,q}^2 \le \|\bar{\boldsymbol{\theta}}_T\|_{2,q}^2 - 2q\eta_T \bar{\boldsymbol{w}}_T \cdot \bar{\boldsymbol{z}}_T + q\eta_T^2\|\bar{\boldsymbol{z}}_T\|_{2,q}^2 \le q\sum_{t=1}^{T}\big(\eta_t^2\|\bar{\boldsymbol{z}}_t\|_{2,q}^2 - 2\eta_t\bar{\boldsymbol{w}}_t \cdot \bar{\boldsymbol{z}}_t\big)\ . \tag{B.3}$$

Using the convex inequality for norms we then get

$$\bar{\boldsymbol{\theta}}_{T+1} \cdot \bar{\boldsymbol{u}} \le \|\bar{\boldsymbol{\theta}}_{T+1}\|_{2,q}\ \|\bar{\boldsymbol{u}}\|_{2,p} \le \|\bar{\boldsymbol{u}}\|_{2,p}\sqrt{q\sum_{t=1}^{T}\big(\eta_t^2\|\bar{\boldsymbol{z}}_t\|_{2,q}^2 - 2\eta_t\bar{\boldsymbol{w}}_t \cdot \bar{\boldsymbol{z}}_t\big)}\ . \tag{B.4}$$

We can further bound the last term by considering that $-\bar{\boldsymbol{w}}_t \cdot \bar{\boldsymbol{z}}_t$ is less than 1. Using that $\|\bar{\boldsymbol{z}}_t\|_{2,q}^2 \le L$ we can further upper bound $\bar{\boldsymbol{\theta}}_{T+1} \cdot \bar{\boldsymbol{u}}$ as follows

$$\bar{\boldsymbol{\theta}}_{T+1} \cdot \bar{\boldsymbol{u}} \le \|\bar{\boldsymbol{u}}\|_{2,p}\sqrt{q\sum_{t=1}^{T}\big(\eta_t^2 L^2 + 2\eta_t\big)}\ . \tag{B.5}$$

For the lower bound we have that

$$\bar{\boldsymbol{\theta}}_{T+1} \cdot \bar{\boldsymbol{u}} = \sum_{t=1}^{T} -\eta_t \bar{\boldsymbol{u}} \cdot \bar{\boldsymbol{z}}_t \ge \sum_{t=1}^{T}\eta_t\big(1 - \ell(\bar{\boldsymbol{u}}, \boldsymbol{x}_t, y_t)\big) \ge \sum_{t=1}^{T}\eta_t - \sum_{t=1}^{T}\eta_t\ell(\bar{\boldsymbol{u}}, \boldsymbol{x}_t, y_t) \tag{B.6}$$

Combining this last inequality with (B.5), we obtain the stated inequality. □

Proof of Theorem 4.1

Proof: We know that at steps when a mistake occurs, $\bar{\boldsymbol{w}}_t \cdot \bar{\boldsymbol{z}}_t \ge 0$; and the steps when a margin error occurs, $-1 \le \bar{\boldsymbol{w}}_t \cdot \bar{\boldsymbol{z}}_t \le 0$. Hence we have that $\eta_t = 1$ at mistakes, and $\eta_t \le 1$ at margin errors. So separating the two type of steps in the inequality from Lemma 4.1, we could obtain that

$$M + \sum_{t\in\mathcal{I}}\eta_t \le \sum_{t\in\mathcal{M}\cup\mathcal{I}}\ell(\bar{\boldsymbol{u}}, \boldsymbol{x}_t, y_t) + \|\bar{\boldsymbol{u}}\|_{2,p}\sqrt{qML^2 + q\sum_{t\in\mathcal{I}}\big(\eta_t^2\|\bar{\boldsymbol{z}}_t\|_{2,q}^2 - 2\eta_t\bar{\boldsymbol{w}}_t \cdot \bar{\boldsymbol{z}}_t\big)}\ . \tag{B.7}$$

Solving the inequality for M, and using the definition of C and S, we obtain

$$M \leq \frac{C}{2} + S + \frac{C}{2}\sqrt{1 + \frac{4}{C}\left(A + S\right)} - \sum_{t \in \mathcal{I}} \eta_t$$

where

$$A = \sum_{t \in \mathcal{I}} \frac{\eta_t^2 \|\bar{\boldsymbol{z}}_t\|_{2,q}^2 - 2\eta_t \bar{\boldsymbol{w}}_t \cdot \bar{\boldsymbol{z}}_t - \eta_t L^2}{L^2} \ .$$

Now, observing that η_t has been chosen so that $A \leq 0$ and overapproximating we obtain the stated bound. □

Proof of Lemma 4.2

Proof: The first relation can be found in the proof of Theorem 20 in Kakade *et al.* (2009). The second one can be obtained differentiating h. The third relation is obtained using Lemma 2 in Shalev-Shwartz and Singer (2007). The last one is obtained from the second one. □

Proof of Lemma 4.3

Proof: Define $g'_t(\boldsymbol{w}) = g_t(\boldsymbol{w}) + \frac{\xi_t}{2}\|\boldsymbol{w} - \boldsymbol{w}_t\|^2$. Using the assumptions of this Lemma, we have that g'_t is $(\sigma + \frac{\xi_t}{\alpha})$-strongly convex w.r.t. to h. Moreover we have that $\partial g'_t(\boldsymbol{w}_t) = \partial g_t(\boldsymbol{w}_t)$, because the gradient of the proximal regularization term is zero when evaluated at $\boldsymbol{w}_t$ (Do *et al.*, 2009). Hence we can apply Theorem 1 from Shalev-Shwartz and Singer (2007) to have

$$\sum_{t=1}^{T} g_t(\boldsymbol{w}_t) - \sum_{t=1}^{T}\left(g_t(\boldsymbol{u}) + \frac{\xi_t}{2}\|\boldsymbol{u} - \boldsymbol{w}_t\|^2\right) = \sum_{t=1}^{T} g'_t(\boldsymbol{w}_t) - \sum_{t=1}^{T} g'_t(\boldsymbol{u}) \leq \frac{1}{2}\sum_{t=1}^{T} \frac{L_t^2}{\sigma t + \frac{\sum_{i=1}^{t}\xi_i}{\alpha}} \ . \tag{B.8}$$

