Collaborative learning from multiple data sources

Li, Guangxia

2013

Li, G. (2013). Collaborative learning from multiple data sources. Doctoral thesis, Nanyang Technological University, Singapore.

<https://hdl.handle.net/10356/55062>

<https://doi.org/10.32657/10356/55062>

COLLABORATIVE LEARNING FROM MULTIPLE DATA SOURCES

LI GUANGXIA

School of Computer Engineering

A thesis submitted to the Nanyang Technological University
in partial fulfilment of the requirement for the degree of
Doctor of Philosophy

2013

Acknowledgments

I would like to express my sincere appreciation to my supervisors Dr. Kuiyu Chang and Dr. Steven C. H. Hoi. Under their supervision, I have learned not only cutting-edge knowledge on machine learning and data mining, but also an inspiring and down-to-earth working attitude.

I would also like to thank my colleagues at Nanyang Technological University for their friendship and insightful discussions.

Last but not least, I wish to thank my family, for their unwavering love and constant encouragement, without which I would not have been able to complete this journey of self-discovery.

Abstract

A machine learning classifier can be trained on an labeled input data set, which comprise samples and their corresponding labels, to predict the labels of samples that it has never seen before. The problem of combining several machine learning classifiers to achieve a result that is greater than the sum of its parts has been studied under the guise of collaborative learning. The vast majority of collaborative learning methods simply train several base classifiers and take a majority vote on the predictions made by each base classifier. In this thesis, we study how machine learning classifiers can work collaboratively in either a data-centric or task-centric fashion. Unlike typical methods, we formulate collaborative learning into a coherent optimization framework, in which different classifiers is learned from different representations or partitions of the data.

We first propose a data-centric collaborative method that is related to semi-supervised learning, which aims to improve the performance of a classifier trained on a limited number of labeled data by utilizing unlabeled data. Specifically, we improve the transductive support vector machine (SVM), which is an existing semi-supervised learning algorithm, by employing a multi-view learning paradigm. Multi-view learning makes use of multiple perspectives, so called views, of each data sample. For example, in text classification, the standard view typically contains large number of raw content features such as term frequency, while a secondary view may contain a small but highly-informative number of domain specific features. We propose a novel two-view transductive SVM that takes advantage of both the abundant amount of unlabeled data and their multiple representations, to improve the overall classifier performance. The idea is fairly simple: train a classifier on each of the two views of both labeled and unlabeled data while imposing a global constraint: all classifiers should assign the same class label to the same data sample, labeled or unlabeled. We also incorporate manifold regularization, which is a

type of graph-based semi-supervised learning into our learning framework. The proposed two-view transductive SVM was evaluated on both synthetic and real-life datasets. Experimental results show that our algorithm performs up to 5% better than a single view learning approach, especially when the amount of labeled data is small.

Following this, we consider the situation in which learning tasks are equipped with plain data (only labeled data with conventional representation), but several similar learning problems are presented at the same time. Algorithms working in this scenario are referred to as the task-centric collaborative learning. We first study the problem of online learning to classify instances belonging to several different, but related tasks. Despite their uniqueness, individual classification problems still share certain characteristics with others in the group. Conventional methods treat each task independently, without considering the latent commonality among tasks. We consider the case where the information learned by one task can be used to enhance the learning of other tasks, and proposed a collaborative online multitask learning method that learns several classification models for each of the related tasks in parallel. The basic idea is to first build a generic global model from the pooled task data, and subsequently leverage the global model to build the personalized classification model of individual tasks in a collaborative learning manner.

Building upon the success of collaborative learning, we then tailor the collaborative online classification method to cope with ranking problems. Ranking resembles classification as both assign one of several possible labels to a new instance, but differs in that there is an order relation between the labels. Our proposed collaborative online ranking algorithm is able to rank data generated by a group of tasks by combining group and individual characteristics. To the best of our knowledge, our work is the first learning-to-rank attempt in an online multitask learning setting.

We illustrate the efficacy of the proposed task-centric collaborative online learning on a synthetic dataset and several real-life problems — spam email filtering, bioinformatics classification, and user generated content analysis. Experimental results show that the proposed algorithms are able to take advantage of both the individual and global models to collaboratively learn multiple correlated tasks, achieving an overall improvement in classification/ranking performance. Moreover, our online collaborative algorithms are highly efficient and scalable, and are especially suitable for streaming data or massive datasets that are too large to fit in memory.

Due to the fundamental differences between data-centric and task-centric collaborative learning, it may be difficult, if not impossible, to combine both into a coherent scheme. Thus one major limitation in the scope of our research is that we do not seek to combine them. Instead, we will study both models independently of one another. Moreover, practitioners are free to use our data or task centric collaborative algorithms based on their real-life needs.

Contents

Acknowledgments	i
Abstract	ii
List of Figures	viii
List of Tables	x
List of Abbreviations	xi
List of Notation	xii
1 Introduction	1
1.1 Background and Motivation	1
1.2 Research Objectives	4
1.3 Research Scope and Limitations	5
1.3.1 Data Centric Collaborative Learning	5
1.3.2 Task Centric Collaborative Learning	6
1.4 Approach and Methodology	6
1.4.1 Two-view Transductive SVM	7
1.4.2 Collaborative Online Learning	8
1.4.3 Collaborative Online Ranking	9
1.5 Main Contributions	9
1.6 Organization of the Thesis	10
2 Literature Survey	11
2.1 Semi-supervised Learning	11
2.1.1 Transductive SVM	11
2.1.2 Graph-based Methods	14
2.2 Co-training and Multi-view Learning	16

2.2.1	Co-training	16
2.2.2	Multi-view Learning	17
2.3	Product Review Filtering	18
2.4	Sentiment Analysis	19
2.5	Online Learning	21
2.6	Multitask Learning	23
2.7	Learning to Rank	25
3	Two-view Transductive Support Vector Machine	28
3.1	Motivation	28
3.2	Derivation of the Optimization Problem	30
3.3	Augmented Lagrangian Technique	33
3.4	Two-view Transductive SVM Algorithm	34
3.5	Experimental Results	36
3.5.1	Toy Dataset	36
3.5.2	Ads Dataset	37
3.5.3	WebKB Course Dataset	39
3.5.4	Product Review Dataset	42
3.6	Discussion	46
4	Collaborative Online Multitask Learning	48
4.1	Motivation	48
4.2	Methodology	50
4.2.1	Overview	50
4.2.2	Formulation and Algorithms	50
4.2.3	Extending Confidence Weighted Learning	53
4.2.4	Time and Space Complexity	56
4.3	Experimental Results	57
4.3.1	Benchmark Setup	57
4.3.2	Synthetic Dataset	58
4.3.3	Spam Email Filtering	60
4.3.4	MHC-I Binding Prediction	63

4.3.5	Micro-blog Sentiment Analysis	64
4.4	Discussion	67
5	Collaborative Online Multitask Ranking	69
5.1	Motivation	69
5.2	Methodology	71
5.2.1	Preliminaries	71
5.2.2	Collaborative Online Ranking	72
5.2.3	Collaborative Online Ranking with Large Margin	74
5.3	Experimental Results	76
5.3.1	Benchmark Setup	76
5.3.2	EachMovie Dataset	77
5.3.3	MovieLens Dataset	79
5.3.4	Multi-Domain Sentiment Dataset	80
5.4	Discussion	82
6	Conclusion and Future Work	85
6.1	Summary of the Thesis	85
6.2	Future Work	87
	Publications	90
	References	91

List of Figures

1.1	Illustration of collaborative learning by two distinct ways.	7
2.1	Illustration of the effect of unlabeled data in a transductive SVM. The dotted lines are decision boundaries obtained by only considering the labeled data. Taking unlabeled data (denoted by the solid circles) into consideration, the decision boundaries should be shifted from the dotted lines to avoid the dense regions, to the boundary denoted by the solid lines [132].	12
2.2	Five types of loss functions. For ramp loss (Figure 2.2.c) and concave loss (Figure 2.2.d), the parameter s is set to -0.3.	13
2.3	Example of unlabeled data affecting the decision boundary [15].	15
3.1	Decision boundaries (denoted by the solid line) obtained by the supervised SVM, CCCP TSVM, and Two-view TSVM. The only two labeled examples are represented by a bold cross and circle. The remaining points are unlabeled. Gaussian and linear kernels are used for views 1 and 2, respectively.	29
3.2	Illustration of the toy dataset.	36
3.3	Variation of the positive and negative classes' F1-measures over 100 splits of the WebKB course dataset. This figure shows CCCP TSVM, Two-view TSVM, and Laplacian SVM's variation on the positive and negative classes' F1-measures.	41
3.4	The five extracted features and their sample dictionary terms where applicable.	44
3.5	Accuracy versus the number of labeled examples for CCCP TSVM and Two-view TSVM on the product review dataset.	46

4.1	Leveraging data from a series of users via collaborative online multitask learning.	51
4.2	Illustration of the synthetic dataset. Two classes are represented each by crosses and circles. Solid lines denote the COML classification boundaries after observing all training samples.	59
4.3	Cumulative error rate (vertical axis) versus number of training samples observed (horizontal axis) on the synthetic dataset.	59
4.4	Cumulative error rate on the MHC-I dataset along the entire online learning process.	66
5.1	Experimental results of the EachMovie dataset.	79
5.2	Experimental results of the MovieLens dataset.	80
5.3	Cumulative error rate of the multi-domain sentiment dataset.	81
5.4	Cumulative rank loss rate of the multi-domain sentiment dataset.	82

List of Tables

3.1	Ads dataset result showing mean accuracy, F1-measure (%) and their standard deviation (in brackets).	38
3.2	WebKB course dataset result showing mean accuracy, F1-measure (%), and their standard deviation (in brackets).	40
3.3	Product review dataset result showing mean accuracy, F1-measure (%) and their standard deviation (in brackets).	45
4.1	Cumulative error rate (%) on the synthetic dataset.	58
4.2	Cumulative error rate and F1-measure (%) on the spam email dataset . .	61
4.3	Run-time (in seconds) for each algorithm	62
4.4	MHC-I dataset result showing average cumulative error rate, F1-measure (%) and their standard deviation (in brackets) of 12 tasks	63
4.5	Details of <i>Twitter</i> sentiment dataset	65
4.6	Cumulative error rate and F1-measure (%) on the <i>Twitter</i> sentiment dataset	68
5.1	Run-time (in seconds) for different algorithms	78

List of Abbreviations

AROW	Adaptive Regularization Of Weights
CCCP	ConCave Convex Procedure
COML	Collaborative Online Multitask Learning
COR	Collaborative Online Ranking
CORM	Collaborative Online Ranking with large Margin
CW	Confidence Weighted learning
DM	Data Mining
EM	Expectation Maximization
IR	Information Retrieval
KCCA	Kernel Canonical Correlation Analysis
LeToR	Learning To Rank
McRank	Multiple Classification Ranking
MHC	Major Histocompatibility Complex
MTL	MultiTask Learning
NLP	Natural Language Processing
NoPRank	Norm-Optimized Prank
OAP-BPM	Online Aggregate Prank-Bayes Point Machine
PA	Passive-Aggressive
PAC	Probably Approximately Correct
PRank	Perceptron Ranking
ROMMA	Relaxed Online Maximum Margin
RKHS	Reproducing Kernel Hilbert Space
SOP	Second Order Perceptron
SVM	Support Vector Machines
TF-IDF	Term Frequency-Inverse Document Frequency
TSVM	Transductive Support Vector Machine
UGC	User Generated Content

List of Notations

$\ \cdot \ _K$	Norm for Mercer kernel $K : X \rightarrow \mathbb{R}$
∇	Gradient
$L(\cdot)$	Hinge loss
$L_s(\cdot)$	Concave loss
$R_s(\cdot)$	Ramp loss function
$(\cdot)^T$	Matrix or vector transpose
$(\cdot)^{-1}$	Matrix or vector inverse
L	Graph Laplacian
W	Data adjacency graph
$f_\theta(x)$	SVM decision function
$\ \mathbf{x} - \mathbf{y}\ ^2$	Euclidean distance between $\mathbf{x}$ and $\mathbf{y}$

Chapter 1

Introduction

1.1 Background and Motivation

The popularity of Web 2.0, along with the increasing use of blogs, wikis, and social networking technologies has catalyzed the quantity and frequency of user generated content (UGC, also known as consumer generated media) for public consumption. The advent of UGC thus marked a paradigm shift among traditional media organizations; from on-line content creators to online facilitators who provide platforms for anyone to publish their content. Social media encourages anyone to publish his or her own content and/or comment on other UGC. As a result, UGC along with its associated applications have recently garnered significant attention from both academia and industry.

With the proliferation of UGC comes the demand for discovering previously unknown, and potentially useful information from it. However, while the benefits derived from analyzing UGC is great, the challenges of analyzing this new type of data cannot be met by simply applying traditional techniques in isolation from the underlying domain or context. To illustrate our point, we shall examine three emerging UGC applications and their corresponding challenges.

The first example stems from the popularity of product reviews that currently manifest itself in a variety of online formats. There are many sites dedicated to a specific type of product where numerous users submit their comments on consumer products. Informal reviews are also available on discussion boards as well as mailing list archives. Product review mining, first introduced by Dave et al. [55] deals with the machine analysis of opinion, sentiment, and subjectivity in online product reviews. It aims to extract

and summarize customer opinions on products from their reviews. Such opinions are helpful for both potential clients and product manufacturers since they contain valuable information about the user experience. However, due to the lack of quality control, online product reviews vary greatly in quality. Even popular sites dedicated to product reviews now contain large amount of spam reviews, such as advertisements, irrelevant topics, and posts with no opinions on the products, which can negatively affect the performance of a review mining system. Therefore, filtering out spam reviews efficiently and precisely is an indispensable preprocessing step for product review mining.

Spam review filtering can be generally treated as a classical text classification task. A model can be trained on a collection of labeled data to predict the labels (spam/non-spam) of new examples. However, in reality, labeled instances are often difficult, expensive, or time-consuming to obtain. How to filter out spam reviews using a classifier trained on only a limited number of training data was therefore the original motivation for our first topic. As will be shown in later chapters, we solve this problem with a novel multi-view semi-supervised learning algorithm that can take advantage of both the abundant amount of unlabeled data and their multiple representations to improve spam classification accuracy.

The second example deals with social media, in particular micro-blogs. Micro-blogs, such as *Twitter*¹ and *Facebook status updates*² allow users to publish short snippets of text, audio, image, and video from anywhere via the Internet, PCs, or mobile devices. The micro-blog posts can be made visible to anyone or just a restricted user group, as determined by the author. Compared to blogs, micro-blogs are typically much shorter in length, but updated much more frequently [79]. Micro-blogging has recently emerged as an important source of obtaining real-time news from the massive number of growing online users worldwide, and is gradually evolving into an informal real-time communication and news medium.

We are interested in the micro-blog sentiment detection problem, which aims to identify and detect if a user's micro-blog post contains emotions or sentiments. If one can accurately classify the emotional and non-emotional micro-blogs, it can have a variety

¹<http://www.twitter.com>

²<http://www.facebook.com>

of real life implications. For example, a user may want to filter out the emotional posts from acquaintances, but may wish to subscribe to every emotional post from his/her family members or close friends. Companies are also interested to monitor opinions toward certain products and services, e.g., budget hotel franchises gathering online customer feedback.

Micro-blog sentiment detection is however quite challenging on several fronts. One difficulty is that micro-blogs are often very short, thus making them hard to classify (even by human beings). Another concern is that micro-blogging services emphasize timely updates, which poses a strict requirement for developing a highly efficient and scalable classification solution. Finally, the emotional or non-emotional posts by different micro-blogging authors can be very diverse, which increases the difficulty of the classification task since traditional classification approaches based on a single global classifier trained from an aggregated collection of every user's data may fail to capture the peculiarity of different users and ultimately work poorly.

Collaborative online learning, the second algorithm presented in this thesis, was originally designed to solve the micro-blog sentiment detection problem, though it has potential applications beyond micro-blogging. The basic idea of collaborative online learning is to first build a generic global model from large amounts of user data, and then subsequently leveraging the global model to build the personalized online classification model of individual tasks in a collaborative learning manner. As shown later, our method can be applied effectively to the micro-blog sentiment detection problem, as well as to a number of classification problems dealing with data generated by multiple related tasks.

As the third example, consider the problem of recommending movies to multiple users by using the classical learning-to-rank method. The recommendation should rely on the user's own data, e.g., user profile, history of movie enjoyed in the past, etc. Yet it is also reasonable to take other similar users' preferences into consideration. Intuitively, individual traits and group characteristics could naturally be combined to achieve a result that is greater than the sum of its parts. When applied to multiple related ranking problems, existing learning-to-rank approaches based on a single ranking model (trained from multiple ranking problem data) may fail to capture the peculiarity of the individual problem. On the other hand, training each model per task using the task-specific data ignores the potential commonness shared among related ranking problems.

Collaborative online ranking, the third algorithm proposed in this thesis, aims to rank data generated by a group of tasks. Again, the idea is to combine group and individual characteristics by running several online ranking models in parallel for each related task. Each model corresponds to a specific task, and is updated by using each task's data and leveraging the information provided by models from fellow tasks. We formulate this idea into an optimization problem, enabling multiple tasks with varying amounts of data to collaborate and achieve overall better ranking performance in an online learning manner. As shown later, our online algorithms are not only effective, but also efficient and thus highly scalable to handle large amounts of real-time data.