Using the hypothesis of this Lemma we obtain

$$\sum_{t=1}^{T} g_t(\boldsymbol{w}_t) - \sum_{t=1}^{T} g_t(\boldsymbol{u}) \leq \frac{1}{2}\sum_{t=1}^{T}\left(\xi_t\|\boldsymbol{u} - \boldsymbol{w}_t\|^2 + \frac{\alpha L_t^2}{\alpha\sigma t + \sum_{i=1}^{t}\xi_i}\right) \tag{B.9}$$

$$\leq \frac{1}{2}\sum_{t=1}^{T}\left(4\xi_t R^2 + \frac{\alpha L_t^2}{\alpha\sigma t + \sum_{i=1}^{t}\xi_i}\right) \ . \tag{B.10}$$

Using the definition of ξ_t in the algorithm and Lemma 3.1 in Bartlett *et al.* (2008), we obtain the stated bound. $\square$

Proof of Theorem 4.2

Proof: Define $h(\bar{\boldsymbol{w}}) : S \to \mathbb{R} = \frac{q}{2}\|\bar{\boldsymbol{w}}\|_{2,p}^2$, where $S = \{\bar{\boldsymbol{w}} : \|\bar{\boldsymbol{w}}\|_{2,p} \le R\}$. Define also $g_t(\bar{\boldsymbol{w}}) = \frac{\lambda}{2}\|\bar{\boldsymbol{w}}\|_{2,p}^2 + \ell(\bar{\boldsymbol{w}}, \boldsymbol{x}_t, y_t) = \frac{\lambda}{q}h(\bar{\boldsymbol{w}}) + \ell(\bar{\boldsymbol{w}}, \boldsymbol{x}_t, y_t)$. Using Lemma 1 in Shalev-Shwartz and Singer (2007), we can see that these two functions satisfy the hypothesis of Lemma 1, with $\alpha = q$, $\sigma = \frac{\lambda}{q}$. It is easy to verify that $\bar{\boldsymbol{w}}_{t+1}$ is equal to $\nabla h^*(\nabla h(\bar{\boldsymbol{w}}_t) - \eta_t \boldsymbol{z}_t)$. In fact, taking into account Properties 2-4 in Lemma 4.2 with with $B = R$, lines 9-11 in Algorithm 5 are equivalent to

$$\bar{\boldsymbol{w}}_{t+1} = \nabla h^*(\theta_t - \eta_t z_t) \ , \tag{B.11}$$

$$\bar{\boldsymbol{\theta}}_{t+1} = \nabla h(\bar{\boldsymbol{w}}_{t+1}) \ . \tag{B.12}$$

We also have that

$$\|\partial g_t(\bar{\boldsymbol{w}})\|_{2,q} \le \frac{\lambda}{q}\|\nabla h(\bar{\boldsymbol{w}}_t)\|_{2,q} + \|\bar{\boldsymbol{z}}_t\|_{2,q} = \lambda\|\bar{\boldsymbol{w}}_t\|_{2,p} + \|\bar{\boldsymbol{z}}_t\|_{2,q} \le c, \tag{B.13}$$

where the equality is due to Properties 2 and 4 in Lemma 4.2. So we have

$$\sum_{t=1}^{T}(g_t(\bar{\boldsymbol{w}}_t) - g_t(\bar{\boldsymbol{w}}^*)) \le \min_{\xi_1,\cdots,\xi_T} \sum_{t=1}^{T}\left[4\xi_t R^2 + \frac{qc^2}{\lambda t + \sum_{i=1}^{t}\xi_i}\right] \ . \tag{B.14}$$

Reasoning as in Shalev-Shwartz *et al.* (2007), we divide by T, take the expection on both side and use the Markov's inequality. So we obtain that with probability at least $1 - \delta$

$$f(\bar{\boldsymbol{w}}_T) - f(\bar{\boldsymbol{w}}^*) \le \min_{\xi_1,\cdots,\xi_T} \frac{1}{\delta T}\sum_{t=1}^{T}\left[4\xi_t R^2 + \frac{qc^2}{\lambda t + \sum_{i=1}^{t}\xi_i}\right] \ . \tag{B.15}$$

Setting all the ξ_i to the same value ξ, the last term in the last equation can be upper bounded by

$$A_T = \min_{\xi} \frac{1}{\delta}\left[4\xi R^2 + \frac{1}{T}\sum_{t=1}^{T}\frac{qc^2}{t(\lambda + \xi)}\right] \tag{B.16}$$

This term is smaller than any specific setting of ξ, in particular if we set ξ to 0, we have that $A_T \le$

$\frac{qc^2(1+\log T)}{\delta \lambda T}$. On the other hand setting optimally the expression over ξ and over-approximating we have that $A_T \leq \frac{4cR\sqrt{q}\sqrt{1+\log T}}{\delta\sqrt{T}}$. Taking the minimum of these two quantities we obtain the stated bound. □

Proof of Theorem 4.3

Proof: Using Lemma 4.1, with simplified update rule (4.6) at fixed learning rate η_0, we have that

$$\eta_0 U \leq \sum_{t \in \mathcal{U}} \eta_0 \ell(\bar{\boldsymbol{u}}, \boldsymbol{x}_t, y_t) + \|\bar{\boldsymbol{u}}\|_{2,p} \sqrt{qU\left(\eta_0^2 L^2 + 2\eta_0\right)}\ . \tag{B.17}$$

Solving for U and overapproximating we obtain the stated bound.

For the second part of the theorem, from inequality (B.3), we have

$$\|\bar{\boldsymbol{\theta}}_{T+1}\|_{2,q}^2 \leq qU\left(\eta_0^2 L^2 + 2\eta_t\right)\ . \tag{B.18}$$

So we can write

$$\|\bar{\boldsymbol{\theta}}_{T+1}\|_{2,q} \leq \eta_0 \sqrt{qU(2/\eta_0 + L^2)}, \tag{B.19}$$

and using the bound of U and the hypothesis of linear separability, we have

$$\|\bar{\boldsymbol{\theta}}_{T+1}\|_{2,q} \leq \eta \sqrt{q^2 \|\bar{\boldsymbol{u}}\|_{2,p}^2 (2/\eta + L^2)^2} = q\|\bar{\boldsymbol{u}}\|_{2,p}\left(2 + \eta L^2\right)\ . \tag{B.20}$$

Using the relation $\|\bar{\boldsymbol{w}}_t\|_{2,p} = \frac{1}{q}\|\bar{\boldsymbol{\theta}}_t\|_{2,q}$, that holds for the 4-th Property in Lemma 4.2, we have the stated bound on R. □

Proof of Theorem 4.5

Proof: (Sketch) Using the hypothesis of this theorem, we have

$$\|\partial \ell\left(\bar{\boldsymbol{w}}_t, \bar{\boldsymbol{\phi}}(\boldsymbol{x}_t, \cdot), y_t\right)\|_{2,q} \leq LF^{1/q} \max_{j=1,\dots,F} \|\phi^j(\boldsymbol{x}_t, \cdot)\|_2 \leq LF^{1/q}$$

The function $h(\bar{\boldsymbol{w}})$ in (4.9) is λ/q-strongly convex w.r.t. the norm $\|\cdot\|_{2,q}$, for any $\alpha \geq 0$. Hence, using Theorem 4.4, with $\eta = 1$ and $g = h$, and using Markov inequality as in Shalev-Shwartz *et al.* (2007) we prove the stated result. ☐

Proof of Theorem 4.6

Proof: (Sketch) From standard Legendre-Fenchel duality, we have that $\nabla h^*(\bar{\boldsymbol{\theta}}) = \arg\max_{\bar{\boldsymbol{w}}} \ \bar{\boldsymbol{w}} \cdot \bar{\boldsymbol{\theta}} - h(\bar{\boldsymbol{w}})$. Setting to zero the derivative of this argmax we have that $\boldsymbol{w}^j$ must must be proportional to $\boldsymbol{\theta}^j$, that is $\boldsymbol{w}^j = c_j \boldsymbol{\theta}^j / \|\boldsymbol{\theta}^j\|$, where c_j are real numbers. So we can focus on the coefficients c_j, rewriting the argmax:

$$\arg\max_{\boldsymbol{c}} \ \boldsymbol{c} \cdot \boldsymbol{a} - \alpha \|\boldsymbol{c}\|_1 - \lambda/2 \ \|\boldsymbol{c}\|_p^2,$$

where $\boldsymbol{a} = [\|\boldsymbol{\theta}_1\|, \cdots, \|\boldsymbol{\theta}_F\|]$, $\boldsymbol{c} = [c_1, \cdots, c_F]$. This problem is analyzed in Section 7.2 of Xiao (2010), and using that theorems we have the stated result. ☐

Proof of Proposition 5.2

Proof: (Sketch) Define

$$\Delta_A(z_m^{(p)}, \mathcal{Z}_m^{(p)}) = \min_{z' \in \mathcal{Z}_m^{(p)}} \Delta_{01}(z_m^{(p)}, z') = \mathbf{1}(z_m^{(p)} \notin \mathcal{Z}_m^{(p)})$$

the ambiguous loss for a single region $\boldsymbol{x}_m$ and an attribute p, with $\mathcal{Z}_m^{(p)}$ being the corresponding CLS for the region. So we can use Proposition 5.1 to obtain

$$E[\Delta_{01}(z_m^{(p)}, y_m^{(p)})] \leq \frac{1}{1-\eta} E[\Delta_A(z_m^{(p)}, \mathcal{Z}_m^{(p)})] \ .$$

Using the definition of the true 0/1 loss and the linearity of expectation, and summing over m and p, we have

$$E\left[\boldsymbol{\Delta}_{01}(\boldsymbol{Z}, \boldsymbol{Y})\right] \leq \frac{1}{1-\eta} E\left[\sum_{m=1}^{M} \sum_{p=1}^{P} \mathbf{1}(z_m^{(p)} \notin \mathcal{Z}_m^{(p)})\right] \ ,$$

while using the relationship that $\bigcup_{m,p} \mathcal{Z}_m^{(p)} \supseteq \mathcal{Z}$, we obtain

$$E\left[\sum_{m=1}^{M} \sum_{p=1}^{P} \mathbf{1}(z_m^{(p)} \notin \mathcal{Z}_m^{(p)})\right] \leq E\left[\boldsymbol{\Delta}_A(\boldsymbol{Z}, \mathcal{Z})\right] \ .$$

Combining them results in the proposed proposition. □

Proof of Proposition 5.3

Proof: Define $\hat{\boldsymbol{z}} = \arg\max_{\boldsymbol{z}\in\mathcal{Y}^M} \ \mathbf{S}(\mathcal{X}, \boldsymbol{z}; \boldsymbol{w})$. If $\hat{\boldsymbol{Z}} \in \mathcal{Z}$ then $\ell_A\left(\mathcal{X}, \mathcal{Z}; \boldsymbol{w}\right) \geq \boldsymbol{\Delta}_A\left(\mathcal{X}, \mathcal{Z}; \boldsymbol{w}\right) = 0$. We now consider the case in which $\hat{\boldsymbol{Z}} \notin \mathcal{Z}$. We have that

$$\begin{aligned}
\boldsymbol{\Delta}_A\left(\mathcal{X}, \mathcal{Z}; \boldsymbol{w}\right) &\leq \boldsymbol{\Delta}_A(\hat{\boldsymbol{Z}}, \mathcal{Z}) + \mathbf{S}_{\boldsymbol{w}}(\mathcal{X}, \hat{\boldsymbol{Z}}) - \max_{\boldsymbol{z}\in\mathcal{Z}} \mathbf{S}_{\boldsymbol{w}}(\mathcal{X}, \boldsymbol{Z}) \\
&\leq \max_{\bar{\boldsymbol{Z}}\notin\mathcal{Z}} \left(\boldsymbol{\Delta}_A(\bar{\boldsymbol{Z}}, \mathcal{Z}) + \mathbf{S}_{\boldsymbol{w}}(\mathcal{X}, \bar{\boldsymbol{Z}})\right) - \max_{\boldsymbol{Z}\in\mathcal{Z}} \mathbf{S}_{\boldsymbol{w}}(\mathcal{X}, \boldsymbol{Z}) \leq \ell_A\left(\mathcal{X}, \mathcal{Z}; \boldsymbol{w}\right) \ .
\end{aligned}$$

□

Bibliography

Andrews, S., Tsochantaridis, I., and Hofmann, T. (2003). Support vector machines for multiple-instance learning. In *Advances in Neural Information Processing Systems*.