1.2 Research Objectives

The objective of this research is to develop novel collaborative learning algorithms. We investigate how several machine learning models can be combined in either a data-centric or task-centric fashion to achieve a result that is greater than the sum of its parts.

From an application perspective, the proposed collaborative learning algorithms aim to improve user generated content analysis. For example, for online review spam filtering, we can improve the classification accuracy of existing methods, especially in the case where training data is scarce via our data-centric collaboration approach. Specifically, we address the following issues:

- (i) On the dearth of training data, how can a better classifier be built using multiple representations of the same problem. We answer this question with a novel multi-view semi-supervised learning algorithm, namely the two-view transductive Support Vector Machine (SVM).
- (ii) How can two-view data be generated for spam review detection. In particular, we define the two views for product reviews as (1) a classical text representation based on the word vector model, and (2) a high-level representation based on semantic analysis of each review sentence.

For the micro-blog sentiment detection problem and the movie recommendation problem, the objective is to improve existing online learning algorithms using a task-centric

collaborative approach. Specifically, we are interested in the following problem. Given a group of tasks, how can online classification/ranking combine both individual and group characteristics so that the group commonness and individual traits are well balanced, leading to overall improved classification/ranking results. As will be shown later, the proposed collaborative online learning algorithms can carry out this task successfully.

1.3 Research Scope and Limitations

We propose two orthogonal approaches for different machine learning classifiers to collaborate, either data-centric or task-centric. The data-centric approach uses the abundant amount of unlabeled data and their multiple representations to train several classifiers, enforcing collaboration by requiring them to maintain a maximum consensus on their predictions. The task-centric approach, in contrast, makes no special requirements on the representation of the input data. It instead assumes that several similar learning problems are presented at the same time. A classifier thus leverages information learned by other classifiers (on different tasks) to enhance the learning of itself.

Due to the fundamental differences between data-centric and task-centric collaborative learning, it may be difficult, if not impossible, to combine both into a coherent model. Thus one major limitation in the scope of our work is that we do not seek to combine them. Instead, we will study both models independently of one another.

1.3.1 Data Centric Collaborative Learning

For data-centric problems with lots of unlabeled data, we proposed a two-view transductive SVM. In fact, the proposed data-centric collaborative approach is closely related to the semi-supervised learning methodology, which uses large amount of unlabeled data together with the labeled data, to build classifiers. Since semi-supervised learning improves accuracy with little human effort, it is of great interest to data-mining academics and practitioners alike.

We evaluated the proposed two-view transductive SVM on both synthetic and real-life datasets against graph-based, single-view semi-supervised, and two-view supervised approaches. Specifically, we apply our two-view transductive SVM to an interesting

problem — product review filtering, which filters valid product reviews from online forum postings.

High complexity remains a major limitation of our two-view transductive SVM. This is partially due to the nature of transductive SVM — the objective function is non-convex, whose optimization procedure can be computationally demanding. Therefore, the two-view transductive SVM is too costly for large data sets (more than a few thousand samples) in practice.

1.3.2 Task Centric Collaborative Learning

For task-centric problems, we proposed a collaborative online classification/ranking approach to classify/rank data generated by a group of tasks by combining group and individual task characteristics. The proposed algorithms belong to the online (incremental) learning family, which updates learning models incrementally to make predictions on a stream of samples that arrive sequentially. Since online learning enjoys a lot of advantages including high efficiency and scalability, it has been actively studied in the machine learning community.

We showed that learning multiple related tasks collaboratively outperforms learning each task independently by comparing our algorithms with state-of-the-art online learners on real-life problems. We also compared the run-time of different algorithms, to show that ours are not only effective, but also equally fast as other online learners.

1.4 Approach and Methodology

The general approach we adopted to solve the problems stated above is called collaborative learning. Collaborative learning is a process in which two or more models are applied on to the same problem, each learning its own set of parameters in cooperation with another model. We demonstrate two distinct ways to train classifiers collaboratively in this thesis, one data-centric, and one task-centric, as shown in Figure 1.1. Unlike traditional ensemble methods, which perform classification by taking a majority vote on the predictions made by each base classifier, the two novel techniques (data-centric and task-centric) shown in this thesis are formulated into optimization problems and solved efficiently. In the following, we give a brief description of each approach.

Figure 1.1: Illustration of collaborative learning by two distinct ways.

1.4.1 Two-view Transductive SVM

Classification, the task of assigning objects to one of several predefined categories, is an active research problem in the data mining and machine learning community [108]. The classical classifier is created by building a machine learning model, e.g., support vector machine (SVM) [81], [119], trained from a collection of labeled data. Unfortunately, in real-world applications, labeled training examples are often difficult to obtain, as they require the effort of human annotators, while unlabeled data are always abundant. We attempt to overcome this limitation with a semi-supervised learning approach, which aims to improve the performance of a classifier trained on a limited number of labeled data by utilizing the unlabeled data. Among various semi-supervised learning algorithms, the transductive support vector machine (TSVM) has drawn a lot of attention since it was first introduced by Vapnik [119]. An intuitive interpretation for the success of transductive SVM is the so-called “cluster assumption” [38]. That is, instead of traversing through high density regions of the data, the decision boundary should always be placed in low density regions. One can implement this assumption into the SVM optimization procedure by exploiting the information of unlabeled data.

To improve the performance of the transductive SVM, we adopt a multi-view collaborative learning approach. In multi-view collaborative learning, a classifier is created for each representation or view of the same problem, with each classifier optimized to maximize the overall consensus in their predictions. Where multiple representations of the same problem are available, a multi-view collaborative learning approach typically

yields equal or better results than those obtained from either view alone. Our proposed two-view semi-supervised learning algorithm, called two-view transductive SVM, extends the supervised two-view learning framework of Farquhar et al. [64] to take advantage of the abundance of unlabeled data.

Venturing a step further, we also apply manifold regularization, which is a kind of graph-based semi-supervised learning. Manifold regularization [15, 16, 113] extends existing supervised learning algorithms to support semi-supervised learning by adding a geometrically based regularization term with the aim of preserving the manifold smoothness. By incorporating a regularizer that records the intrinsic manifold structure of training data, we finally obtain a joint learning framework that integrates transductive SVM, graph-based semi-supervised learning, and multi-view learning. We formulate this learning framework into an optimization problem and develop an efficient way to solve it.

1.4.2 Collaborative Online Learning

Collaborative online learning is a highly efficient and scalable online classification framework that is designed to classify micro-blog posts generated by a group of users by combining group and individual characteristics. We opt for the online learning framework, which incrementally learns a classification model from a stream of examples. The motivation of our choice is two-fold. First, online learning enjoys many advantages, chief among which counts its high efficiency and scalability. In fact, scalability is an essential requirement for the real-time classification of hundreds of thousands of online users at the same time. Second, a user's micro-blog is always changing; he could be writing about different topics at various time of the day, week, or month. Any personalized classifier will have to be able to quickly adapt to the latest trends at both the individual and group level. Whenever a post is published, it needs to be classified right away and the model updated immediately, simply leaving no room for periodic retraining that is typical of batch algorithms.

The basic idea of collaborative online learning is to first build a generic *global* model from large amounts of user data, and then subsequently leveraging the global model

to build the *personalized* online classification model of individual tasks in a *collaborative* learning manner. We evaluate the empirical performance of our algorithm for micro-blogging sentiment detection tasks based on a dataset crawled from *Twitter*. The encouraging results validate the effectiveness of our method.

1.4.3 Collaborative Online Ranking

There are many applications where it is desirable to rank or order instances instead of classifying them. Ranking resembles classification as both assign one of several possible labels to a new instance, but differs in that there is an ordering among the labels. We thus study the problem of learning-to-rank instances belonging to multiple related tasks and proposed the multitask collaborative online ranking algorithm. The basic idea is to run N online ranking models in parallel for N related tasks. Each model corresponds to a specific task, and is updated by using the task's own data and leveraging the information provided by models of fellow tasks. We formulate this idea into an optimization problem, enabling multiple tasks with varying amounts of data to collaborate and achieve overall better ranking performance in an online learning manner.

1.5 Main Contributions

The contributions of this research can be summarized as follows:

- (i) We devised a systematic methodology to collect two-view data from online product review forums for spam review detection.
- (ii) We extended the existing two-view supervised learning algorithm to a semi-supervised setting, and created a new multi-view semi-supervised learning algorithm, namely the two-view transductive SVM, which takes advantage of both the abundant amount of unlabeled data and their multiple representations to improve classification performance.
- (iii) We incorporated manifold regularization, which is a type of graph-based semi-supervised learning into our two-view transductive SVM framework. Experimental results indicate that incorporating additional manifold information can also help to improve the classification results.

- (iv) We studied the multitask learning problem under the online learning setting, and presented several collaborative online learning algorithms for classifying or ranking data generated by a group of different but related tasks. The proposed collaborative online ranking algorithm is the first attempt at exploring ranking problems in an online multitask learning setting.
- (v) We investigated a new micro-blogging data mining problem, “micro-blogging sentiment detection”, which aims to identify and classify emotional micro-blogging posts.
- (vi) We evaluated the empirical performances of our algorithms on various datasets. The encouraging results validated the effectiveness of our methods.

1.6 Organization of the Thesis

This rest of this thesis is organized as follows. Chapter 2 summarizes related work. Chapter 3 presents our two-view transductive SVM algorithm. Chapter 4 describes our collaborative online learning framework. Chapter 5 shows our collaborative online ranking framework. Finally, Chapter 6 concludes the thesis and discusses future directions.

Chapter 2

Literature Survey

In this chapter, we conduct a comprehensive literature survey on various topics related to this research. We first review existing work on semi-supervised learning, focusing on transductive support vector machines and graph-based methods, followed by multi-view learning algorithms. Since our two-view transductive SVM will be used on product review filtering task, we also give a brief survey on review mining. Finally, we review related work on online learning, multitask learning, and learning-to-rank.

2.1 Semi-supervised Learning

Semi-supervised learning, i.e., learning from both labeled and unlabeled data, has been well studied for the past twenty years. Popular approaches include co-training, transductive support vector machines, and graph-based methods, etc. We first review transductive support vector machines and graph-based methods, followed by co-training and multi-view learning.

2.1.1 Transductive SVM

Support Vector Machines are techniques for solving certain class of problems by learning from examples, in particular, approximating a multivariate function from sparse data. This problem is ill-posed and the classical solution involves regularization theory [18] [117]. The problem can be formulated as finding a function that minimizes an objective function, which comprises an empirical cost function and a complexity measure in an appropriately chosen Reproducing Kernel Hilbert Space (RKHS).

Figure 2.1: Illustration of the effect of unlabeled data in a transductive SVM. The dotted lines are decision boundaries obtained by only considering the labeled data. Taking unlabeled data (denoted by the solid circles) into consideration, the decision boundaries should be shifted from the dotted lines to avoid the dense regions, to the boundary denoted by the solid lines [132].

To be more specific, for a Mercer kernel $K : X \rightarrow \mathbb{R}$, there is an associated RKHS $\mathcal{H}_K$ of functions $X \rightarrow \mathbb{R}$ with the corresponding norm $\|\cdot\|_K$. Given a set of ℓ labeled examples $\{(\mathbf{x}_i, y_i)\}_{i=1}^{\ell}$, $\mathbf{x}_i \in \mathbb{R}^n$, $y_i \in \{-1, 1\}$, the standard SVM estimates a function f^* by minimizing

$$f^* = \operatorname{argmin}_{f \in \mathcal{H}} \sum_{i=1}^{\ell} L(y_i f(\mathbf{x}_i)) + \lambda \|f\|_K^2 \quad (2.1)$$

where λ is the regularization parameter, and $L(\cdot) = \max(0, 1 - \cdot)$ is a loss function.

The classical Representer Theorem states that the solution to problem (2.1) exists and can be written as

$$f = \sum_{i=1}^{\ell} \alpha_i K(\mathbf{x}, \mathbf{x}_i) \quad (2.2)$$

The transductive SVM can be viewed as a standard SVM with an extra regularization term defined over the set of unlabeled data [132]. Intuitively, unlabeled data helps to guide the decision boundary away from high density regions as shown in Figure 2.1.

Suppose a training set contains ℓ labeled examples $\{(\mathbf{x}_i, y_i)\}_{i=1}^{\ell}$, $\mathbf{x}_i \in \mathbb{R}^n$, $y_i \in \{-1, 1\}$, and u unlabeled examples $\{\mathbf{x}_i\}_{i=\ell+1}^{\ell+u}$. The SVM decision function has the form

$$f_{\theta}(\mathbf{x}) = \mathbf{w} \cdot \Phi(\mathbf{x}) + b \quad (2.3)$$

where $\theta = (\mathbf{w}, b)$, $\mathbf{w} \in \mathbb{R}^n$, $b \in \mathbb{R}$ are the parameters of the model, and $\Phi(\cdot)$ is the feature map. The transductive SVM adds an additional regularizer defined over unlabeled data

Figure 2.2: Five types of loss functions. For ramp loss (Figure 2.2.c) and concave loss (Figure 2.2.d), the parameter s is set to -0.3.

to the classical SVM optimization function. To be specific, the transductive SVM solves the following optimization problem

$$\min \frac{1}{2} \|\mathbf{w}\|^2 + C \sum_{i=1}^{\ell} L(y_i f_{\theta}(\mathbf{x}_i)) + C^* \sum_{i=\ell+1}^{\ell+u} L(|f_{\theta}(\mathbf{x}_i)|) \quad (2.4)$$

where $L(\cdot) = \max(0, 1 - \cdot)$ is the classical hinge loss for labeled examples as illustrated in Figure 2.2.a, $L(|\cdot|) = \max(0, 1 - |\cdot|)$ is the symmetric hinge loss for unlabeled examples as illustrated in Figure 2.2.b, C and C^* are adjustable parameters.

Problem (2.4) is hard to optimize since the objective function is non-convex. To solve this problem, a suite of algorithms have been proposed. Chapelle et al. [38] proposed ∇ SVM, which approximates the symmetric hinge loss (Figure 2.2.b) with a smoothed version — $S(t) = \exp(-3t^2)$ (Figure 2.2.e), and subsequently directly minimizes the objective function (2.4) by gradient descent. Collobert et al. [43] employed an approximate optimization technique known as the concave convex procedure (CCCP) [124]. It decomposes a non-convex function f into a convex part f_{vec} and a concave part f_{cave} , which are

then solved iteratively. In each iteration, the concave part is replaced by its tangential approximation. Then, the sum of the convex part and the tangential approximation is minimized. The CCCP pseudo code is shown in Algorithm 1.

Algorithm 1: Minimizing a non-convex function with CCCP.

Require: $f = f_{vec} + f_{cave}$, starting point $\mathbf{x}_0$.
 $t = 0$
while $\nabla f(\mathbf{x}_t) \neq 0$ **do**
 $\mathbf{x}_{t+1} = \operatorname{argmin}_{\mathbf{x}} f_{vec}(\mathbf{x}_t) + \nabla f_{cave}(\mathbf{x}_t) \cdot \mathbf{x}_t$
 $t = t + 1$
end while

For CCCP transductive SVM [43], the loss function applied to unlabeled data is called “ramp loss” (Figure 2.2.c), which can be expressed as the sum of a hinge loss function (Figure 2.2.a) and a concave loss function (Figure 2.2.d). Specifically, the ramp loss function $R_s(\cdot)$ has the form

$$R_s(\cdot) = \min(1 - s, \max(0, 1 - \cdot)) = L(\cdot) + L_s(\cdot) \quad (2.5)$$

where L is the hinge loss, L_s is the concave loss with the form $L_s(\cdot) = -\max(0, s - \cdot)$, and s is a predefined parameter such that $-1 < s \leq 0$.

According to Collobert et al. [43], training a transductive SVM with the CCCP method is equivalent to training a SVM using the hinge loss for labeled data, and the ramp loss for unlabeled data. For a binary classification problem, each unlabeled example is accounted for twice, each time assuming the role of one class, that is, $\{(\mathbf{x}_i, y_i = 1)\}_{i=\ell+1}^{\ell+u}$, $\{(\mathbf{x}_i, y_i = -1) : \mathbf{x}_i = \mathbf{x}_{i-u}\}_{i=\ell+u+1}^{\ell+2u}$. The corresponding optimization problem of CCCP transductive SVM is given by

$$\min \frac{1}{2} \|\mathbf{w}\|^2 + C \sum_{i=1}^{\ell} L(y_i f_{\theta}(\mathbf{x}_i)) + C^* \sum_{i=\ell+1}^{\ell+2u} R_s(y_i f_{\theta}(\mathbf{x}_i)) \quad (2.6)$$

2.1.2 Graph-based Methods

Graph-based semi-supervised learning methods are based on a label smoothness assumption, which states that similar examples should be assigned the same class labels. Figure 2.3 shows how the smoothness assumption can completely alter the appropriate choice

of classification functions. Graph-based methods first define a graph where labeled and unlabeled data are represented as vertices, with edge weights encoding the similarity between examples. It then estimates a function over the graph, while simultaneously satisfying two conditions [132]: (1) the function should yield assignments similar to the labeled nodes, and (2) it should be smooth throughout the graph. These two conditions can be expressed quantitatively in an optimization framework using a loss function and a regularizer derived from data features.

Figure 2.3: Example of unlabeled data affecting the decision boundary [15].