Bach, F. R., Lanckriet, G. R. G., and Jordan, M. I. (2004). Multiple kernel learning, conic duality, and the SMO, algorithm. In *Proceedings of the International Conference on Machine Learning*.

Barnard, K., Duygulu, P., Forsyth, D., de Freitas, N., Blei, D., and Jordan, M. (2003). Matching words and pictures. *Journal of Machine Learning Research*, **3**, 1107–1135.

Bartlett, P., Hazan, E., and Rakhlin, A. (2008). Adaptive online gradient descent. In *Advances in Neural Information Processing Systems 20*.

Bay, H., Ess, A., Tuytelaars, T., and Gool, L. V. (2008). Surf: Speeded up robust features. *Computer Vision and Image Understanding*, **110**, 346–359.

Berg, T., Berg, A., Edwards, J., and Forsyth, D. (2004a). Names and faces in the news. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*.

Berg, T., Berg, A., Edwards, J., and Forsyth, D. (2004b). Who's in the picture? In *Advances in Neural Information Processing Systems*.

Bertsekas, D. P. (2003). *Convex Analysis and Optimization*. Athena Scientific.

Bickel, S., Brckner, M., and Scheffer, T. (2007). Discriminative learning for differing training and test distributions. In *Proceedings of the International Conference on Machine Learning*.

Bishop, C. M. (2006). *Pattern Recognition and Machine Learning*. Springer.

Bosch, A., Zisserman, A., and Munoz, X. (2007). Representing shape with a spatial pyramid kernel. In *Proceedings of 6th ACM International Conference on Image and Video Retrieval*.

Boutell, M. R., Luo, J., Shen, X., and Brown, C. M. (2004). Learning multi-label scene classification. *Pattern Recognition*, **37**, 1537–439.

Boyd, S. and Vandenberghe, L. (2004). *Convex Optimization*. Cambridge University Press.

Bunescu, R. C. and Mooney, R. J. (2007). Multiple instance learning for sparse positive bags. In *Proceedings of the International Conference on Machine Learning*.

Cavallanti, G., Cesa-Bianchi, N., and Gentile, C. (2008). Linear algorithms for online multitask classification. In *Proceedings of the Annual Conference on Learning Theory*.

Cavallanti, G., Cesa-Bianchi, N., and Gentile, C. (2009). Linear algorithms for online multitask classification. In *Proceedings of the Annual Conference on Learning Theory*.

Cesa-Bianchi, N. and Lugosi, G. (2006). *Prediction, learning, and games*. Cambridge University Press.

Cesa-Bianchi, N., Conconi, A., and Gentile, C. (2004). On the generalization ability of on-line learning algorithms. *IEEE Transaction on Information Theory*, **50**(9), 2050–2057.

Chang, C. C. and Lin, C. J. (2001). *LIBSVM: A Library for Support Vector Machines*. Software available at `www.csie.ntu.edu.tw/~cjlin/libsvm`.

Chang, W. and Sridhar, M. (2008). Manifold alignment using procrustes analysis. In *Proceedings of the International Conference on Machine Learning*.

Chapelle, O. (2007). Training a support vector machine in the primal. *Neural Computation*, **19**(5), 1155–1178.

Chapelle, O., Vapnik, V., Bousquet, O., and Mukherjee, S. (2002). Choosing multiple parameters for support vector machines. *Machine Learning*, **46**(1), 131159.

Cormen, T. H., Leiserson, C. E., Rivest, R. L., and Stein, C. (2003). *Introduction to Algorithms*. MIT Press.

Cortes, C. (2009). Can learning kernels help performance? (invited talk). In *Proceedings of the International Conference on Machine Learning*.

Cortes, C., Mohri, M., and Rostamizadeh, A. (2009a). Learning non-linear combinations of kernels. In *Advances in Neural Information Processing Systems 22*.

Cortes, C., Gretton, A., Lanckriet, G., Mohri, M., and Rostamizadeh., A. (2009b). Proceedings of the Advances in Neural Information Processing Systems Workshop on Kernel Learning: Automatic Selection of Optimal Kernels.

Cortes, C., Mohri, M., and Rostamizadeh, A. (2010). Two-stage learning kernel algorithms. In *Proceedings of the International Conference on Machine Learning*.

Cour, T., Sapp, B., Jordan, C., and Taskar, B. (2009). Learning from ambiguously labeled images. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*.

Crammer, K. and Singer, Y. (2002). On the algorithmic implementation of multiclass kernel-based vector machines. *Journal of Machine Learning Research*, **2**, 265–292.

Crammer, K., Dekel, O., Keshet, J., Shalev-Shwartz, S., and Singer, Y. (2006). Online passive-aggressive algorithms. *Journal of Machine Learning Research*, **7**, 551–585.

Cristianini, N. and Shawe-Taylor, J. (2000). *An Introduction to Support Vector Machines and other kernel-based learning methods*. Cambridge University Press.

Dai, W., Jin, O., Xue, G.-R., Yang, Q., and Yu, Y. (2009). Eigentransfer: a unified framework for transfer learning. In *Proceedings of the International Conference on Machine Learning*.

Dasgupta, S. and Langford, J. (2009). Active learning tutorial.

Daumé III, H. (2007). Frustratingly easy domain adaptation. In *Proceedings of Annual Meeting of the Association for Computational Linguistics*.

Deng, J., Dong, W., Socher, R., Li, L.-J., Li, K., and Fei-Fei, L. (2009). Imagenet: A large-scale hierarchical image database. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*.

Deschacht, K. and Moens, M.-F. (2009). Semi-supervised semantic role labeling using the latent words language model. In *Proceedings of the Conference on Empirical Methods in Natural Language Processing*.

Dietterich, T. G., Lathrop, R. H., Lozano-Perez, T., and Pharmaceutical, A. (1997). Solving the multiple-instance problem with axis-parallel rectangles. *Artificial Intelligence*, **39**, 31–71.

Do, C. B., Le, Q. V., and Foo, C.-S. (2009). Proximal regularization for online and batch learning. In *Proceedings of the International Conference on Machine Learning*.