Thus, the existing suite of graph-based methods differ from one another largely in the choice of loss functions and regularizers. Blum et al. [23] considered semi-supervised learning as a graph min-cut problem. Their algorithm uses a similarity measure between data to construct a graph. It then generates a classification that partitions the graph such that the number of similar pairwise examples that are assigned different labels is minimized. Gaussian random fields and harmonic function methods [133] introduce a quadratic loss function with infinity weight for labeled data, and incorporate unlabeled data with a regularizer based on the graph combinatorial Laplacian. Zhou et al. [129] proposed the local and global consistency method with a quadratic loss function and the normalized Laplacian as the regularizer. Belkin et al. [15, 14] developed the generative manifold regularization framework, which exploits the geometry of the data probability distribution and incorporates it as an additional regularization term. The probability distribution is supported on a sub-manifold of the ambient space.

To be specific, suppose labeled examples $(\mathbf{x}, y)$ are drawn from a probability distribution P on $X \times \mathbb{R}$, and unlabeled examples $\mathbf{x} \in X$ are drawn from the marginal distribution $\mathcal{P}_X$ of P . The manifold regularization framework assumes that if two points

$\mathbf{x}_1, \mathbf{x}_2 \in X$ are close in the intrinsic geometry of $\mathcal{P}_X$, then their conditional distributions $\mathcal{P}(y|\mathbf{x}_1)$ and $\mathcal{P}(y|\mathbf{x}_2)$ are similar.

To incorporate additional information about the geometric structure of marginal distribution $\mathcal{P}_X$, an additional regularizer is added to the original SVM optimization problem (2.1). This leads to the following optimization problem for manifold regularization

$$\min \frac{1}{\ell} \sum_{i=1}^{\ell} L(y_i f(\mathbf{x}_i)) + \lambda \|f\|_K^2 + \lambda_I \|f\|_I^2 \quad (2.7)$$

where $\|f\|_I^2$ is a regularization term induced by the labeled and unlabeled data. It reflects the intrinsic structure of $\mathcal{P}_X$, and can be approximated on the basis of labeled and unlabeled data using the graph Laplacian [14] [41]. That is

$$\|f\|_I^2 = \mathbf{f}^T L \mathbf{f} \quad (2.8)$$

where $\mathbf{f}$ is the vector of f evaluation on the labeled and unlabeled data, given by $\mathbf{f} = [f(\mathbf{x}_1), \dots, f(\mathbf{x}_{\ell+u})]^T$, and L is the graph Laplacian given by $L = D - W$ where W_{ij} are the edge weights in the data adjacency graph. The diagonal matrix D is given by $D_{ii} = \sum_{j=1}^{\ell+u} W_{ij}$.

2.2 Co-training and Multi-view Learning

2.2.1 Co-training

Blum et al. [24] came up with the co-training algorithm, which bootstraps two classifiers defined on two sets of feature by training both with high confidence data labeled by its peer classifier. Co-training works under a two-view setting, which assumes that two feature sets are given, where each set is sufficient for learning and is conditionally independent of the other given the class label. At the beginning, two separate classifiers are trained with the labeled data of the two feature sets. Each classifier then classifies the unlabeled data, and “teaches” the other classifier those predicted labels they feel most confident about. In co-training, unlabeled data helps by reducing the version space size. In other words, the two classifiers must agree on the much larger unlabeled data as well as on the labeled data.

Nigam et al. [96] showed that co-training performs well if the conditional independence assumption holds. With a PAC (Probably Approximately Correct) style theoretical analysis, Dasgupta et al. [54] proved that the co-trained classifiers could make fewer generalization errors by maximizing their agreement over the unlabeled data. There are also many variants and applications of co-training, including [5, 68, 130, 131], which incorporate modifications to the basic principle.

2.2.2 Multi-view Learning

Multi-view learning utilizes the agreement among learners trained on different representations of the same problem to improve the overall classification result. The basic idea of using two views with unlabeled data was first introduced by Sa [58]. Sindhwani et al. [112] defined a co-regularization approach that learns a multi-view classifier from partially labeled data using a view consensus based regularization term. In particular, we focus on the work of Farquhar et al. [64]. They observed that when two views of the same problem are available, applying the kernel canonical correlation analysis (KCCA) [70] to the two feature space can improve the performance of the classifier. They also proposed a supervised learning algorithm named SVM-2K, which imposes a similarity constraint between two distinct SVMs, each trained from one view of the data. The constraint they employed is

$$|f_{\theta}^A(\mathbf{x}_i^A) - f_{\theta}^B(\mathbf{x}_i^B)| \leq \eta_i + \varepsilon \quad (2.9)$$

where $f_{\theta}^{A/B}(\cdot)$ are the SVM decision functions belonging to each of the two views denoted by superscripts A and B , respectively, η_i is a variable that imposes consensus between the two views, and ε is a slack variable for allowing some samples to violate the constraint. Combining this constraint with the standard SVM objective functions for each view yields a multi-view learning algorithm, which was shown to perform better than the single view approach on an image classification task.

Not restricting to labeled data, Szedmák et al. [115] went further by exploiting unlabeled sources of data via multi-view learning. They require the two classifiers to give similar solutions on the unlabeled samples. The similarity is measured by the absolute value of differences between two real-valued predictions of the unlabeled data, and is

minimized simultaneously with the error occurring in the estimation of the labeled cases. The optimization problem of their learning framework is

$$\min_{\theta^{A/B}, \xi^{A/B}, \eta} \quad \frac{1}{2} \|\mathbf{w}^A\|^2 + \frac{1}{2} \|\mathbf{w}^B\|^2 + C^A \sum_{i=1}^{\ell} \xi_i^A + C^B \sum_{i=1}^{\ell} \xi_i^B + D \sum_{i=\ell+1}^u \eta_i \quad (2.10a)$$

$$\text{s.t.} \quad y_i f_{\theta}^{A/B}(\mathbf{x}_i^{A/B}) \geq 1 - \xi_i^{A/B} \quad \forall 1 \leq i \leq \ell \quad (2.10b)$$

$$\xi_i^{A/B} \geq 0 \quad \forall 1 \leq i \leq \ell \quad (2.10c)$$

$$|f_{\theta}^A(\mathbf{x}_i^A) - f_{\theta}^B(\mathbf{x}_i^B)| \leq \eta_i \quad \forall \ell + 1 \leq i \leq \ell + u \quad (2.10d)$$

$$\eta_i \geq 0 \quad \forall \ell + 1 \leq i \leq \ell + u \quad (2.10e)$$

where (2.10b) and (2.10c) are standard SVM constraints defined over labeled data, (2.10d) and (2.10e) impose the classifiers to give similar solutions on the unlabeled data. It should be noted that in the above learning framework, the loss function is defined only on the labeled data. In contrast, our proposed method also contains loss function defined over unlabeled data. The difference between the work of Szedmák et al. [115] and ours is that our method finds a labeling of the unlabeled data, so that a decision boundary has the maximum margin on both the original labeled data and the (newly labeled) unlabeled data.

2.3 Product Review Filtering

We give a brief review on the product review filtering problem, which is one of the main application domains for our algorithm. Both classification and regression based methods have been applied to rate product reviews. Liu et al. [89] detected low-quality product reviews (spam review) with a binary classifier trained from manually annotated data. Some researchers used regression models to predict the utility scores of product reviews crawled from the Amazon website [80, 94, 126]. A multitude of semantic features, term statistics, and metadata of review sentence were extracted to build the regression model. Since the ground-truth of review quality is difficult to obtain, humans were asked to vote on each review's helpfulness, which were subsequently used to train the ranking models. Jindal et al. [80] solved the problem in a different way: they first tried to recognize duplicate reviews, which are then treated as spam reviews for training. Their

approach was based on the intuition that a large number of duplicate reviews constitutes a representative set of spam reviews. But as Pang et al. [99] stated, the assumption that duplicate reviews constitute some sort of manipulation attempt is weakened by the fact that duplicate reviews in Amazon can be largely attributed to Amazon’s own cross-posting mechanism. We thus propose that in the absence of user ratings (ground truth), a semi-supervised learning method like our two-view transductive SVM is a better way to tackle the problem.

2.4 Sentiment Analysis

The solutions studied in this thesis was originally motivated by problems in sentiment analysis and opinion mining. Sentiment analysis determines if a text snippet contains subjective information or not. In the presence of subjective content, it further determines the sentiment polarity (positive or negative opinion) of the opinionated text with respect to a topic or context. Given the enormous amount of data published in Web, sentiment analysis has recently become a field of interest for many areas of research ranging from computer science to online social marketing.

Current research in sentiment analysis typically treat the task as a two-stage process. The first stage distinguishes subjective instances from objective instances. The second stage determines the sentiment polarity of opinionated texts that were recognized in the first stage. Identifying subjective instances has often proved to be more difficult than the subsequent task polarity classification [99]. An alternative way is to perform subjectivity detection and sentiment polarity identification at the same time, yielding a result that classifies a given text as objective (lack of opinion), positive (expressing a positive opinion), or negative (expressing a negative opinion). Whatever the approach, sentiment analysis can be postulated as a classification problem where the task is to assign text instances to one of several pre-defined categories. Text classification using machine learning techniques is a well studied field and has been widely used to solve sentiment analysis problems. It has been shown that machine learning techniques achieve better results on classifying movie review sentiments compared to methods based on hand-picked rules [100].

To solve classification problems with machine learning techniques, a training set consisting of samples whose sentiment class labels are known must be provided. Each sample in the training set can be a microblog post, whose class label can be positive, negative, or neutral. The training set comprising samples of attributes and associated class labels is used to build a classification model, which is subsequently applied to records with unknown class labels. A learning algorithm is employed to build a classification model that estimates the relationship between the attributes and the class label of the training data. The model generated by a learning algorithm should both fit the input data well and correctly predict the class labels of records it has never seen before. Examples of popular learning algorithms include decision trees, neural networks, naïve Bayes classifiers, and support vector machines (SVM).

A human labeled set of training data is typically necessary for training a classifier. Generally, the accuracy of the classifier increases with increasing amount of labeled data. Take the micro-blog sentiment detection problem as an example, it is easy to collect millions of tweets with the Twitter API¹. However, manually labeling tweets as positive or negative is a laborious and expensive endeavour. Researchers have investigated various ways to automatically generate labeled training data for Twitter microblog sentiment analysis. For example, several researchers rely on emoticons to automatically label training data [20, 98]. An emoticon is a pictorial representation of a facial expression using the characters available on the standard keyboard, usually written to express a person's feelings or mood. Emoticons in tweet are good indicators of a writer's emotional state, and thus can be used to annotate training data accurately. [98] queried Twitter for two types of emoticons: happy emoticons and sad emoticons. The collected tweets were used to train a classifier to recognize positive and negative sentiments. In order to collect a corpus of objective posts, the author retrieved tweets from Twitter accounts of popular newspapers and magazines. Twitter uses hashtags — words or phrases prefixed with a “#” sign to denote subjects or categories. Some of these tags are sentiment tags, which assign one or more sentiment values to a tweet. [57] made use of the *#sarcasm* tag to automatically collect sarcastic tweets [56, 84] manually selected a set of frequent hashtags

¹<https://dev.twitter.com>

that are indicative of positive, negative, and neutral tweets. These hashtags were used to create the required training data.

Apart from training set size, the accuracy of the learned classifier also depends strongly on how the input sample is represented. Typically, the input sample is transformed into a feature vector, which contains a number of features that are descriptive of the sample. In classical data mining, this is known as the feature selection problem. A variety of features have been studied for Twitter sentiment classification, including unigrams, bigrams, part-of-speech (POS) tags, punctuation based features, etc. A natural choice would be to use the word based n -gram features. Each word appearing in a sentence or each consecutive word sequence containing n words serves as a binary feature. Weighting techniques have been applied to assign higher weights to rare words [56]. [10] mapped each word in a tweet to its POS using a POS dictionary. For each tweet, [84] treated the number of verbs, adverbs, adjectives, nouns, etc., as POS features. The intuition is that certain POS tags are good indicators of sentiment. Generic features such as sentence length, number of special punctuations, presence of emoticons and abbreviations have also been used [56, 84]. Lexicon features have also been investigated. They map a word to its prior subjectivity (weak or strong subjectivity), and polarity (positive, negative, or neutral) according to a lexicon [10, 84].

2.5 Online Learning

Online learning has been extensively studied [35, 47, 52, 107, 122, 127, 67, 88]. It aims to make a decision about the present based on knowledge of the past [22] using the latest available information. Unlike batch learning methods, which assume all training samples to be available before the learning task begins, online learning algorithms incrementally build a model from a stream of samples, allowing for simple and fast model updates. They are thus naturally capable of dealing with large datasets and problems for which data arrive sequentially.

Online learning algorithms make a trade-off between the two factors: on one hand, the classifier should be updated whenever it misclassifies a new instance; on the other hand, the classifier shouldn't be changed too rapidly especially if it correctly predicts most of

the previous instances. The well-known Perceptron algorithm [107] used a simple linear combination of the current hyperplane and the current instance vector to cope with these two requirements. The classifier is updated by moving the hyperplane closer to the misclassified sample. That is, adding a new example with a constant weight into the current set of support vectors when it is misclassified. Although the algorithm uses a simple update rule, it performs well in practice. The work in [83] enumerates numerous similar additive and multiplicative online algorithms for regression. To minimize the sum of distances between the new classifier and the current classifier and the loss (if any) on the current example, a gradient-descent method is adopted as the update rule. Grove et al. [69] introduced a general framework of quasi-additive binary algorithms, which include the Perceptron as a special case. Gentile [67] proposed an extension to a subset of the quasi-additive algorithms, which uses an additive conservative update rule with decreasing learning rates.

To overcome the limitation of the simple Perceptron-like methods, many modern online learning algorithms [67, 88, 52, 47] have been proposed over the past decade. These algorithms are inspired to some extent by the maximum margin learning principle that has been successfully applied to batch mode learning. Specifically, the Relaxed Online Maximum Margin (ROMMA) algorithm [88] repeatedly chooses a hyper-plane that correctly classifies existing training examples with the maximum margin. Another representative example is the Passive-Aggressive (PA) method [47]. The family of PA algorithms [47] maintains a trade-off between the amount of progress made on each training round and information retained from previous rounds. In particular, the PA algorithm updates the classification model whenever a new example is misclassified or when its classification score is smaller than some predefined margin. Empirical studies showed that the maximum margin based online learning algorithms are generally more effective than the Perceptron algorithm. However, despite the difference, most online learning algorithms only update the weight of the newly added support vector, and keep the weights of the existing support vectors unchanged.

The above online learning algorithms in general belong to the family of first-order online learning techniques. Recent years have witnessed emerging studies on exploring second-order information for online learning. The second-order online learning algorithms

that improve upon the Perceptron-like methods include the second-order Perceptron (SOP) [36], confidence weighted (CW) learning [60] and its successors [49, 48, 50, 95]. The confidence weighted learning [60] algorithm maintains a probabilistic measure of confidence in each component of its weight vector using a Gaussian distribution. The weight distribution is updated by minimizing the Kullback-Leibler (KL) divergence between the new weight distribution and the old one under the constraint that the probability of correct classification is greater than a threshold. The learner makes online updates based on its confidence in the current parameters, making larger changes in the weights of infrequently observed features. Empirical evaluation has demonstrated the advantages of this approach for a number of natural language processing tasks.

AROW [50], which stands for “adaptive regularization of weight vectors”, softens the hard constraint in confidence weighted learning as regularizers. It also employs a more sophisticated update strategy, leading to extra robustness in the case of non-separable data. AROW is also similar to the second order Perceptron (SOP) [36]. The SOP performs the same type of update as AROW, but only when it makes an error. AROW, on the other hand, updates even when its prediction is correct if there is insufficient margin. Confidence weighted (CW) algorithms update the mean and confidence parameters simultaneously, while AROW makes a decoupled update and softens the hard constraint of CW. The AROW algorithm can also be viewed as a variant of the PA-II algorithm [47] where the regularization is modified according to the data. Empirically, AROW has been demonstrated to have state-of-the-art performance for common online learning tasks.

2.6 Multitask Learning

The problem of jointly solving several related learning tasks by leveraging the commonality among tasks has been studied in the machine learning community under the guise of multitask learning. There has been an active line of research in multitask learning [7, 9, 17, 61, 25, 111, 4]. The pioneering work [33] assumes there is one primary task and other secondary tasks, also known as extra tasks, which are solely used to improve the learning of the primary task. Task relationships are modeled on the assumption that all functions learned are close to each other according to some norm [9, 61, 93, 123].

For example, the differences in the parameters used by all tasks are measured via the Frobenius norms [62, 90, 125, 101]. As an intuitive and principled tool for learning from high-dimensional data, regularization can impose various prior knowledge on the problem. Evgeniou et al. [61] introduced the multitask kernel and considered batch multitask learning as a regularized optimization problem. Argyriou [6] used a trace-norm based regularizer to minimize the empirical risk of all tasks on the parameter matrix, using the parameters of all tasks.

An intuitive way to model task relatedness is to consider the sharing of relevant information across the multiple related tasks. Ando et al. [4] formulated multitask relations by enforcing the predictive functions from different tasks to belong to the same hypothesis set. [123, 86, 76] assume that the parameters used by all tasks share a common prior [123, 86, 76]. Similarly, the work presented in [34] assumes a fixed task relationship matrix. [13] considered the problem of choosing an optimal hypothesis space from a family of hypothesis spaces for bias learning. [3, 8] consider a hierarchical Bayes model, where the parameters used by all tasks are assumed to be sampled from a Gaussian prior, and the iterative Gibbs sampling method are adopted to estimate both the individual functions and the shared Gaussian parameters. [9] suggests a similar hierarchical model by assuming a mixture of Gaussian models instead of a single Gaussian.