Downs, T., Gates, K., and Masters, A. (2001). Exact simplification of support vectors solutions. *Journal of Machine Learning Research*, **2**, 293–297.

Duan, L., Tsang, I., Xu, D., and Maybank, S. (2009). Domain transfer svm for video concept detection. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*.

Duchi, J. and Singer, Y. (2009). Efficient online and batch learning using forward-backward splitting. *Journal of Machine Learning Research*, **10**, 2873–2898.

Duda, R. O., Hart, P. E., and Stork, D. G. (2001). *Pattern Classification*. Wiley, New York.

Eaton, E., desJardins, M., and Lane, T. (2008). Modeling transfer relationships between learning tasks for improved inductive transfer. In *Proceedings of the European Conference on Machine Learning*.

Eichner, M. and Ferrari, V. (2009). Better appearance models for pictorial structures. In *Proceedings of the British Machine Vision Conference*.

Ernst, M. O. and Banks, M. S. (2002). Humans integrate visual and haptic information in a statistically optimal fashion. *Nature*, **415**(6870), 429–433.

Ernst, M. O. and Bulthoff, H. H. (2004). Merging the senses into a robust percept. *Trends in Cognitive Sciences*, **8**(4), 162–169.

Everingham, M., Sivic, J., and Zisserman, A. (2006). Hello! my name is... buffy - automatic naming of characters in tv video. In *Proceedings of the British Machine Vision Conference*.

Fan, R.-E., Chang, K.-W., Lin, C.-J., Keerthi, S. S., and Sundarajan, S. (2008). LIBLINEAR: A library for large linear classification. *Journal of Machine Learning Research*, **9**, 1871–1874.

Farhadi, A., Endres, I., Hoiem, D., and Forsyth, D. (2009). Describing objects by their attributes. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*.

Fei-Fei, L., Fergus, R., and Perona, P. (2004). One-shot learning of object categories. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, **28**(4), 594–611.

Felzenszwalb, P., Girshick, R., McAllester, D., and Ramanan, D. (2010). Object detection with discriminatively trained part based models. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, **10**(9).

Felzenszwalb, P. F. and Huttenlocher, D. P. (2004). Efficient graph-based image segmentation. *International Journal of Computer Vision*, **59**(2).

Ferrari, V., Marin, M., and Zisserman, A. (2008a). Pose search: retrieving people using their pose. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*.

Ferrari, V., Marin, M., and Zisserman, A. (2008b). Progressive search space reduction for human pose estimation. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*.

Fink, M., Shalev-Shwartz, S., Singer, Y., and Ullman, S. (2006). Online multiclass learning by interclass hypothesis sharing. In *Proceedings of the International Conference on Machine Learning*.

Finley, T. and Joachims, T. (2008). Training structural svms when exact inference is intractable. In *Proceedings of the International Conference on Machine Learning*.

Gai, K., Chen, G., and Zhang, C. (2010). Learning kernels with radiuses of minimum enclosing balls. In *Advances in Neural Information Processing Systems 23*.

Gehler, P. and Nowozin, S. (2009a). Let the kernel figure it out: Principled learning of pre-processing for kernel classifiers. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*.

Gehler, P. and Nowozin, S. (2009b). On feature combination for multiclass object classification. In *Proceedings of IEEE International Conference on Computer Vision*.

Gentile, C. (2003). The robustness of the p-norm algorithms. *Machine Learning*, **53**(3), 265–299.

Gönen, M. and Alpaydin, E. (2010). Cost-conscious multiple kernel learning. *Pattern Recognition Letter*, **31**, 959–965.

Grandvalet, Y. (2002). Logistic regression for partial labels. In *Proceedings of the International Conference on Information Processing and Management of Uncertainty*.

Grangier, D. and Bengio, S. (2008). A discriminative kernel-based approach to rank images from text queries. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, **30**(8), 1371–1384.

Griffin, G., Holub, A., and Perona, P. (2007). Caltech 256 object category dataset. Technical Report UCB/CSD-04-1366, California Institue of Technology.

Guillaumin, M., Mensink, T., Verbeek, J., and Schmid, C. (2008). Automatic face naming with caption-based supervision. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*.

Guillaumin, M., Verbeek, J., and Schmid, C. (2010). Multiple instance metric learning from automatically labeled bags of faces. In *Proceedings of the European Conference on Computer Vision*.

Gupta, A. and Davis, L. (2008). Beyond nouns: Exploiting prepositions and comparative adjectives for learning visual classifiers. In *Proceedings of the European Conference on Computer Vision*.

Hüllermeier, E. and Beringe, J. (2006). Learning from ambiguously labelled example. *Intelligent Data Analysis*, **10**, 419–439.

Hush, D., Kelly, P., Scovel, C., and Steinwart, I. (2006). Qp algorithms with guaranteed accuracy and run time for support vector machines. *Journal of Machine Learning Research*, **7**.

Jie, L. and Orabona, F. (2010). Learning from candidate labeling sets. In *Advances in Neural Information Processing Systems 23*.

Jie, L., Orabona, F., and Caputo, B. (2009a). An online framework for learning novel concepts over multiple cues. In *Proceedings of the 9th Asian Conference on Computer Vision*.

Jie, L., Caputo, B., and Ferrari, V. (2009b). Who's doing what: Joint modeling of names and verbs for simultaneous face and pose annotation. In *Advances in Neural Information Processing Systems 22*.

Jie, L., Orabona, F., Fornoni, M., Caputo, B., and Cesa-Bianchi, N. (2010). OM-2: An online multi-class multi-kernel learning algorithm. In *Proceedings of the 4th IEEE Online Learning for Computer Vision Workshop*.

Jie, L., Orabona, F., Caputo, B., and Ferrari, V. (2011a). Learning from images with captions using the maximum margin set algorithm. Technical Report Idiap-RR-30-2011, Idiap Research Institute.

Jie, L., Tommasi, T., and Caputo, B. (2011b). Multiclass transfer learning from unconstrained priors. In *Proceedings of the 13th International Conference on Computer Vision*.

Jin, R. and Ghahramani, Z. (2002). Learning with multiple labels. In *Advances in Neural Information Processing Systems*.

Joachims, T. (1998). Making large-scale SVM learning practical. In B. Schölkopf, C. Burges, and A. Smola, editors, *Advances in Kernel Methods – Support Vector Learning*, pages 169–185. MIT Press.

Kakade, S., Shalev-Shwartz, S., and Tewari, A. (2009). On the duality of strong convexity and strong smoothness: Learning applications and matrix regularization. Technical report, TTI.

Kloft, M., Brefeld, U., Sonnenburg, S., Laskov, P., Müller, K.-R., and Zien, A. (2009). Efficient and accurate lp-norm multiple kernel learning. In *Advances in Neural Information Processing Systems 22*.

Lampert, C. H., Nickisch, H., and Harmeling, S. (2009). Learning to detect unseen object classes by between-class attribute transfer. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*.

Lanckriet, G., Cristianini, N., Bartlett, P., and Ghaoui, L. E. (2004a). Learning the kernel matrix with semidefinite programming. *Journal of Machine Learning Research*, **5**.

Lanckriet, G. R. G., Cristianini, N., Bartlett, P., El Ghaoui, L., and Jordan, M. I. (2004b). Learning the kernel matrix with semidenite programming. *Journal of Machine Learning Research*, **5**, 27–72.

Lazebnik, S., Schmid, C., and Ponce, J. (2006). Beyond bags of features: Spatial pyramid matching for recognizing natural scene categories. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*.

LeCun, Y., Bottou, L., Bengio, Y., and Haffner, P. (1998). Gradient-based learning applied to document recognition. *Proceedings of the IEEE*, **86**, 2278–2324.

Leibe, B. and Schiele, B. (2003). Analyzing appearance and contour based methods for object categorization. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*.

Li, L.-J., Su, H., Xing, E. P., and Fei-Fei, L. (2010). Object Bank: A High-Level Image Representation for Scene Classification & Semantic Feature Sparsification. In *Advances in Neural Information Processing Systems*.

Linde, O. and Lindeberg, T. (2005). Object recognition using composed receptive field histograms of higher dimensionality. In *Proceedings of the International Conference on Pattern Recognition*.

Littlestone, N. and Warmuth, M. (1989). Weighted majority algorithm. In *IEEE Symposium on Foundations of Computer Science*.

Lowe, D. (2004). Distinctive image features from scale-invariant keypoints. *International Journal of Computer Vision*, **60**(2), 91–110.

Luo, J., Pronobis, A., Caputo, B., and Jensfelt, P. (2007). Incremental learning for place recognition in dynamic environments. In *Proceedings of the IEEE/RSJ International Conference on Intelligent Robots and Systems*.

Micchelli, C. A. and Pontil, M. (2005). Learning the kernel function via regularization. *Journal of Machine Learning Research*, **6**, 1099–1125.

Mozos, O. M., Stachniss, C., and Burgard, W. (2005). Supervised learning of places from range data using adaboost. In *Proceedings of the IEEE International Conference on Robotics and Automation*.

Nath, J. S., Dinesh, G., Ramanand, S., Bhattacharyya, C., Ben-Tal, A., and Ramakrishnan, K. R. (2009). On the algorithmics and applications of a mixed-norm based kernel learning formulation. In *Advances in Neural Information Processing Systems 22*.

Nilsback, M. E. and Caputo, B. (2004). Cue integration through discriminative accumulation. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*.

Nilsback, M.-E. and Zisserman, A. (2006). A visual vocabulary for flower classification. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*.

Ojala, T., Pietikaäinen, M., and Maäenpaäaä, T. (2002). Multiresolution gray-scale and rotation invariant texture classification with local binary patterns. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, **24**(7), 971–987.

OpenNLP (2010). `http://incubator.apache.org/opennlp/`.

Orabona, F. (2009). *DOGMA: a MATLAB toolbox for Online Learning*. Software available at `http://dogma.sourceforge.net`.

Orabona, F. and Jie, L. (2011). Ultra-fast optimization algorithm for sparse mkl. In *Proceedings of the 28th International Conference on Machine Learning*.

Orabona, F., Keshet, J., and Caputo, B. (2009). Bounded kernel-based online learning. *Journal of Machine Learning Research*, **10**(Nov), 2643–2666.

Orabona, F., Jie, L., and Caputo, B. (2010). Online-batch strongly convex multi kernel learning. In *Proceedings of the 23rd IEEE Conference on Computer Vision and Pattern Recognition*.

Pinto, N., Cox, D. D., and Dicarlo, J. J. (2008). Why is real-world visual object recognition hard? *PLoS Computational Biology*, **4**, e27.

Platt, J. C. (1999). *Fast training of support vector machines using sequential minimal optimization*, pages 185–208. MIT Press, Cambridge, MA, USA.

Polikar, R. (2006). Ensemble based systems in decision making. *IEEE Circuits and Systems Magazine*, **6**(3), 21–45.

Pronobis, A., Luo, J., and Caputo, B. (2010). The more you learn, the less you store: Memory-controlled incremental SVM for visual place recognition. *Image and Vision Computing*, **28**, 1080-1097.

Quattoni, A., Collins, M., and Darrell, T. (2008). Transfer learning for image classication with sparse prototype representations. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*.

Raina, R., Battle, A., Lee, H., Packerand, B., and Ng, A. Y. (2007). Self-taught learning: transfer learning from unlabeled data. In *Proceedings of the International Conference on Machine Learning*.

Rakotomamonjy, A., Bach, F., Grandvalet, Y., and Canu, S. (2008). SimpleMKL. *Journal of Machine Learning Research*, **9**, 2491–2521.

Rodriguez, Y. (2006). *Face Detection and Verification using Local Binary Patterns*. Ph.D. thesis, École Polytechnique Fédérale de Lausanne.

Rohrbach, M., Stark, M., Szarvas, G., Gurevych, I., and Schiele, B. (2010). What helps where? and why? semantic relatedness for knowledge transfer. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*.

Rosenblatt, F. (1958). The Perceptron: A probabilistic model for information storage and organization in the brain. *Psychological Review*, **65**, 386–407.

Rosenstein, M., Marx, Z., and Kaelbling, L. P. (2005). To transfer or not to transfer. In *Advances in Neural Information Processing Systems*.