The other way to model task relatedness is to assume that all tasks share a common yet latent feature representation [33, 13, 17, 4]. For example, for object recognition, the objects in the human visual system are represented using a common set of features learned, e.g., local filters resembling wavelets [109]. Argyriou et al. [6] proposed a framework to learn shared features using convex optimization. Kang et al. [82] studied the problem of multitask learning of shared feature representations among tasks, while simultaneously determining “with whom” each task should share. Some other studies tried to explore the underlying spectral dependencies among tasks [7, 6, 77]. In [6], the problem of inferring task clusters is cast as learning a set of kernel functions, one for each group. Jacob et al. [77] consider the problem of automatically clustering tasks, where tasks in the same cluster are more related to each other than to tasks in a different cluster. In their work, the task relatedness is modeled by their parameter similarity. Dissimilar tasks are then placed in different clusters and learnt independently from each other. Similar

ideas were also explored in an earlier work [116], where the similarity between two tasks is quantified by a fitness measure: how well one task’s parameters perform in the other task.

Unlike the existing batch multitask learning studies, our work is closer to the on-line multitask learning methodology. The online multitask learning problem was first addressed in [59], who assumed a very general setting wherein the tasks were related by a global loss function and the goal was to reduce the cumulative loss (for all tasks involved) over all rounds of the online algorithm. On a similar but somewhat different note, [1] and [2] considered an alternative formulation of online multitask learning under the traditional expert advice model. In their regret-minimization framework, the notion of task relatedness was captured in terms of experts with the hope that experts that perform well on one task should also do well on other related tasks. The goal was to find a small subset of experts that perform well throughout the learning process. This, in a way, is analogous to finding a low-dimensional common representation for the multiple related tasks [61, 104]. Following the same line of thought, [92] extended the notion of experts to the set of decisions the forecaster is allowed to take. Like earlier approaches, the idea is to impose task relatedness by constraining the different tasks to choose their decisions from a small subset. Apart from minimizing the cumulative loss and regrets, reducing mistake bounds for the online multitask learning has also been considered in [34].

Instead of explicitly modeling the specific task relatedness measure, our technique jointly learns a generic global model shared by many parallel learners. Individual collaborative models are required to learn from one another. Our approach not only enjoys a considerable improvement in classification performance, but also retains the hallmark low computational cost of online learning algorithms.

2.7 Learning to Rank

Ranking or ordering objects according to their degree of relevance, preference, or importance is a key problem for information retrieval and many data mining applications.

Applying machine learning technologies to solve ranking problems has been studied extensively under the name “learning-to-rank”. Specifically, learning-to-rank methods automatically construct a ranking model from training data, which comprises a set of input objects and their corresponding ranks, such that the learned model can sort unseen objects in a reasonable manner. According to [91], learning-to-rank algorithms can be grouped into three categories of approaches, namely, point-wise, pair-wise, and list-wise. We review representative work from each category as follows.

The point-wise approach transforms the ranking problem (ranking model creation) into a classification, regression, or ordinal classification problem, ignoring the group ranking structure. Representatives of this category include PRank [51], OAP-BPM [71], Subset Ranking with Regression [46], Ranking with Large Margin Principles [110], Constraint Ordinal Regression [40], McRank [87], etc.

One classical point-wise approach is PRank (Perceptron Rank) [51, 53], which projects each instance onto the real line. The real line is thus divided up into sub-intervals, which forms a partial order that is also known as the interval order. From the interval order, instances and their rankings can be easily computed. It has been shown that the number of the updates of the PRank algorithm is bounded. However, there is no guarantee on the size of PRank’s margin. Online Aggregate PRank-Bayes Point Machine (OAP-BPM) [71] is a modification to PRank that bounds the margins.

Ordinal regression based methods map ordinal scales into numeric values, and then formulate ranking into a standard regression problem. As an example, the method proposed in [46] ranks web pages by treating the relevance between the web page and query as a real number. The relevance function is learned by regression. Batch mode SVM algorithms have been used to tackle the ordinal regression problem [40, 106, 110].

The ranking problem has also been cast as a multiclass classification problem. The Multiple Classification Ranking (McRank) algorithm [87] is one of the representative approaches. In order to convert classification results to ranking scores, the output of the classifier is translated into probability via a logistic function. The probability denotes how likely a document belongs to a specific category. A boosting tree algorithm for multiple classification is implemented to learn the class probabilities. The difference between

ordinal regression and classification is that ordinal regression takes the ordinal relationship among the ground truth labels into consideration. In addition, ordinal regression predicts rank by thresholding a continuous score.

In the pair-wise approach, ranking is transformed into a pair-wise classification or pair-wise regression problem, where a classifier is learnt to predict the ranking orders of document pairs. The group structure of ranking is also ignored as in the point-wise approach. Approaches belonging to this category include Learning-to-Retrieve-Information [11], Ranking SVM [72], RankBoost [65], Linear Discriminant Model (LDM) [66], RankNet [28], Frank [118], Multiple Hyperplane Ranker (MHR) [103], Round Robin Ranking [105], QBRank [128], Magnitude-Preserving Ranking (MPRank) [45], IRSVM [29], etc.

For list-based approaches, the ranking problem is addressed in a more straightforward way since the group structure of ranking orders is maintained and ranking evaluation measures can be more directly incorporated into the loss functions during the learning phase. List-based methods include LambdaRank [27], AdaRank [121], ListNet [31], BoltzRank [120], LambdaMART [26], etc.

Chapter 3

Two-view Transductive Support Vector Machine

In this chapter, we first use a synthetic dataset as an example to illustrate the motivation for two-view semi-supervised learning. Then we propose our two-view transductive support vector machine learning framework, followed by the corresponding optimization approach and algorithm.

3.1 Motivation

We extend the two-view supervised learning algorithm proposed by Farquhar et al. [64] by incorporating unlabeled data, making it a two-view semi-supervised learning approach. The basic idea is to construct two transductive SVM classifiers from both labeled and unlabeled data based on different representations of the original problem, and train these classifiers simultaneously by requiring that they always maintain a maximum consensus on their predictions. By enforcing different classifiers trained from different views to agree on both labeled and unlabeled training data, the structure learnt from each view can reinforce one another. Once trained, a voting or weighting scheme can be used to combine both classifier outputs to classify test samples.

To illustrate the advantage of two-view transductive learning, consider a toy dataset in which samples from two classes appear as two moons in one view and two lines in another, as shown in Figure 3.1 (crosses and circles are used to represent the two classes, respectively). Given only two labeled examples (denoted by a bold cross and circle),

Figure 3.1: Decision boundaries (denoted by the solid line) obtained by the supervised SVM, CCCP TSVM, and Two-view TSVM. The only two labeled examples are represented by a bold cross and circle. The remaining points are unlabeled. Gaussian and linear kernels are used for views 1 and 2, respectively.

the solid lines in Figure 3.1.a and Figure 3.1.d turn out to be the maximum margin hyperplane of the two training instances. They are clearly suboptimal with respect to the underlying distribution of unlabeled data (denoted by the small crosses and circles). Taking unlabeled data into consideration, a transductive SVM [43] shifts the decision boundary away from dense regions, but still fails to yield a good result in either view (Figure 3.1.b and Figure 3.1.e). On the contrary, once a consensus between the two views is imposed on both classifiers, a much better decision boundary is obtained in each view. This is shown in Figure 3.1.c and Figure 3.1.f (decision boundaries are obtained via our two-view transductive SVM), in which the solid decision boundary clearly separates the two classes of data.

3.2 Derivation of the Optimization Problem

Consider a multi-view semi-supervised learning problem on a set of ℓ labeled examples $\{(\mathbf{x}_i^A, \mathbf{x}_i^B), y_i\}_{i=1}^\ell$, $\mathbf{x}_i^{A/B} \in \mathbb{R}^n$, $y_i \in \{-1, 1\}$, and a set of u unlabeled examples $\{\mathbf{x}_i^A, \mathbf{x}_i^B\}_{i=\ell+1}^{\ell+u}$. Superscripts A and B denote the two views, respectively. For each view, we aim to find a decision function $f_\theta(\mathbf{x})$ as shown in Equation (2.3).

According to Collobert et al. [43], for each view, the CCCP transductive SVM has the following objective function

$$J(\theta^{A/B}) = \frac{1}{2} \|\mathbf{w}^{A/B}\|^2 + C^{A/B} \sum_{i=1}^\ell \xi_i^{A/B} + C^{*A/B} \sum_{i=\ell+1}^{\ell+2u} \xi_i^{A/B} + \sum_{i=\ell+1}^{\ell+2u} \rho_i^{A/B} y_i f_\theta^{A/B}(\mathbf{x}_i^{A/B}) \quad (3.1)$$

where $\rho_i^{A/B}$ is related to the derivative of the concave loss function mentioned in Section 2.1.1, written as

$$\rho_i^{A/B} = \begin{cases} C^{*A/B} & \text{if } y_i f_\theta^{A/B}(\mathbf{x}_i^{A/B}) < s \text{ and } i \geq \ell + 1 \\ 0 & \text{otherwise} \end{cases} \quad (3.2)$$

where s is the parameter of the loss function.

In our basic approach, we construct two transductive SVM objective functions based on each the two views, respectively, then add a regularizer to penalize the decision function of each view if it deviates from the consensus, and minimize them simultaneously. To go a step further, we also explore the structure of the data manifold by adding two regularizers that penalize “rapid changes” of the function, which is evaluated on neighboring samples in the Laplacian graph. This leads to the following minimization problem of our two-view transductive SVM

$$\begin{aligned} \min_{\theta^{A/B}, \xi^{A/B}, \eta} \quad & \frac{1}{2} \|\mathbf{f}_\theta^A\|^2 + C^A \sum_{i=1}^\ell \xi_i^A + C^{*A} \sum_{i=\ell+1}^{\ell+2u} \xi_i^A + \sum_{i=\ell+1}^{\ell+2u} \rho_i^A y_i f_\theta^A(\mathbf{x}_i^A) + \\ & \frac{1}{2} \|\mathbf{f}_\theta^B\|^2 + C^B \sum_{i=1}^\ell \xi_i^B + C^{*B} \sum_{i=\ell+1}^{\ell+2u} \xi_i^B + \sum_{i=\ell+1}^{\ell+2u} \rho_i^B y_i f_\theta^B(\mathbf{x}_i^B) + \\ & D \sum_{i=1}^{\ell+2u} \eta_i + E^A \mathbf{f}_\theta^{AT} \mathbf{L}^A \mathbf{f}_\theta^A + E^B \mathbf{f}_\theta^{BT} \mathbf{L}^B \mathbf{f}_\theta^B \end{aligned} \quad (3.3a)$$

$$\text{s.t.} \quad y_i f_\theta^{A/B}(\mathbf{x}_i^{A/B}) \geq 1 - \xi_i^{A/B} \quad (3.3b)$$

$$\xi_i^{A/B} \geq 0 \quad (3.3c)$$

$$|f_{\theta}^A(\mathbf{x}_i^A) - f_{\theta}^B(\mathbf{x}_i^B)| \leq \eta_i \quad (3.3d)$$

$$\eta_i \geq 0 \quad (3.3e)$$

$$\frac{1}{u} \sum_{i=\ell+1}^{\ell+u} f_{\theta}^{A/B}(\mathbf{x}_i^{A/B}) = \frac{1}{\ell} \sum_{i=1}^{\ell} y_i \quad (3.3f)$$

where constraints (3.3b) and (3.3c) are the standard SVM constraints, constraints (3.3d) and (3.3e) impose the consensus between the two views, and constraint (3.3f) is a balancing constraint, which aims to prevent extremely skewed classification result caused by assigning all unlabeled examples to only one class. It has been previously used in [38, 43]. Positive parameters $C^{*A/B}$ control the influence of unlabeled data on the objective function, while parameters $E^{A/B}$ control the influence of the graph-based regularizers. Setting $C^{*A/B}$ and $E^{A/B}$ to zeros will result in a fully-supervised two-view SVM; setting $E^{A/B}$ alone to zero causes the two-view transductive SVM to ignore the manifold information of training samples.

By the Representer Theorem, we know that the decision function has the form

$$f_{\theta}(\mathbf{x}) = \sum_{i=1}^{\ell+2u} \alpha_i K(\mathbf{x}_i, \mathbf{x}) + b = \mathbf{K}\boldsymbol{\alpha} + b \quad (3.4)$$

where $\boldsymbol{\alpha} \in \mathbb{R}^{\ell+2u}$ is a column vector, and b is a scalar bias.

The optimization problem (3.3) can also be written as

$$\begin{aligned} \min_{\boldsymbol{\theta}^{A/B}, \boldsymbol{\xi}^{A/B}, \boldsymbol{\eta}} \quad & \frac{1}{2} \boldsymbol{\alpha}^{AT} \mathbf{K}^A \boldsymbol{\alpha}^A + C^A \sum_{i=1}^{\ell} \xi_i^A + C^{*A} \sum_{i=\ell+1}^{\ell+2u} \xi_i^A + \sum_{i=\ell+1}^{\ell+2u} \rho_i^A y_i \left(\sum_{j=1}^{\ell+2u} \alpha_j^A K^A(\mathbf{x}_i^A, \mathbf{x}_j^A) + b^A \right) + \\ & \frac{1}{2} \boldsymbol{\alpha}^{BT} \mathbf{K}^B \boldsymbol{\alpha}^B + C^B \sum_{i=1}^{\ell} \xi_i^B + C^{*B} \sum_{i=\ell+1}^{\ell+2u} \xi_i^B + \sum_{i=\ell+1}^{\ell+2u} \rho_i^B y_i \left(\sum_{j=1}^{\ell+2u} \alpha_j^B K^B(\mathbf{x}_i^B, \mathbf{x}_j^B) + b^B \right) + \\ & D \sum_{i=1}^{\ell+2u} \eta_i + E^A \boldsymbol{\alpha}^{AT} \mathbf{K}^{AT} \mathbf{L}^A \mathbf{K}^A \boldsymbol{\alpha}^A + E^B \boldsymbol{\alpha}^{BT} \mathbf{K}^{BT} \mathbf{L}^B \mathbf{K}^B \boldsymbol{\alpha}^B \end{aligned} \quad (3.5a)$$

$$\text{s.t.} \quad y_i \left(\sum_{j=1}^{\ell+2u} \alpha_j^A K^A(\mathbf{x}_i^A, \mathbf{x}_j^A) + b^A \right) \geq 1 - \xi_i^A \quad (3.5b)$$

$$\xi_i^A \geq 0 \quad (3.5c)$$

$$y_i \left(\sum_{j=1}^{\ell+2u} \alpha_j^B K^B(\mathbf{x}_i^B, \mathbf{x}_j^B) + b^B \right) \geq 1 - \xi_i^B \quad (3.5d)$$

$$\xi_i^B \geq 0 \quad (3.5e)$$

$$\left| \sum_{j=1}^{\ell+2u} \alpha_j^A K^A(\mathbf{x}_i^A, \mathbf{x}_j^A) + b^A - \sum_{j=1}^{\ell+2u} \alpha_j^B K^B(\mathbf{x}_i^B, \mathbf{x}_j^B) - b^B \right| \leq \eta_i \quad (3.5f)$$

$$\eta_i \geq 0 \quad (3.5g)$$

$$\frac{1}{2u} \sum_{i=\ell+1}^{\ell+2u} \left(\sum_{j=1}^{\ell+2u} \alpha_j^A K^A(\mathbf{x}_i^A, \mathbf{x}_j^A) + b^A \right) = \frac{1}{\ell} \sum_{i=1}^{\ell} y_i \quad (3.5h)$$

$$\frac{1}{2u} \sum_{i=\ell+1}^{\ell+2u} \left(\sum_{j=1}^{\ell+2u} \alpha_j^B K^B(\mathbf{x}_i^B, \mathbf{x}_j^B) + b^B \right) = \frac{1}{\ell} \sum_{i=1}^{\ell} y_i \quad (3.5i)$$