Saenko, K., Kulis, B., Fritz, M., and Darrell, T. (2010). Adapting visual category models to new domains. In *Proceedings of the European Conference on Computer Vision*.

Sanderson, C. and Paliwal, K. K. (2004). Identity verification using speech and face information. *Digital Signal Processing*, **14**(5), 449–480.

Schapire, R. and Singer, Y. (1999). Improved boosting algorithms using confidence-rated predictions. *Machine Learning*, **37**, 297–336.

Schölkopf, B. and Smola, A. J. (2001). *Learning with Kernels: Support Vector Machines, Regularization, Optimization, and Beyond*. MIT Press.

Schölkopf, B., Herbrich, R., Smola, A., and Williamson, R. (2000). A generalized representer theorem. In *Proceedings of the Annual Conference on Learning Theory*.

Shalev-Shwartz, S. (2007). *Online Learning: Theory, Algorithms, and Applications*. Ph.D. thesis, The Hebrew University of Jerusalem.

Shalev-Shwartz, S. and Kakade, S. M. (2008). Mind the duality gap: Logarithmic regret algorithms for online optimization. In *Advances in Neural Information Processing Systems 21*.

Shalev-Shwartz, S. and Singer, Y. (2007). Logarithmic regret algorithms for strongly convex repeated games. Technical Report 2007-42, The Hebrew University.

Shalev-Shwartz, S. and Srebro, N. (2008). SVM, optimization: inverse dependence on training set size. In *Proceedings of the 25th International Conference on Machine Learning*.

Shalev-Shwartz, S. and Tewari, A. (2009). Stochastic methods for l1 regularized loss minimization. In *Proceedings of the 26th International Conference on Machine Learning*.

Shalev-Shwartz, S., Singer, Y., and Srebro, N. (2007). Pegasos:primal estimated sub-gradient solver for svm. In *Proceedings of the International Conference on Machine Learning*.

Shechtman, E. and Irani, M. (2007). Matching local self-similarities across images and videos. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*.

Shental, N., Bar-Hillel, A., Hertz, T., and Weinshall, D. (2003). Computing gaussian mixture models with em using equivalence constraints. In *Advances in Neural Information Processing Systems*.

Sivic, J. and Zisserman, A. (2003). Video Google: A text retrieval approach to object matching in videos. In *Proceedings of the International Conference on Computer Vision*.

Smola, A. J., Vishwanathan, S. V. N., and Hofmann, T. (2005). Kernel methods for missing variables. In *Proceedings of the International Conference on Artificial Intelligence and Statistics*.

Sonnenburg, S., Rätsch, G., Schäfer, C., and Schölkopf, B. (2006). Large scale multiple kernel learning. *Journal of Machine Learning Research*, **7**.

Stark, M., Goesele, M., and Schiele, B. (2009). A shape-based object class model for knowledge transfer. In *Proceedings of the International Conference on Computer Vision*.

Tang, K., Tappen, M., Sukthankar, R., and Lampert, C. (2010). Optimizing one-shot recognition with micro-set learning. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*.

Thrun, S. (1996). Is learning the n-th thing any easier than learning the first? In *Advances in Neural Information Processing Systems*.

Tomioka, R. and Suzuki, T. (2010). Sparsity-accuracy trade-off in MKL. Technical Report arXiv:1001.2615. Available at `http://arxiv.org/abs/1001.2615`.

Tommasi, T., Orabona, F., and Caputo, B. (2008). Discriminative cue integration for medical image annotation. *Pattern Recognition Letters, Special issue on ImageCLEF Med benchmark evaluation*, **29**, 1996–2002.

Tommasi, T., Orabona, F., and Caputo, B. (2010). Safety in numbers: Learning categories from few examples with multi model knowledge transfer. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*.

Torresani, L., Szummer, M., and Fitzgibbon, A. (2010). Efficient object category recognition using classemes. In *Proceedings of the European Conference on Computer Vision*.

Tsochantaridis, I., Hofmann, T., Joachims, T., and Altun, Y. (2004). Support vector machine learning for interdependent and structured output spaces. In *Proceedings of the International Conference on Machine Learning*.

Tsochantaridis, I., Joachims, T., Hofmann, T., and Altun, Y. (2005). Large margin methods for structured and interdependent output variables. *Journal of Machine Learning Research*, **6**, 1453–1484.

Tuzel, O., Porikli, F., and Meer, P. (2007). Human detection via classification on Riemannian manifolds. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*.

van de Sande, K., Gevers, T., and Snoek, C. (2008). Evaluation of color descriptors for object and scene recognition. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*.

Varma, M. and Babu, B. R. (2009). More generality in efficient multiple kernel learning. In *Proceedings of the International Conference on Machine Learning*.

Varma, M. and Ray, D. (2007). Learning the discriminative power-invariance trade-off. In *Proceedings of the International Conference on Computer Vision*.

Viola, P. and Jones, M. (2004). Robust real-time object detection. *International Journal of Computer Vision*, **57**(2), 137–154.

Vishwanathan, S. V. N., Sun, Z., Theera-Ampornpunt, N., and Varma, M. (2010). Multiple kernel learning and the SMO algorithm. In *Advances in Neural Information Processing Systems 23*.

Vogel, J. and Schiele, B. (2008). Semantic modeling of natural scenes for content-based image retrieval. *International Journal of Computer Vision*, **10**, 133–157.

Wang, G. and Forsyth, D. (2009). Joint learning of visual attributes, object classes, and visual saliency. In *Proceedings of the International Conference on Computer Vision*.

Wang, J., Markert, K., and Everingham, M. (2009). Learning models for object recognition from natural language descriptions. In *Proceedings of the British Machine Vision Conference*.

Wang, Y. and Mori, G. (2010). A discriminative latent model of image region and object tag correspondence. In *Advances in Neural Information Processing Systems*.

Wolpert, D. H. (1992). Stacked generalization. *Neural Networks*, **5**(2).

Xiao, L. (2010). Dual averaging methods for regularized stochastic learning and online optimization. *Journal of Machine Learning Research*, **11**, 2543–2596.

Xu, Z., Jin, R., King, I., and Lyu, M. R. (2008). An extended level method for efficient multiple kernel learning. In *Advances in Neural Information Processing Systems 21*.

Yao, Y. and Doretto, G. (2010). Boosting for transfer learning with multiple sources. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*.

Yu, C.-N. and Joachims, T. (2009). Learning structural svms with latent variables. In *Proceedings of the International Conference on Machine Learning*.

Yuille, A. and Rangarajan, A. (2003). The concave-convex procedure. *Neural Computation*, **15**, 915–936.

Zhang, M.-L. and Zhou, Z.-H. (2008). M^{3}MIML: A maximum margin method for multi-instance multi-label learning. In *Proceedings of the International Conference on Data Mining*.

Zhou, Z.-H. and Zhang, M.-L. (2006). Multi-instance multi-label learning with application to scene classification. In *Advances in Neural Information Processing Systems*.

Zhu, X. (2005). Semi-supervised learning literature survey. Technical Report 1530, Computer Sciences, University of Wisconsin-Madison.

Zien, A. and Ong, C. S. (2007). Multiclass multiple kernel learning. In *Proceedings of the International Conference on Machine Learning*.

Curriculum Vitae

LUO, Jie

Idiap Research Institute

Rue Marconi 19

CH-1920 Martigny

Switzerland

E-mail: jluo@idiap.ch

Education

- Doctor of Philosophy (Ph.D.)

  School of Computer and Communication Science

  Ecole Polytechnique Fédérele de Lausanne (EPFL), Switzerland

  May 2007 - Dec 2011

- Master of Science (M.Sc.)

  Department of Computer and System Science

  Royal Institute of Technology (KTH), Stockholm, Sweden

  Aug 2005 - Dec 2006

- Bachelors in Engineering (B.E.)

  Department of Communication Engineering, College of Electronic Information

  Wuhan University, Wuhan, China

  Aug 2001 - June 2005

Professional Experience

- Research Assistant
  Idiap Research Institute
  Martigny, Switzerland
  April 2007 - present

Publications

Journal Papers

- **Jie, L.**, Orabona, F., Caputo, B., and Ferrari, V. "Learning from Images with Captions Using the Maximum Margin Set Algorithm". Manuscript submitted to *IEEE Transactions on Pattern Analysis and Machine Intelligence*.

- Orabona, F., **Jie, L.**, and Caputo, B. "Multi Kernel Learning with Online-Batch Optimization". Manuscript submitted to *Journal of Machine Learning Research*. (Under minor revisions)

- Pronobis, A., **Jie, L.**, and Caputo, B. "The More you Learn, the Less you Store: Memory-controlled Incremental SVM for Visual Place Recognition". *Image and Vision Computing*, 28(7), 2010.

- Orabona, F., Castellini, C., Caputo, B., **Jie, L.**, and Sandini, G. "On-line Independent Support Vector Machines". *Pattern Recognition*, 43(4), 2010.

Conference Proceedings

- **Jie, L.**, Tommasi, T., and Caputo, B. "Multiclass Transfer Learning from Unconstrained Priors". In *Proceedings of the 13th International Conference on Computer Vision* (ICCV11). Barcelona, Spain. November, 2011.

- Özcan, M., **Jie, L.**, Ferrari, V., and Caputo, B. "A Large-Scale Database of Images and Captions for Automatic Face Naming". In *Proceedings of the 22nd British Machine Vision Conference* (BMVC11). Dundee, UK. August, 2011.

- Orabona, F. and **Jie, L.**. “Ultra-Fast Optimization Algorithm for Sparse Multi Kernel Learning”. In *Proceedings of the 28th International Conference on Machine Learning* (ICML11). Bellevue, Washington. June, 2011.

- **Jie, L.** and Orabona, F. “Learning from Candidate Labeling Sets”. In *Advances in Neural Information Processing Systems 23* (NIPS10). Vancouver, Canada. December, 2010.

- **Jie, L.**, Orabona, F., Fornoni, M., Caputo, B., and Cesa-Bianchi, N. “OM-2: An online multi-class multi-kernel learning algorithm”. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, Workshop on On-line Learning for Computer Vision* (OLCV10). San Francisco, California. June, 2010.

- Orabona, F., **Jie, L.**, and Caputo, B. “Online-Batch Strongly Convex Multi Kernel Learning”. In *Proceedings of IEEE Conference on Computer Vision and Pattern Recognition* (CVPR10), San Francisco, California. June 2010.

- **Jie, L.**, Caputo, B., and Ferrari, V. “Who’s Doing What: Joint Modeling of Names and Verbs for Simultaneous Face and Pose Annotation”. In *Advances in Neural Information Processing Systems 22* (NIPS09). Vancouver, Canada. December, 2009.

- **Jie, L.**, Orabona, F., and Caputo, B. “An online framework for learning novel concepts over multiple cues”. In *Proceedings of the Asian Conference on Computer Vision* (ACCV09). Xi’an, China, September, 2009.

- Weinshall, D., Hermansky, H., Zweig, A., **Jie, L.**, Jimison, H., Ohl, F., and Pavel, M. “Beyond Novelty Detection: Incongruent Events, when General and Specific Classifiers Disagree”. In *Advances in Neural Information Processing Systems 21* (NIPS08). Vancouver, Canada. December, 2008.

- **Jie, L.**, Caputo, B., Zweig, A., Bach, J.-H., and Anemuller, J. “Object Category Detection Using Audio-visual Cues”. In *Proceedings of the International Conference on Computer Vision System* (ICVS08). Santorini, Greece. May, 2008.

- Ullah, M. M., Pronobis, A., Caputo, B., **Luo, J.**, Jensfelt, P., and Christensen, H. I. “Towards Robust Place Recognition for Robot Localization”. In *Proceedings of the IEEE International*

Conference on Robotics and Automation (ICRA08). Pasadena, California. May, 2008.

- **Luo, J.**, Pronobis, A., Caputo, B. and Jensfelt, P. "Incremental Learning for Place Recognition in Dynamic Environments". In *Proceedings of the IEEE/RSJ International Conference on Intelligent Robots and Systems* (IROS07). San Diego, California. October, 2007.
- **Luo, J.**, Pronobis, A., and Caputo, B. "SVM-based Transfer of Visual Knowledge Across Robotic Platforms". In *Proceedings of the International Conference on Computer Vision System* (ICVS07). Bielefeld, Germany. March, 2007.