We apply the Lagrange multiplier technique to solve the optimization problem (3.5). The assignment between the Lagrange multipliers and the constraints is summarized as follows

$$\beta_i^A : y_i \left(\sum_{j=1}^{\ell+2u} \alpha_j^A K^A(\mathbf{x}_i^A, \mathbf{x}_j^A) + b^A \right) \geq 1 - \xi_i^A \quad (3.6a)$$

$$\beta_i^B : y_i \left(\sum_{j=1}^{\ell+2u} \alpha_j^B K^B(\mathbf{x}_i^B, \mathbf{x}_j^B) + b^B \right) \geq 1 - \xi_i^B \quad (3.6b)$$

$$\gamma_i^+ : \sum_{j=1}^{\ell+2u} \alpha_j^B K^B(\mathbf{x}_i^B, \mathbf{x}_j^B) + b^B - \sum_{j=1}^{\ell+2u} \alpha_j^A K^A(\mathbf{x}_i^A, \mathbf{x}_j^A) - b^A \geq -\eta_i \quad (3.6c)$$

$$\gamma_i^- : \sum_{j=1}^{\ell+2u} \alpha_j^A K^A(\mathbf{x}_i^A, \mathbf{x}_j^A) + b^A - \sum_{j=1}^{\ell+2u} \alpha_j^B K^B(\mathbf{x}_i^B, \mathbf{x}_j^B) - b^B \geq -\eta_i \quad (3.6d)$$

$$\delta^A : \frac{1}{2u} \sum_{i=\ell+1}^{\ell+2u} \left(\sum_{j=1}^{\ell+2u} \alpha_j^A K^A(\mathbf{x}_i^A, \mathbf{x}_j^A) + b^A \right) = \frac{1}{\ell} \sum_{i=1}^{\ell} y_i \quad (3.6e)$$

$$\delta^B : \frac{1}{2u} \sum_{i=\ell+1}^{\ell+2u} \left(\sum_{j=1}^{\ell+2u} \alpha_j^B K^B(\mathbf{x}_i^B, \mathbf{x}_j^B) + b^B \right) = \frac{1}{\ell} \sum_{i=1}^{\ell} y_i \quad (3.6f)$$

$$\zeta_i^A : \xi_i^A \geq 0 \quad (3.6g)$$

$$\zeta_i^B : \xi_i^B \geq 0 \quad (3.6h)$$

$$\sigma : \eta_i \geq 0 \quad (3.6i)$$

Applying the Lagrange multiplier technique, the minimization problem (3.5) is equivalent to the following problem

$$\begin{aligned} \min_{\tilde{\beta}^{A/B}, \delta^{A/B}, \gamma} & \frac{1}{2} (\mathbf{Y} \tilde{\beta}^A - \gamma + \frac{\delta^A}{2u} \mathbf{J})^T \mathbf{K}^A (\mathbf{I} + 2E^A \mathbf{L}^A \mathbf{K}^A)^{-1} (\mathbf{Y} \tilde{\beta}^A - \gamma + \frac{\delta^A}{2u} \mathbf{J}) + \\ & \frac{1}{2} (\mathbf{Y} \tilde{\beta}^B + \gamma + \frac{\delta^B}{2u} \mathbf{J})^T \mathbf{K}^B (\mathbf{I} + 2E^B \mathbf{L}^B \mathbf{K}^B)^{-1} (\mathbf{Y} \tilde{\beta}^B + \gamma + \frac{\delta^B}{2u} \mathbf{J}) - \\ & \sum_{i=1}^{\ell+2u} (\tilde{\beta}_i^A + \tilde{\beta}_i^B) - \frac{\delta^A + \delta^B}{\ell} \sum_{i=1}^{\ell} y_i \end{aligned} \quad (3.7a)$$

$$\text{s.t.} \quad 0 \leq \tilde{\beta}_i^A \leq C^A \quad \forall 1 \leq i \leq \ell \quad (3.7b)$$

$$- \rho_i^A \leq \tilde{\beta}_i^A \leq C^{*A} - \rho_i^A \quad \forall \ell + 1 \leq i \leq \ell + 2u \quad (3.7c)$$

$$0 \leq \tilde{\beta}_i^B \leq C^B \quad \forall 1 \leq i \leq \ell \quad (3.7d)$$

$$- \rho_i^B \leq \tilde{\beta}_i^B \leq C^{*B} - \rho_i^B \quad \forall \ell + 1 \leq i \leq \ell + 2u \quad (3.7e)$$

$$-D \leq \gamma_i \leq D \quad \forall \ell + 1 \leq i \leq \ell + 2u \quad (3.7f)$$

$$\sum_{i=1}^{\ell+2u} (\tilde{\beta}_i^A y_i - \gamma_i) + \delta^A = 0 \quad (3.7g)$$

$$\sum_{i=1}^{\ell+2u} (\tilde{\beta}_i^B y_i + \gamma_i) + \delta^B = 0 \quad (3.7h)$$

where $\tilde{\beta}^A = \beta^A - \rho^A$, $\tilde{\beta}^B = \beta^B - \rho^B$, $\gamma = \gamma^+ - \gamma^-$, $\mathbf{Y} = \text{diag}(y_1, \dots, y_{\ell+2u})$ is a diagonal matrix, and $\mathbf{J}$ is a $(\ell + 2u) \times 1$ column vector whose first ℓ elements equal to zero and last $2u$ elements equal to one.

3.3 Augmented Lagrangian Technique

To solve the minimization problem (3.7), we employ the augmented Lagrangian [97] technique as Farquhar et al. [64] did. Augmented Lagrangian is a method for solving constrained optimization problems. It reformulates a constrained optimization problem into an unconstrained one by adding Lagrange multipliers and an extra penalty term for each constraint to the original objective function. The augmented Lagrangian function corresponding to the minimization problem

$$\min_x f(x) \quad (3.8a)$$

$$\text{s.t. } c_i(x) = 0 \quad \forall 1 \leq i \leq n \quad (3.8b)$$

can be written as

$$\min_x f(x) - \sum_{i=1}^n \lambda_i c_i(x) + \frac{\mu}{2} \sum_{i=1}^n c_i^2(x) \quad (3.9)$$

where the first two terms correspond to the Lagrangian and the last term is the penalty for violating the constraints. The penalty term is positive when the current point x violates the constraint and zero otherwise. It is multiplied by a positive coefficient μ . By making this coefficient larger, we penalize constraint violations more severely, thereby forcing the minimizer (to the penalty function) closer to the feasible region for the constrained problem.

The minimization problem (3.9) can be solved iteratively. In each iteration, $\boldsymbol{\lambda}$ is fixed to some estimate of the optimal Lagrange multiplier and the penalty parameter μ is set to some positive value, then one can perform minimization with respect to x . In subsequent iterations, $\boldsymbol{\lambda}$ and μ are updated; and the process is repeated until some stopping criterion is met. It is shown that convergence of the augmented Lagrangian method is assured provided that μ is not increasing indefinitely [97].

3.4 Two-view Transductive SVM Algorithm

Let us denote the equality constraints (3.7g) and (3.7h) as h_1 and h_2 , and introduce corresponding Lagrange multipliers λ_1 and λ_2 . We can rewrite the minimization problem (3.7) into the augmented Lagrangian form as follows

$$\begin{aligned} \min_{\tilde{\beta}^{A/B}, \delta^{A/B}, \gamma} & \frac{1}{2} (\mathbf{Y} \tilde{\beta}^A - \gamma + \frac{\delta^A}{2u} \mathbf{J})^T \mathbf{K}^A (\mathbf{I} + 2E^A \mathbf{L}^A \mathbf{K}^A)^{-1} (\mathbf{Y} \tilde{\beta}^A - \gamma + \frac{\delta^A}{2u} \mathbf{J}) + \\ & \frac{1}{2} (\mathbf{Y} \tilde{\beta}^B + \gamma + \frac{\delta^B}{2u} \mathbf{J})^T \mathbf{K}^B (\mathbf{I} + 2E^B \mathbf{L}^B \mathbf{K}^B)^{-1} (\mathbf{Y} \tilde{\beta}^B + \gamma + \frac{\delta^B}{2u} \mathbf{J}) - \\ & \sum_{i=1}^{\ell+2u} (\tilde{\beta}_i^A + \tilde{\beta}_i^B) - \frac{\delta^A + \delta^B}{\ell} \sum_{i=1}^{\ell} y_i - \sum_{i=1}^2 \lambda_i h_i + \frac{\mu}{2} \sum_{i=1}^2 \|h_i\|^2 \end{aligned} \quad (3.10a)$$

$$\text{s.t. } 0 \leq \tilde{\beta}_i^A \leq C^A \quad \forall 1 \leq i \leq \ell \quad (3.10b)$$

$$-\rho_i^A \leq \tilde{\beta}_i^A \leq C^{*A} - \rho_i^A \quad \forall \ell + 1 \leq i \leq \ell + 2u \quad (3.10c)$$

$$0 \leq \tilde{\beta}_i^B \leq C^B \quad \forall 1 \leq i \leq \ell \quad (3.10d)$$

Algorithm 2: Two-view Transductive SVM

Require: Labeled and unlabeled data of two views.

Initialize $\tilde{\beta}^{A/B}$, $\rho^{A/B}$, γ , λ and μ .

repeat

Solve the following sub-problem.

repeat

Solve the minimization problem (3.10) with fixed λ^k and μ^k .

Compute $f_\theta^{A(t+1)}$ and $f_\theta^{B(t+1)}$ by equation (3.11) with the solution of the minimization problem (3.10).

Compute $\rho^{A(t+1)}$ and $\rho^{B(t+1)}$ by equation (3.2) with the value of $f_\theta^{A(t+1)}$ and $f_\theta^{B(t+1)}$.

Update the lower and upper bounds of $\tilde{\beta}^{A(t+1)}$ and $\tilde{\beta}^{B(t+1)}$ by equation (3.10c) and (3.10e).

until $\rho^{A(t+1)} = \rho^{A(t)}$ and $\rho^{B(t+1)} = \rho^{B(t)}$

Update the Lagrange multiplier λ by

$$\lambda^{k+1} = \lambda^k + \mu^k \mathbf{h}^k$$

Update the penalty parameter μ by

$$\mu^{k+1} = \phi \mu^k$$

until $\|\mathbf{h}^k\| \leq \epsilon$

return The decision functions corresponding to the two views calculated by equation (3.11).

$$-\rho_i^B \leq \tilde{\beta}_i^B \leq C^{*B} - \rho_i^B \quad \forall \ell + 1 \leq i \leq \ell + 2u \quad (3.10e)$$

$$-D \leq \gamma_i \leq D \quad \forall \ell + 1 \leq i \leq \ell + 2u \quad (3.10f)$$

where $\mu > 0$ is the penalty parameter.

Once the minimization problem (3.10) is solved with the augmented Lagrangian method, the decision functions corresponding to the two views can be calculated as follows

$$f_\theta^A(\mathbf{x}^A) = \sum_{i=1}^{\ell+2u} (y_i \tilde{\beta}_i^A + \gamma_i) K^A(\mathbf{x}_i^A, \mathbf{x}^A) + b^A \quad (3.11a)$$

$$f_\theta^B(\mathbf{x}^B) = \sum_{i=1}^{\ell+2u} (y_i \tilde{\beta}_i^B - \gamma_i) K^B(\mathbf{x}_i^B, \mathbf{x}^B) + b^B \quad (3.11b)$$

A hybrid decision function can be written as a linear combination of the two classifiers as

$$f(\mathbf{x}) = \omega f_{\theta}^A(\mathbf{x}^A) + (1 - \omega) f_{\theta}^B(\mathbf{x}^B) \quad (3.12)$$

where $0 \leq \omega \leq 1$.

Algorithm 2 summarizes the two-view transductive SVM algorithm. The convergence of the CCCP procedure is described in [43]. A detailed convergence analysis of the Lagrange multiplier iteration, which corresponds to the outer loop of Algorithm 2 can be found in [19]. In our experiments, we observed that Algorithm 2 always converges as long as the parameters are selected appropriately.

3.5 Experimental Results

In this section, we evaluate the performance of our two-view transductive SVM on a toy dataset and three real-life datasets: the ads dataset, the well-known WebKB course dataset, and our product review dataset.

3.5.1 Toy Dataset

Figure 3.2: Illustration of the toy dataset.

For the toy dataset, 200 samples from two classes appear as two moons in one view and two lines in another as shown in Figure 3.2. For each view, two samples are treated as labeled data while the remaining 198 samples are unlabeled. Figure 3.1 depicts the classification results of supervised SVM (trained with only two labeled samples), CCCP

transductive SVM [43], and our two-view transductive SVM. The superiority of our two-view transductive SVM is self-evident by comparing the contours of the various generated decision boundaries. The failure of supervised SVM is not surprising due to insufficient labeled training data. The sub-standard performance of CCCP TSVM shown in Fig. 3.1.e may be ascribed to the fact that the two lines are too close, blurring the boundary between the two classes. We found experimentally that unlabeled data can affect the decision boundary of CCCP TSVM if the gap between the two lines were enlarged.

3.5.2 Ads Dataset

The ads dataset was first used by Kushmerick [85] to study methods that automatically remove advertisement images from Internet pages. The task is to predict whether an online image in the dataset is used for advertisement or not. The ads dataset has more than two views, but we used only three of the views to evaluate our algorithm. They are the *image URL view* (457 features related to the image server name), the *destination URL view* (472 features related to the image URL), and the *alt view* (111 features related to “alternate” words in the HTML). Amongst 3279 examples in the dataset, 459 examples belong to the positive class (ads), with the remaining as non-ads.

We benchmarked our two-view transductive SVM (Two-view TSVM) against the standard supervised SVM (LIBSVM [37] trained with a few labeled examples), the two-view supervised learning algorithm — SVM-2K [64] (trained with only labeled examples), the two-view semi-supervised learning algorithm — co-training [24], the graph-based semi-supervised learning algorithm — Laplacian SVM [113], and the single-view transductive SVM — CCCP TSVM [43]. We generated 10 random splits of the ads dataset. Each split contains 32 labeled and 3247 unlabeled examples. For all algorithms, the unlabeled data were used as the test set. Since the distribution of ads dataset is skewed (only 459 of 3279 examples belong to the positive class), we reported the F1-measure besides accuracy. The F1-measure is the harmonic mean of the precision and recall values. Hence, a high F1-measure ensures that both precision and recall are reasonably high. Note that it is always harder for a classifier to achieve a good F1-measure compared to accuracy on a skewed dataset.

Table 3.1: Ads dataset result showing mean accuracy, F1-measure (%) and their standard deviation (in brackets).

(a) Image URL View and Destination URL View

	Image URL View			Destination URL View			Hybrid View		
	Acc	Pos F1	Neg F1	Acc	Pos F1	Neg F1	Acc	Pos F1	Neg F1
SVM (32 labeled)	87.01 (2.22)	16.90 (24.33)	92.93 (1.14)	90.02 (2.06)	49.44 (19.02)	94.44 (1.14)	-	-	-
SVM-2K (32 labeled)	83.83 (12.70)	41.91 (17.57)	89.88 (10.11)	92.03 (1.10)	64.82 (5.79)	95.50 (0.62)	91.91 (0.84)	64.13 (6.51)	95.44 (0.45)
Co-training	81.87 (4.43)	39.05 (10.06)	89.29 (2.83)	83.98 (3.92)	43.20 (9.76)	90.63 (2.46)	83.15 (4.01)	42.95 (9.94)	90.87 (2.84)
Laplacian SVM	87.59 (1.63)	25.02 (15.48)	93.23 (0.86)	89.07 (1.27)	35.73 (12.35)	94.02 (0.65)	89.35 (1.01)	36.07 (13.46)	92.55 (1.91)
CCCP TSVM	88.01 (3.58)	45.15 (20.05)	93.20 (2.23)	90.91 (1.66)	61.35 (7.94)	94.83 (0.99)	-	-	-
Two-view TSVM	90.17 (7.08)	66.27 (11.38)	94.14 (4.78)	91.82 (6.56)	72.38 (10.20)	95.08 (4.51)	92.13 (6.71)	73.99 (11.33)	95.27 (4.54)
Two-view TSVM with Laplacian	92.53 (1.73)	70.10 (5.62)	95.73 (1.02)	93.69 (1.93)	75.19 (6.46)	96.38 (1.14)	93.81 (2.09)	76.31 (7.00)	96.44 (1.23)

(b) Destination URL View and Alt View

	Destination URL View			Alt View			Hybrid View		
	Acc	Pos F1	Neg F1	Acc	Pos F1	Neg F1	Acc	Pos F1	Neg F1
SVM (32 labeled)	90.02 (2.06)	49.44 (19.02)	94.44 (1.14)	87.73 (1.47)	28.48 (20.54)	93.27 (0.72)	-	-	-
SVM-2K (32 labeled)	92.59 (1.02)	68.11 (4.76)	95.81 (0.59)	87.98 (0.87)	45.80 (1.78)	93.23 (0.55)	91.24 (1.03)	66.04 (4.58)	94.96 (0.61)
Co-training	81.64 (4.07)	44.30 (5.95)	88.97 (2.68)	83.27 (3.38)	37.37 (5.50)	90.31 (2.17)	83.87 (4.19)	40.55 (5.21)	90.18 (2.87)
Laplacian SVM	89.07 (1.27)	35.73 (12.35)	94.02 (0.65)	88.96 (0.41)	39.48 (5.14)	93.92 (0.20)	90.23 (1.51)	37.54 (11.01)	94.67 (0.57)
CCCP TSVM	90.91 (1.66)	61.35 (7.94)	94.83 (0.99)	88.65 (0.95)	44.30 (1.30)	93.67 (0.60)	-	-	-
Two-view TSVM	91.06 (1.52)	67.27 (2.97)	94.81 (0.97)	86.78 (1.72)	42.37 (3.32)	92.52 (1.08)	90.97 (1.79)	68.25 (3.17)	94.73 (1.15)
Two-view TSVM with Laplacian	92.64 (0.64)	71.02 (2.23)	95.78 (0.38)	88.57 (0.97)	45.02 (2.04)	93.62 (0.59)	92.27 (0.89)	71.18 (2.28)	95.53 (0.55)

We manually tuned and found the best parameters for each algorithm, using the positive class F1-measure on the unlabeled dataset. For simplicity and fairness, we first tuned the parameters (C_1 and C_2) for CCCP TSVM and used the same values for our Two-view TSVM. The disagreement penalty (D) was selected from a small range of values for the Two-view TSVM. Note that in the following, “Two-view TSVM” denotes the two-view transductive SVM algorithm without manifold regularization (i.e., parameters E^A and E^B are set to zero), and “Two-view TSVM with Laplacian” includes manifold regularization. The “Hybrid View” for Laplacian SVM uses the sum of graph Laplacians in each view for regularization (see [113] for details), while the “Hybrid View” for Two-view TSVM uses a linear combination of the views’ outputs.

The average accuracy, class-specific F1-measures, and corresponding standard deviations on the unlabeled test examples across 10 random splits of the ads dataset are shown in Table 3.1 for different combination of views. To assess the statistical significance of the Two-view TSVM result, we performed an unpaired t-test at 5% significance level with CCCP TSVM as a baseline. Results shown in bold are considered statistically significant. From Table 3.1, we can see that most of the algorithms achieved high accuracies, but many of them scored remarkably low F1-measures for the positive class. Considering the minority of the positive class in the ads dataset (14% of samples belong to the positive class), one can conclude that algorithms with low positive class F1-measure actually fail to make the right prediction on the test set. The poor performance of plain SVM is as expected since the model is only trained with a few labeled data in the traditional supervised sense. The failure of co-training is probably because the ads dataset violates its key assumption, that sub-features are sufficiently good and conditionally independent.

Compared to other algorithms, our Two-view TSVM achieves consistently higher accuracy and F1-measures in most cases. It is also noted that two-view TSVM with Laplacian graph regularization yields better results than the plain two-view TSVM. The improvement on the positive class’s F1-measure is significant, e.g., 70.10% for two-view TSVM with Laplacian graph regularization versus 45.15% for CCCP TSVM for Image URL view in Table 3.1(a). These results show that the proposed two-view semi-supervised learning algorithm not only performs more accurately, but also achieves considerably more stable results than the regular single-view learning approach.

3.5.3 WebKB Course Dataset

The WebKB course dataset has been frequently used in the empirical study of multi-view learning since it was first introduced by Blum et al. [24]. The dataset contains 1051 web pages collected from computer science departments of four universities. The task is to classify each page into two classes: course or non-course. The two views are the textual content of a web page (*page view*) and the words that occur in the hyperlinks of other web pages pointing to that one (*link view*). We borrowed a processed WebKB course dataset from Sindhvani et al. [113] and used it in our experiment.

Table 3.2: WebKB course dataset result showing mean accuracy, F1-measure (%), and their standard deviation (in brackets).

	Page View			Link View			Hybrid View		
	Acc	Pos F1	Neg F1	Acc	Pos F1	Neg F1	Acc	Pos F1	Neg F1
SVM (12 labeled)	71.17 (17.22)	5.74 (15.88)	80.65 (19.35)	73.81 (17.20)	6.47 (16.05)	82.13 (20.98)	-	-	-
SVM-2K (12 labeled)	88.90 (5.83)	73.07 (13.07)	92.76 (5.79)	88.73 (10.03)	72.68 (11.47)	92.03 (13.12)	90.87 (9.10)	78.56 (11.43)	93.60 (10.69)
Co-training	62.46 (17.30)	22.05 (19.81)	70.98 (20.27)	62.94 (16.41)	21.84 (20.16)	72.03 (17.83)	63.80 (15.28)	22.85 (19.77)	73.84 (18.27)
Laplacian SVM	89.49 (5.77)	64.82 (29.46)	93.73 (3.16)	78.90 (18.42)	39.71 (33.72)	83.20 (24.66)	87.95 (8.17)	52.52 (39.95)	93.02 (4.51)
CCCP TSVM	87.33 (9.04)	62.76 (33.50)	91.78 (7.07)	90.65 (7.98)	73.76 (20.27)	93.85 (8.88)	-	-	-
Two-view TSVM	88.67 (5.46)	78.71 (7.99)	92.23 (4.07)	93.04 (2.84)	83.28 (7.13)	95.60 (1.78)	92.49 (3.09)	83.80 (5.90)	95.10 (2.10)
Two-view TSVM with Laplacian	89.94 (5.31)	80.79 (7.92)	93.14 (3.93)	93.64 (2.68)	84.89 (6.68)	95.97 (1.69)	93.23 (2.74)	85.34 (5.35)	95.59 (1.84)

The set of optimal parameters (C and C^*) was chosen from a small range of values, and applied to both the CCCP TSVM and Two-view TSVM. Both algorithms were run over 100 random splits of the WebKB course dataset. Each split contains 12 labeled and 1039 unlabeled examples. Since the distribution of the WebKB course dataset is skewed (only 230 of 1051 examples belong to the positive class), we reported the F1-measure besides accuracy. The mean and standard deviation results of accuracy and F1-measure on the unlabeled test examples are tallied in Table 3.2 for Laplacian SVM, CCCP TSVM, and Two-view TSVM, respectively. The result for Laplacian SVM was originally reported in [113].

Compared to Laplacian SVM and CCCP TSVM, Two-view TSVM achieved consistently higher accuracy and F1-measure values. Specifically, performance for the positive class's F1-measure is more than 11% better (84.89% for Two-view TSVM versus 73.76% for CCCP TSVM). Further, the variation (standard deviation shown in brackets) for Two-view TSVM is on average three to four times lower than that of the CCCP TSVM. For example, the Two-view TSVM accuracy has a standard deviation of 2.84, compared to the standard deviation of 7.98 for CCCP TSVM accuracy. These results show that the proposed Two-view TSVM performs not only more accurately but also achieves considerably more stable results compared to the single-view approach.

Figure 3.3 depicts the detailed F1-measure results of both positive and negative classes over 100 random splits of the test dataset for both the CCCP TSVM, Two-view TSVM,

3.3.a: Variation of the positive class's F1-measure (page view).

3.3.b: Variation of the positive class's F1-measure (link view).

3.3.c: Variation of the negative class's F1-measure (page view).

3.3.d: Variation of the negative class's F1-measure (link view).

Figure 3.3: Variation of the positive and negative classes' F1-measures over 100 splits of the WebKB course dataset. This figure shows CCCP TSVM, Two-view TSVM, and Laplacian SVM's variation on the positive and negative classes' F1-measures.

and Laplacian SVM. It can be seen that the performance of CCCP TSVM and Laplacian SVM is rather unstable, which oscillates between zero and non-zero F1-measures. This happens when every test example is classified into one class (despite the balancing constraint (3.3f) imposed on CCCP TSVM).

On the contrary, by simultaneously training two transductive SVMs based on two views, the Two-view TSVM successfully overcame this problem. In fact, the F1-measure for Two-view TSVM remains relatively stable, regardless of changes in the training/test data. Since the amount of labeled data in semi-supervised learning is relatively small, there are always variations in the small training set. The variability among training examples is considered one of the primary sources of errors in a classifier. By requiring two classifiers to agree with each other, the structure learnt from each view can reinforce

one another, and the effect of large variations in the training set can be reduced. Further, the hybrid classifier output is a weighted sum of the individual classifier outputs, which effectively reduces the probability of large swings; any major disagreement between the two classifiers is essentially averaged out by the linear combination.

3.5.4 Product Review Dataset

Our product review dataset was downloaded from two popular online Chinese cell-phone forums¹. Redundant punctuations and stop words were removed and reviews containing less than four characters were eliminated, since they may not hold enough information for a review mining system. We then manually labeled 1000 true reviews and 1000 spam reviews according to the following criteria. A product review is regarded as useful or non-spam if (1) it contains a declarative sentence (all questions are regarded as spam reviews), and (2) it expresses opinions on a product or product feature. Opinions include the reviewer's personal sentiment (positive or negative) towards a product or product feature, and/or the pros and cons analysis of a product or product feature.

To illustrate the idea, consider the following translated cellphone forum snippet.

Example 1 *This cell phone works better than any that I've ever had. I really like it.*

Example 2 *I will buy this phone to try it out.*

Example 3 *How is the battery life of this cell phone?*

Based on our criteria, Example 1 is a useful product review with positive opinion on the product, whereas Examples 2 and 3 are spam reviews since they fail to comment on the product.

We treat the product review filtering task as a binary classification problem. To train the classifier, we define two sets of features: one based on the review content (which we call the *lexical view*) and the other based on the characteristics of the review sentences (*formal view*). For the lexical view, since there is no space separator between Chinese words, raw reviews were preprocessed by a Chinese lexical analyzer — ICTCLAS².

¹<http://www.club.mobile.163.com> and <http://www.3533.com>

²<http://www.ictclas.org>

ICTCLAS performs word segmentation and part-of-speech tagging. Each sentence was converted to a word vector using the standard TF-IDF (term frequency-inverse document frequency) representation. For the formal view, five types of features are enumerated as follows.

- (i) Proportion of opinion-bearing phrases in the review sentence. Opinion-bearing phrase refers to adjectives and other terms that are used to express subjective opinions. A review that contains many opinion-bearing phrases is likely to contain comments on the product. We manually crafted a list of dictionary phrases to detect opinion-bearing phrases in review sentences.
- (ii) Proportion of questioning patterns in the review sentence. Prior work has found that the linguistic style in which reviews are written is a good indicator of spam/non-spam reviews [126]. We extracted the number of questioning patterns from a review. The questioning pattern dictionary includes common Chinese question structures.
- (iii) Proportion of numerical digits in the review sentence. Empirical observations suggest that sentences containing many numerical digits tend to talk about product pricing or refer to a spammer's contact number rather than product review. This distinction is useful for detecting spam reviews.
- (iv) Percentage of brand mentions in the review sentence. A high percentage of brand mentions in the sentence suggests that the review is a comment on brand only, or an advertisement, both of which are treated as spam reviews.
- (v) Length of review sentence. This feature is chosen since advertisements (which are regarded as spam reviews) are typically longer than true reviews. For example, the average length of true reviews in our dataset is 21 characters, whereas that of advertisements is 56.

Figure 3.4 shows each of the five features along with some sample dictionary terms and the dictionary size information.

To evaluate the discriminative power of our proposed high level features against the text corpus features, we trained a supervised SVM to classify product reviews based only

Feature Type	Sample Dictionary Terms	Dictionary Size
Opinion phrases	漂亮 (beautiful), 难看 (ugly), 贵 (expensive), 好用 (easy to use), 清晰 (vivid) ...	551 phrases
Question patterns	?, 哪里 (where), 怎么样 (how about), 谁 (who), 为何 (why), 什么 (what) ...	32 patterns
Digits	0 - 9	10 characters
Brands	摩托罗拉 (Motorola), 多普达 (Dopod), 诺基亚 (Nokia), 三星 (Samsung) ...	12 brands
Length of review	Counts the total number of Unicode Characters (Chinese, English letters, and digits) in the review, excluding punctuations.	—

Figure 3.4: The five extracted features and their sample dictionary terms where applicable.

on the lexical or formal view. Ten fold cross validation test accuracy was 89.90% and 85.29% for the lexical and formal views, respectively. This indicates that each of the two views contain sufficient information to train a standalone classifier.

To simulate the sparsity of labeled data, we generated 100 random splits of the product review dataset. Each split contains 20 labeled, 1580 unlabeled, and 400 validation examples. We tuned the parameters for each algorithm with respect to their accuracy on the validation dataset, making use of the labels. For the supervised SVM and SVM-2K, the models were trained with only 20 labeled examples. For the semi-supervised CCCP TSVM and Two-view TSVM, models were trained with 20 labeled and 1580 unlabeled examples. For all algorithms, we used the unlabeled data as the test set.

The average accuracy and its standard deviation for each algorithm across the 100 dataset splits is shown in Table 3.3. It can be seen that our Two-view TSVM achieves the best accuracy compared to all other methods. To assess the importance of unlabeled data in situations where labeled data are really sparse, we evaluated the performances of CCCP TSVM versus our Two-view TSVM for increasing amount of labeled data starting from 20 up to 1000. Figure 3.5 gives a plot of accuracy versus number of labeled data for CCCP TSVM and Two-view TSVM. As expected, both algorithms improve with increasing number of labeled examples. The key thing to note here is that the performance of the Two-view TSVM is around 5% better than the best CCCP TSVM at the lowest amount of labeled data. As the number of labeled data increases, both algorithms performed more or less in the same ballpark. From the figure, we can conclude that the Two-view TSVM

Table 3.3: Product review dataset result showing mean accuracy, F1-measure (%) and their standard deviation (in brackets).

	Lexical View			Formal View			Hybrid View		
	Acc	Pos F1	Neg F1	Acc	Pos F1	Neg F1	Acc	Pos F1	Neg F1
SVM (20 labeled)	68.10 (10.97)	65.88 (13.91)	69.01 (10.68)	55.02 (20.05)	51.43 (28.56)	56.12 (13.74)	-	-	-
SVM-2K (20 labeled)	74.22 (5.24)	73.97 (6.13)	73.63 (5.66)	70.80 (4.76)	66.32 (9.99)	72.74 (5.61)	75.18 (4.95)	74.91 (6.17)	74.87 (5.64)
Co-training	70.15 (12.98)	70.86 (13.42)	69.36 (12.52)	66.83 (9.07)	64.95 (23.37)	62.88 (14.91)	78.65 (13.80)	69.01 (24.35)	68.04 (15.22)
Laplacian SVM	58.92 (7.88)	40.48 (25.61)	60.15 (23.10)	69.36 (7.49)	68.08 (8.30)	69.86 (8.84)	66.01 (5.79)	48.82 (19.98)	65.27 (7.81)
CCCP TSVM	76.30 (2.27)	74.50 (2.48)	77.85 (2.21)	74.39 (6.03)	73.74 (6.10)	74.78 (6.60)	-	-	-
Two-view TSVM	79.98 (4.80)	79.35 (6.70)	80.23 (5.05)	74.78 (5.35)	74.39 (5.29)	75.03 (5.77)	77.43 (5.41)	77.29 (5.51)	77.46 (5.74)
Two-view TSVM with Laplacian	80.42 (3.99)	80.14 (4.18)	80.57 (4.31)	74.97 (5.36)	74.58 (5.26)	75.22 (5.84)	77.55 (5.37)	77.48 (5.27)	77.52 (5.80)

shines when the amount of labeled data is very small, but it also slightly outperformed the single-view classifiers as the amount of labeled data increases. Therefore, it is safe to employ the Two-view TSVM regardless of the amount of labeled data at hand, as it always produces equal or better results than the classifier trained on a single view.

The experimental results shown in Table 3.3 are obtained by averaging multiple runs over 100 random splits of the product review dataset. We randomly picked out one result to perform error analysis. Amongst the 315 wrongly classified reviews, 153 are false positives (spams wrongly predicted as true reviews), and 162 are false negatives (true reviews wrongly predicted as spams). Through manual inspection of errors, we found out that a large portion of errors are sentences with a few characters. This is expected as compared to long sentences, short text segments provide fewer inputs to the classifier. For further improvement, one may refine the data set by excluding review segments whose character count is lower than a threshold. In practice, one can process those short review segments by alternative methods, such as using hard-crafted rules. Another concern is about the feature representation. Recall that we represent each review sentence as a vector with TF-IDF weight as components. In this view of a sentence, known in the literature as the bag-of-words model, the exact ordering of the terms in the sentence and the grammatical relations are ignored but the number of occurrences of each term matters. It is therefore useful to build feature set considering constructs of co-occurrence terms and their relationships. We may have to turn to some more sophisticated models such as latent semantic analysis. But that's beyond the scope of this research.

Figure 3.5: Accuracy versus the number of labeled examples for CCCP TSVM and Two-view TSVM on the product review dataset.

We also note that although the proposed two-view transductive SVM performs well on this dataset, there is still long way to go to build a realistic spam review filter. One possible direction is to refine the definition of spam reviews. The current definition treats non-declarative sentences as spam reviews. Although rare in our dataset, there are cases where a product review segment contains both declarative and question sentences. In this situation, using the aforementioned criteria to determine whether it is spam or not leads to contradictory conclusion. Therefore, a two-step procedure for spam review filtering is needed, wherein it first detects spam on the sentence level and then decide the overall possibility of a text segment as spam by aggregating the individual sentence decisions.

3.6 Discussion

In this chapter, we investigated the problem of multiview semi-supervised learning, and proposed a two view transductive SVM technique, which is able to take advantage of multiple representations of the same problem to achieve an improvement in classification performance for problems lacking labeled data. Our technique was motivated by extending the existing two-view supervised learning algorithm into a semi-supervised learning setting. We also incorporated the idea of graph-based semi-supervised learning into our algorithm by utilizing the intrinsic manifold structure of the data. We formulated our learning framework into an optimization problem and presented an effective way to solve it. Experimental results on both synthetic and real-life datasets validated the efficiency of the proposed two-view transductive learning algorithm when comparing

with the state-of-the-art single-view/multi-view supervised/semi-supervised learning approaches. In particular, the proposed technique always makes the classifier more accurate and stable by requiring two views to maintain a maximum consensus on both labeled and unlabeled data. Further, incorporating additional manifold information can also help to improve the classification results.

Our two-view transductive SVM was also partially motivated by the need to detect spam product reviews from online forums. Experimental results were promising on the review spam detection task: a model trained with a few labeled data using our algorithm is comparable to one trained on a significantly larger amount of labeled data using the supervised learning approach. The task of product review mining can be enhanced by applying our method to detect and filter spam reviews.

Given data represented by a set of features, how can we split the features into two or more views so that the multi-view setup achieves better results than the single view feature set? The answer to this question is specific to each dataset. In general, if the two views are generated independently and differ by a large margin in characteristics like dimensionality, then the two view approach would be preferred.

Chapter 4

Collaborative Online Multitask Learning

In this chapter, we present the first task-centric collaborative learning method — collaborative online multitask learning. We begin with a brief motivation, followed by methodology and experimental evaluations.

4.1 Motivation

Classical machine learning methods are often formulated as a single task learning problem, which by definition learns one task at a time. On the contrary, multitask learning aims to solve multiple related learning tasks in parallel. Many real-world problems are essentially multitask learning, although they are often broken down into smaller single learning tasks, which are then solved individually by classical learning methods. Multitask learning has been extensively studied in machine learning and data mining over the past decade [33, 61, 44, 39]. Empirical findings have demonstrated the advantages of multitask learning over single task learning across a variety of application domains.

The classical multitask learning methodology [33] often makes two assumptions. First, it assumes there is one primary task and other related tasks are simply secondary tasks whose training data are simply exploited by multitask learning to improve the primary task. Thus, the classical multitask learning approach focuses on learning the primary task independent of how the other tasks are learned or fared. Second, the classical multitask learning problem is often studied in a batch learning setting, which assumes that the

training data of all tasks are available. On one hand, this assumption is not realistic for many real-world problems where data arrives sequentially. On the other hand, the batch multitask learning algorithms usually have fairly intensive training costs and poor scalability performance, as far as large real applications are concerned.

We investigate the problem of online multitask learning, which differs from the classical multitask learning in two aspects. First, our goal is to improve the learning performance of all tasks instead of focusing on a single primary task. Second, we frame the multitask learning problem in an online learning setting by assuming that the data for each task arrives sequentially, which is a more realistic scenario for real-world applications. Unlike batch learning techniques, online learning is a family of efficient and scalable learning techniques, which learns over a sequence of data by processing each sample upon arrival. At each round, the learner first receives one instance, makes a prediction, and receives the true label. This error information is then used to update the learning model.

Our study was motivated by the need to classify online user-generated content (UGC), e.g., micro-blog posts or spam email tags. For UGC, each individual exhibits uniqueness, but also shares certain characteristics with others in the group. It is thus desirable to develop an efficient and scalable classifier that can solve individual task by adapting to the global knowledge shared by all users. Consider the real problem of micro-blog sentiment analysis on a group of users where the goal is to classify micro-blog posts generated by each user into several emotional or non-emotional categories in a near real-time manner, i.e., online. When solving this problem by classical machine learning techniques, we will face a dilemma, i.e., a single global classification model trained on the entire collection of data from all users may fail to capture the peculiarity of individual users and thus often works poorly. On the other hand, a fully personalized model for each user may be inaccurate due to insufficient training data, especially at the early stage of the learning task. This thus motivates us to study online multitask learning techniques.

We propose a novel collaborative online multitask learning (COML) technique to attack the challenge of solving multiple related tasks in parallel by an online learning approach. The basic idea of the proposed COML technique is to first build a generic *global* model from large amount of data gathered from all users, and then subsequently leverage the global model to build the *personalized* classification models for individual users

through a *collaborative* learning process. We formulate this idea into an optimization problem under an online learning setting, and propose two different COML algorithms by exploring different kinds of online learning methodologies.

To evaluate the efficacy of the proposed technique, we conducted extensive experiments by comparing our algorithms against a variety of state-of-the-art techniques on a synthetic dataset and three real-life applications, including online spam email filtering, peptide binding prediction in bioinformatics, and micro-blog sentiment analysis. Our results show that the proposed COML algorithms outperform (1) a single task online learning approach that simply learns a global model over the entire collection of data gathered from all the tasks, (2) a single task online learning approach that solves each task independently, and (3) a state-of-the-art approach for online multitask learning.

4.2 Methodology

4.2.1 Overview

Figure 4.1 illustrates the idea of our method. Specifically, the collaborative online multitask learning operates in a sequential manner. At each learning round, it collects the current global set of data; one from each of the engaged users/tasks, which are employed to update the global classification model. At the same time, a collaborative personalized model is maintained for each user/task. The individual collaborative classification model is subsequently updated using the latest individual data and the global model parameters. Therefore, our approach can leverage global knowledge for classification, while adapting to individual nuances via collaborative learning.

4.2.2 Formulation and Algorithms

We now formulate the problem in a binary classification setting. Our algorithm can be easily extended to address the multi-class problems by adopting techniques described in [47, 48].

Online multitask classification proceeds in rounds by observing a sequence of examples, each belonging to some user/task from a set of K users/tasks. On each round, there are K separate online binary classification problems being solved jointly. We assume that

Figure 4.1: Leveraging data from a series of users via collaborative online multitask learning.

data from all users/tasks can be represented in the same global feature space, so that it is possible to use the shared information between tasks to enhance each learning task. Denote by $(\mathbf{x}_t^k, y_t^k)$ a training instance belonging to the k -th user at round t , where $\mathbf{x}_t^k \in \mathbb{R}^d$ is a d -dimensional vector representing the example and $y_t^k \in \{1, -1\}$ refers to its class label. We henceforth omit the superscript of $\mathbf{x}_t^k$ below for brevity.

Our goal is to learn a set of classification models to maximize the online prediction accuracy of every task, i.e., $f^{(k)}(\cdot) : \mathbb{R}^d \mapsto \{1, -1\}, k = 1, \dots, K$. In this work, we consider a linear classification model for each task, which is parameterized by a weight vector $\mathbf{w}$, i.e., $f(\mathbf{x}) = \text{sign}(\mathbf{w} \cdot \mathbf{x})$.

4.2.2.1 Building a Global Model by Online Learning

The first step of the collaborative online multitask learning builds a global classification model to exploit the commonality among tasks. We adopt the online passive aggressive (PA) framework [47] to build a global model using data collected from *all* users at round t , that is

$$f_t(\mathbf{x}) = \text{sign}(\mathbf{u}_t \cdot \mathbf{x})$$

where $\mathbf{u}_t \in \mathbb{R}^d$ is the weight vector of the global model learned at round t .

Specifically, at round t , the algorithm uses the latest training instance $(\mathbf{x}_t, y_t)$ to update the classification model as follows

$$\mathbf{u}_{t+1} = \underset{\mathbf{u} \in \mathbb{R}^d}{\text{argmin}} \quad \frac{1}{2} \|\mathbf{u} - \mathbf{u}_t\|^2 + C\xi \quad (4.1a)$$

$$\text{s.t. } \ell(\mathbf{u}; (\mathbf{x}_t, y_t)) \leq \xi \quad (4.1b)$$

$$\xi \geq 0 \quad (4.1c)$$

where C is a positive parameter controlling the influence of the slack variable ξ on the objective function, and ℓ is the hinge loss function defined as

$$\ell(\mathbf{u}; (\mathbf{x}_t, y_t)) = \max(0, 1 - y_t \mathbf{u} \cdot \mathbf{x}_t)$$

The above formulation aims to achieve two objectives: (1) variation of the new weight vector $\mathbf{u}_{t+1}$ from the previous weight vector $\mathbf{u}_t$ should be as small as possible, and (2) the new weight vector should correctly classify the current example $\mathbf{x}_t$ with a sufficiently large margin. By doing so, it maintains a trade-off between the amount of progress made on each round and information retained from previous rounds. The closed-form solution of the optimization problem (4.1) is

$$\mathbf{u}_{t+1} = \mathbf{u}_t + \tau_t y_t \mathbf{x}_t$$

where τ_t is given by $\tau_t = \min\{C, \frac{\ell_t}{\|\mathbf{x}_t\|^2}\}$. The proof can be found in [47].

4.2.2.2 Learning the Collaborative Models

The critical step of our collaborative online multitask learning is to apply the existing global model to collaboratively learn each of the K individual user models. Using the same PA formulation, the goal is to learn a classification model for the k -th user as

$$f_t^{(k)}(\mathbf{x}) = \text{sign}(\mathbf{w}_t^{(k)} \cdot \mathbf{x})$$

where $\mathbf{w}_t^{(k)} \in \mathbb{R}^d$ is the weight vector of the k -th user's collaborative model learned at round t . For simplicity, we use $\mathbf{w}_t$ to denote $\mathbf{w}_t^{(k)}$ henceforth.

The next step is to use the shared information learned by the global model to enhance the individual learning model. We formulate the collaborative learning model as a convex optimization problem that minimizes the deviation of the new weight vector from the prior collaborative one and the global one, as follows

$$\mathbf{w}_{t+1} = \underset{\mathbf{w} \in \mathbb{R}^d}{\text{argmin}} \quad \frac{\eta_1}{2} \|\mathbf{w} - \mathbf{w}_t\|^2 + \frac{\eta_2}{2} \|\mathbf{w} - \mathbf{u}_t\|^2 + C\xi \quad (4.2a)$$

$$\text{s.t. } \ell(\mathbf{w}; (\mathbf{x}_t, y_t)) \leq \xi \quad (4.2b)$$

$$\xi \geq 0 \quad (4.2c)$$

where η_1 and η_2 are two parameters that balance the tradeoff between the global model $\mathbf{u}$ and the collaborative model $\mathbf{w}$, and parameter $C \geq 0$ controls the influence of the slack variable ξ on the objective function.

The above formulation aims to achieve a balance between the global and individual models, i.e., in spite of its uniqueness, each individual also shares some commonality with other members in the group. It coherently combines the collaborative model with the global one. In particular, if we set $\eta_2 = 0$, the optimization reduces to the approach of learning an individual classification model without engaging the global model; if we set $\eta_1 = 0$, it reduces to the global model. Accordingly, we can fine tune the contribution of each model by setting appropriate parameters.

Applying the Lagrangian multiplier technique, the update rule for optimization problem (5.4) can be derived as

$$\mathbf{w}_{t+1} = \frac{\eta_1 \mathbf{w}_t + \eta_2 \mathbf{u}_t + \tau y_t \mathbf{x}_t}{\eta_1 + \eta_2} \quad (4.3)$$

where τ is given by

$$\tau = \min \left\{ C, \frac{\eta_1 + \eta_2 - y_t(\eta_1 \mathbf{w}_t + \eta_2 \mathbf{u}_t) \cdot \mathbf{x}_t}{\|\mathbf{x}_t\|^2} \right\}$$

The pseudo code for the proposed collaborative online multitask learning is given in Algorithm 3.

4.2.3 Extending Confidence Weighted Learning

A current trend in online learning research is to use parameter confidence information to guide online learning. Confidence weighted learning, proposed by Crammer et al. [60, 49, 50], models the linear classifier hypotheses uncertainty with a multivariate Gaussian distribution over weight vectors, which is then used to control the direction and scale of parameter updates. Conceptually, to classify an instance $\mathbf{x}$, a confidence weighted classifier draws a parameter vector $\mathbf{w} \sim N(\mu, \Sigma)$ and predicts the label according to $\text{sign}(\mathbf{w} \cdot \mathbf{x})$. In practice, however, the average weight vector $E(\mathbf{w}) = \mu$ is used to make

Algorithm 3: Collaborative online multitask learning

Input: training data of K users $(\mathbf{x}^k, y^k)$, $k = 1, \dots, K$, parameters $\eta_1 \geq 0$, $\eta_2 \geq 0$,
 $C \geq 0$
 Initialize $\mathbf{w}_0^k \leftarrow \mathbf{0}$, $\mathbf{u}_0 \leftarrow \mathbf{0}$;
for $t \leftarrow 1$ **to** T **do**
 // Update the collaborative models
 for $k \leftarrow 1$ **to** K **do**
 adjust parameter η ;
 receive global model weight vector $\mathbf{u}_t$;
 receive training instance $(\mathbf{x}_t^k, y_t^k)$;
 set loss $\ell(\mathbf{w}^k; (\mathbf{x}_t^k, y_t^k)) = \max(0, 1 - y_t^k \mathbf{w}^k \cdot \mathbf{x}_t^k)$;
 if $\ell(\mathbf{w}^k; (\mathbf{x}_t^k, y_t^k)) = 0$ **then**
 $\tau = 0$
 else
 $\tau = \min\{C, \frac{1 - y_t^k (\eta \mathbf{w}_t^k + (1 - \eta) \mathbf{u}_t) \cdot \mathbf{x}_t^k}{\|\mathbf{x}_t^k\|^2}\}$
 end
 update $\mathbf{w}_{t+1}^k = \eta \mathbf{w}_t^k + (1 - \eta) \mathbf{u}_t + \tau y_t^k \mathbf{x}_t^k$;
 end
 // Update the global model
 for $k \leftarrow 1$ **to** K **do**
 receive training instance $(\mathbf{x}_t^k, y_t^k)$;
 set loss $\ell(\mathbf{u}_t; (\mathbf{x}_t^k, y_t^k)) = \max(0, 1 - y_t^k \mathbf{u}_t \cdot \mathbf{x}_t^k)$;
 if $\ell(\mathbf{u}_t; (\mathbf{x}_t^k, y_t^k)) > 0$ **then**
 $\sigma = \min\{C, \frac{1 - y_t^k \mathbf{u}_t \cdot \mathbf{x}_t^k}{\|\mathbf{x}_t^k\|^2}\}$;
 update $\mathbf{u}_{t+1} = \mathbf{u}_t + \sigma y_t^k \mathbf{x}_t^k$;
 end
 end
end

the prediction. Confidence weighted learning algorithms have been shown to perform well on many tasks. In this section, we extend the proposed collaborative online multitask learning with the confidence weighted hypothesis.

We make use of the state-of-the-art confidence weighted learning algorithm — Adaptive Regularization of Weights (AROW) [50] in our work. It solves the following unconstrained objective function on each round

$$\operatorname{argmin}_{\mu \in \mathbb{R}^d, \Sigma \in \mathbb{R}^{d \times d}} \mathrm{D}_{\mathrm{KL}}(\mathcal{N}(\mu, \Sigma) \parallel \mathcal{N}(\mu_t, \Sigma_t)) + \frac{1}{2r} \ell^2(\mu; (\mathbf{x}_t, y_t)) + \frac{1}{2r} \mathbf{x}_t^T \Sigma \mathbf{x}_t$$

where the first term ensures that the updated distribution is similar to the current distribution $\mathcal{N}(\mu_t, \Sigma_t)$ in the Kullback-Leibler divergence sense. The second term $\ell^2(\mu; (\mathbf{x}_t, y_t) = (\max(0, 1 - y_t \mu \cdot \mathbf{x}_t))^2$ is the squared hinge loss suffered from using the weight vector μ to predict the output for input $\mathbf{x}_t$ when the true label is y_t . The third term is related to a probabilistic constraint used in confidence weighted learning, i.e., a classifier drawn from the updated distribution should classify the example correctly with a high probability (see [60, 50] for details).

As in collaborative online multitask learning, we maintain a global classification model parameterized by a Gaussian distribution over weight vectors with mean $\tilde{\mu}$ and covariance $\tilde{\Sigma}$, and use the global model information together with each user data to train the collaborative models. The weight distribution is updated by minimizing the KL divergence between the new and old weight distributions, together with the KL divergence between the weight distributions of individual and global models. On each round, the updated collaborative model parameters (μ_t, Σ_t) are set to the result of the following optimization problem

$$\begin{aligned} \operatorname{argmin}_{\mu \in \mathbb{R}^d, \Sigma \in \mathbb{R}^{d \times d}} & \eta_1 \mathrm{D}_{\mathrm{KL}}(\mathcal{N}(\mu, \Sigma) \parallel \mathcal{N}(\mu_t, \Sigma_t)) + \frac{1}{2r} \mathbf{x}_t^T \Sigma \mathbf{x}_t \\ & + \eta_2 \mathrm{D}_{\mathrm{KL}}(\mathcal{N}(\mu, \Sigma) \parallel \mathcal{N}(\tilde{\mu}_t, \tilde{\Sigma}_t)) + \frac{1}{2r} \ell^2(\mu; (\mathbf{x}_t, y_t)) \end{aligned}$$

where η_1 , η_2 and r are tradeoff parameters.

Following Crammer et al. [50], we use a correspondingly conservative update for the confidence parameter Σ , updating it only when μ changes. Assuming $1 - y_t \mu \cdot \mathbf{x}_t > 0$, taking the derivative of the objective function with respect to μ and Σ , respectively, and setting them to zero, we have the following update rules

$$\begin{aligned} \mu_{t+1} &= MN + \frac{1 - y_t MN \mathbf{x}_t}{r + \mathbf{x}_t^T M \mathbf{x}_t} N y_t \mathbf{x}_t \\ \Sigma_{t+1} &= (\eta_1 + \eta_2) \left(M - \frac{M \mathbf{x}_t \mathbf{x}_t^T M}{r + \mathbf{x}_t^T M \mathbf{x}_t} \right) \end{aligned}$$

where M and N are given by

$$\begin{aligned} M &= (\eta_1 \Sigma_t^{-1} + \eta_2 \tilde{\Sigma}_t^{-1})^{-1} \\ N &= \eta_1 \Sigma_t^{-1} \mu_t + \eta_2 \tilde{\Sigma}_t^{-1} \tilde{\mu}_t \end{aligned}$$

The pseudo code for confidence weighted collaborative online multitask learning is listed in Algorithm 4.

Algorithm 4: Confidence-weighted collaborative online multitask learning

Input: training data of K users $(\mathbf{x}^k, y^k)$, parameters $\eta_1 \geq 0, \eta_2 \geq 0, C < 0$
 Initialize $\mu_0^k \leftarrow \mathbf{0}, \Sigma_0^k \leftarrow I, \tilde{\mu}_0 \leftarrow \mathbf{0}, \tilde{\Sigma}_0 \leftarrow I$;
for $t \leftarrow 1$ **to** T **do**
 // Update the collaborative models
 for $k \leftarrow 1$ **to** K **do**
 receive global model parameters $(\tilde{\mu}_t, \tilde{\Sigma}_t)$;
 receive training instance $(\mathbf{x}_t^k, y_t^k)$;
 compute margin $m_t^k = \mu_{t-1}^k \cdot \mathbf{x}_t^k$;
 if $m_t^k y_t^k < 1$ **then**
 $\mu_{t+1}^k = MN + \frac{1-y_t^k MN \mathbf{x}_t^k}{r+\mathbf{x}_t^{kT} M \mathbf{x}_t^k} N y_t^k \mathbf{x}_t^k$ $\Sigma_{t+1}^k = (\eta_1 + \eta_2)(M - \frac{M \mathbf{x}_t^k \mathbf{x}_t^{kT} M}{r+\mathbf{x}_t^{kT} M \mathbf{x}_t^k})$
 end
 end
 // Update the global model
 for $k \leftarrow 1$ **to** K **do**
 receive training instance $(\mathbf{x}_t^k, y_t^k)$;
 compute margin $\tilde{m}_t = \tilde{\mu}_{t-1} \cdot \mathbf{x}_t$;
 if $\tilde{m}_t y_t < 1$ **then**
 $\tilde{\mu}_{t+1} = \tilde{\mu}_t + \frac{1-y_t \mathbf{x}_t^T \tilde{\mu}_t}{r+\mathbf{x}_t^T \tilde{\Sigma}_t \mathbf{x}_t} \tilde{\Sigma}_t y_t \mathbf{x}_t$ $\tilde{\Sigma}_{t+1} = \tilde{\Sigma}_t - \frac{\tilde{\Sigma}_t \mathbf{x}_t \mathbf{x}_t^T \tilde{\Sigma}_t}{r+\mathbf{x}_t^T \tilde{\Sigma}_t \mathbf{x}_t}$;
 end
 end
 end
 end

4.2.4 Time and Space Complexity

It is clear that the proposed algorithm in general enjoys a linear time complexity with respect to the number of instances and dimensions, which is not different from typical online learners. In terms of space complexity, for a group of K tasks, in addition to a set of K collaborative models, it only needs to maintain one extra global model. For confidence weighted collaborative online multitask learning, we can use diagonal matrices instead of full matrices for the covariance Σ to achieve significant computation costs and space savings. It is also feasible to exploit the sparsity of data via sparse matrix data structures to reduce the memory cost in practice. Therefore, our algorithm is both efficient and scalable for large-scale online learning tasks.

4.3 Experimental Results

We evaluate the performance of our algorithm on a synthetic dataset and three real-life datasets. We start by introducing our experimental setup, followed by discussions on the results.

4.3.1 Benchmark Setup

We compare our COML algorithm with two batch learning methods (LIBSVM [37] with linear kernel and LIBLINEAR [63]) and four online learning algorithms (online multitask learning [59], hereafter OML, ROMMA [88], PA [47], and AROW [50]). Apart from testing various values of the penalty parameter C for the two batch learning methods, all other parameters were set to default values. Batch learning methods were modified to handle online data by periodically retraining them with the most recent 50 samples after every new sample arrives. To further examine the effectiveness of learning multiple related tasks together, we compare our method with a few variations of the PA and AROW algorithms as described below.

- **Global Model** It learns a single classification model from *all* tasks' data by applying the PA/AROW algorithm. At each online learning round, the algorithm receives a training sample from each task, and uses that sample to update its weight vector.
- **Personal Model** It employs the PA/AROW algorithm to train a personal classification model for *each* task only using its own data. In other words, every task is associated with a personalized classification model.
- **Simple Model** It simply switches between the Global and Personal models according to their cumulative error counts in previous online learning rounds. In particular, at each round, it sets its weight vector to that of the best model (Global or Personal), i.e., one with the least cumulative errors to-date. Benchmarking against this method is important as it will show whether the proposed COML algorithm is more effective than a naive combination.

We adopt the *cumulative error rate*, i.e., the ratio of the number of mistakes made by the online learning algorithm over total number of samples received to-date as a metric for comparing different algorithms. Despite its extensive usage in online learning studies, the cumulative error rate is not suitable for evaluating performances on class-imbalanced datasets. This is because for a highly imbalanced dataset, it is possible to deploy a trivial classifier (i.e., blanket prediction of the majority class) that has low error rates but actually is of little use. We thus also report the F1-measure, which is the harmonic mean of precision and recall. It is typically harder for a classifier to achieve a good F1-measure compared to the error rate on an imbalanced dataset. For simplicity, we fix the parameter C and r to 1 for all PA/AROW variations and our algorithm. We also fix η_2 to 1 and only choose η_1 from a small range of values.

Table 4.1: Cumulative error rate (%) on the synthetic dataset.

	Task1	Task2	Task3	Task4	Task5	Average
PA-Global	19.00	21.50	24.00	17.00	27.50	21.80
PA-Personal	16.50	25.50	19.50	15.00	20.50	19.40
PA-Simple	17.00	23.50	19.50	16.00	20.50	19.30
COML	11.50	17.50	15.00	10.00	19.50	14.70
AROW-Global	12.50	14.50	14.00	8.50	25.00	14.90
AROW-Personal	13.50	22.00	15.00	11.00	20.00	16.30
AROW-Simple	13.00	16.00	15.50	10.00	21.50	15.20
CW-COML	10.50	16.50	12.00	9.00	20.50	13.70

4.3.2 Synthetic Dataset

We used a synthetic dataset designed by Sheldon [111] to show that solving multiple related tasks jointly outperforms the solution that treats each task in isolation. The goal in the synthetic dataset is to discriminate two classes (positive class and negative class) in a two-dimensional plane with non-linear decision boundaries as shown in Figure 4.2. Denote by $\mathbf{x} = (x_1, x_2)$ a point in the two-dimensional space. The basic classification boundaries are generated according to the rule $g(\mathbf{x}; \mathbf{a}) = \text{sign}(x_2 - h(x_1; \mathbf{a}))$, where $h(x; \mathbf{a})$ is a simple family of nonlinear functions consisting of the first few terms of an arbitrary Fourier series defined as $h(x; \mathbf{a}) = a_1 \sin(x - a_0) + a_2 \sin(2(x - a_0)) + a_3 \cos(x - a_0) + a_4 \cos(2(x - a_0))$. Rotation is also applied to the decision boundaries. Let R_θ be the

Figure 4.2: Illustration of the synthetic dataset. Two classes are represented each by crosses and circles. Solid lines denote the COML classification boundaries after observing all training samples.

Figure 4.3: Cumulative error rate (vertical axis) versus number of training samples observed (horizontal axis) on the synthetic dataset.

operator that rotates a vector by θ radians in a counter-clockwise direction about the origin. The final family of classifier is $f(\mathbf{x}; \mathbf{a}, \theta) = g(R_\theta \mathbf{x}; \mathbf{a})$ with θ as an additional parameter.

A total of five related tasks are involved in the experiment. Task parameters are generated via a random walk in parameter space with Gaussian increments [111]. Initial values are $\mathbf{a}^{(1)} = (0, 1, 1, 1, 1)$ and $\theta^{(1)} = 0$. For $t = 2, \dots, 5$, $\mathbf{a}^{(t)} = \mathbf{a}^{(t-1)} + \epsilon_t$, $\epsilon_t \sim N(0, \sigma^2 I)$; $\theta^{(t)} = \theta^{(t-1)} + \delta_t$, $\delta_t \sim N(0, \sigma^2(\pi/4)^2)$. The parameter σ^2 controls step sizes and hence task similarity, and is set to 0.5. A training sample of size 200 is generated for each task by choosing 200 inputs $\mathbf{x}$ uniformly at random from the square $x_1, x_2 \in [-3, 3]$, and then labeling them according to $f(\mathbf{x}; \mathbf{a}, \theta)$. To fit the online learning scenario, generated samples are supplied to learning algorithms one by one. Since the problem is not linearly-separable, we added seven additional features/dimensions, which are derived from the original x_1 and x_2 via a mapping $(x_1, x_2) \mapsto (x_1, x_2, x_1x_2, x_1^2, x_2^2, x_1^3, x_2^3, x_1x_2^2, x_1^2x_2)$.

Table 4.1 summarizes the experimental result on the synthetic dataset. Figure 4.3 depicts the detailed evolution of cumulative error rate for the entire online learning

process of every task. From these results, we can make the following observations.

First, PA-Global performed the worst among all the PA algorithms because it cannot adapt to the individual tasks. However, when the data covariance information is incorporated, AROW-Global is actually the second best performer. This shows that additional data information is helpful.

Second, PA-Simple performed slightly better than PA-Global due to its greedy selection approach. It actually performed similarly as PA-Personal, which means that most of the time it selected the personal model. When additional information is added via confidence weighing, AROW-Simple again performed better than the AROW-Personal model, and is only marginally worse off than AROW-Global.

Finally, we see that COML has the overall lowest error rate of 14.70%, which is almost 5% better than the runner-up, PA-Simple. However, when additional covariance information is incorporated, CW-COML still leads but is only 1.2% better than the runner-up, AROW-Global. In general, the proposed collaborative online multitask learning algorithms achieve lower cumulative error rates for all of the five tasks, which show that they are effective in learning problems with a common shared representation across multiple related tasks. The failure of the Global model is consistent with our intuition that learning related tasks via a single model is inappropriate as it ignores the individual task characteristic. A naive combination of global and individual models is also ineffective, as indicated by the suboptimal performance of the Simple method.

4.3.3 Spam Email Filtering

We apply online collaborative multitask learning to construct effective personalized spam email filters. The task is to classify each new incoming email message into two categories: *legitimate* or *spam*. We use a dataset hosted by the Internet Content Filtering Group¹. The dataset contains 7068 emails collected from mailboxes of four users (denoted by *pu1*, *pu2*, *pu3*, and *pua*). Strictly speaking, the set of all emails received by a user is not generated per se by that specific user. However, the characteristic of each user's email can be said to match his or her interest, whatever that may be. Each email

¹<http://labs-repos.iit.demokritos.gr/skel/i-config/>

Table 4.2: Cumulative error rate and F1-measure (%) on the spam email dataset

	pu1			pu2		
	Error	Legit F1	Spam F1	Error	Legit F1	Spam F1
SVM	8.99	92.05	89.66	7.32	95.58	78.69
LINEAR	6.79	93.99	92.19	7.46	95.53	77.25
OML	4.86	95.68	94.44	7.61	95.30	80.15
ROMMA	11.83	89.20	86.85	14.08	90.22	70.89
PA-Global	7.16	93.65	91.81	8.45	94.81	77.27
PA-Personal	5.50	95.13	93.67	10.85	93.30	71.59
PA-Simple	5.60	95.04	93.59	8.73	94.66	75.97
COML	4.13	96.33	95.28	6.62	95.96	81.71
AROW-Global	5.60	95.03	93.60	7.61	95.30	80.00
AROW-Personal	4.50	96.00	94.87	7.32	95.47	80.88
AROW-Simple	4.59	95.92	94.76	7.46	95.38	80.59
CW-COML	4.50	96.00	94.87	6.06	96.28	83.77
	pu3			pua		
	Error	Legit F1	Spam F1	Error	Legit F1	Spam F1
SVM	8.99	92.31	89.20	9.75	89.97	90.52
LINEAR	5.04	95.51	94.26	6.67	93.21	93.44
OML	3.25	97.11	96.30	6.76	93.24	93.24
ROMMA	7.65	93.05	91.46	14.14	85.69	86.04
PA-Global	4.07	96.38	95.36	9.39	90.75	90.64
PA-Personal	3.73	96.69	95.73	8.43	91.59	91.55
PA-Simple	3.83	96.60	95.63	8.87	91.15	91.12
COML	2.98	97.36	96.59	5.88	94.14	94.10
AROW-Global	3.83	96.59	95.64	8.08	91.90	91.94
AROW-Personal	3.27	97.09	96.27	6.76	93.23	93.25
AROW-Simple	4.80	97.01	95.23	6.85	93.15	93.16
CW-COML	2.40	97.87	97.26	5.88	94.15	94.09

entry is converted to a word document vector using the TF-IDF (term frequency-inverse document frequency) representation.

Since the email dataset has no time stamp, each user’s email was shuffled into a random sequence. The cumulative error rate and F1-measure results are listed in Table 4.2. We also report each algorithm’s run-time, i.e., the time consumed by both training and test phase during the complete online learning process in Table 4.3. From these results, we can make several observations.

First, the proposed collaborative online multitask learning methods (COML and CW-

COML) consistently beat the other four online learners in terms of error rate and F1-measure. In particular, in accordance to the results from the synthetic dataset, learning tasks collaboratively outperforms the baselines Global model, Personal model, and a simple combination of either.

Second, the performance of the proposed collaborative online multitask learning methods is better than that of the two batch learning algorithms (SVM and LINEAR). It should be noted that compared to online learners who update models based only on the current sample, batch learning methods have the advantage of keeping a substantial amount of recent training samples, at the cost of storage space and higher complexity. In fact, the proposed COML algorithm is more efficient than batch incremental methods, e.g., it is 100 times faster than batch SVM as shown in Table 4.3 (1.02 secs versus 109.33 secs). COML does not store recent training samples. It only uses the current training sample and a simple rule to update the model. In contrast, batch learning algorithms need to keep a certain number of recent training samples in memory, leading to extra burden on storage and complexity. What's more, both SVM and LINEAR need to solve an optimization problem in an iterative manner. SVM has to additionally compute the kernel matrix. For practical applications involving hundreds of millions of users and features, the batch learning algorithms are no longer feasible, while online learners remain highly efficient and scalable.

Table 4.3 shows that COML is slightly slower than the original PA algorithm. This is expected since COML has to update an additional global model. However, for a group of users, only one group model is needed. The extra computational cost is trivial compared to the combined cost to update every user model. Therefore, the proposed COML algorithm is efficient and applicable for solving large-scale problems.

Table 4.3: Run-time (in seconds) for each algorithm

	COML	PA-G	PA-P	OML	ROMMA	SVM	LINEAR
Spam	1.02	0.67	0.70	52.28	0.53	109.33	7.84
MHC-I	1.28	0.73	0.79	65.43	1.02	113.91	12.55
Twitter	9.76	6.75	7.10	-	10.24	44.69	9.73

Table 4.4: MHC-I dataset result showing average cumulative error rate, F1-measure (%) and their standard deviation (in brackets) of 12 tasks

	Error Rate	Positive F1	Negative F1
SVM	34.31 (0.34)	36.40 (0.79)	72.77 (0.30)
LINEAR	44.87 (0.29)	46.52 (0.27)	60.50 (0.35)
OML	41.56 (0.20)	51.13 (0.28)	63.08 (0.25)
ROMMA	46.88 (0.61)	48.71 (0.60)	55.97 (0.69)
PA-Global	44.80 (0.38)	45.44 (0.34)	61.28 (0.49)
PA-Personal	41.62 (0.21)	51.08 (0.28)	63.02 (0.29)
PA-Simple	42.01 (0.31)	50.07 (0.34)	62.92 (0.34)
COML	37.62 (0.24)	55.45 (0.25)	66.82 (0.31)
AROW-Global	40.67 (0.30)	50.36 (0.33)	64.77 (0.25)
AROW-Personal	41.32 (0.26)	50.95 (0.27)	63.43 (0.33)
AROW-Simple	40.22 (0.43)	51.74 (0.59)	64.70 (0.41)
CW-COML	39.31 (0.21)	53.75 (0.18)	65.12 (0.33)

4.3.4 MHC-I Binding Prediction

Computational methods are widely used in bioinformatics to build models to infer properties from biological data. In this experiment, we evaluate several methods to predict peptide binding to human MHC (major histocompatibility complex) class I molecules. It is known that peptides binding to MHC-I molecules plays a crucial role in the vertebrate immune system [102]. The prediction of such binding has valuable application in vaccine designs, the diagnosis and treatment of cancer, etc. Recent work has demonstrated that there exists common information between related molecules (alleles) and such information can be leveraged to improve the peptide MHC-I binding prediction [78, 39].

We use a subset of the binary labeled MHC-I dataset². The task is to predict whether a peptide binds to a certain human MHC-I molecule or not, i.e., *binder* or *non-binder*. The data consists of peptide sequences for 12 human MHC-I alleles. Each allele is indicated by a prefix letter followed by four digits. In total, there are 18664 samples (A0201=3793, A0202=1363, A0203=1363, A0206=1358, A0301=2633, A2402=763, A2902=535, A3002=415, A3101=2435, A3301=1179, A6801=1183, A6802=1644). On average, 33% of sample is labeled as binder (positive class). The peptide sequences are

²<http://ailab.cs.iastate.edu/red/mhci.html>

represented by strings of length 9 over the alphabet of 20 amino acids. To incorporate string features, we apply the bigram amino acid encoding to the protein sequence. The bigram features are a pair of values (v_i, c_i) , where v_i is the i -th feature and c_i denotes the number of occurrences of this feature in the sequence for $i = 1, \dots, 20^2$. For example, the protein sequence “ALAKAAAI” has a bigram feature vector of $\{(AL, 1), (LA, 1), (AK, 1), (KA, 1), (AA, 3), (AI, 1)\}$.

The experiment was conducted over 10 random permutations of the original dataset, and parameters were chosen from a separate validation set. We report the average cumulative error rate and F1-measure of 12 tasks in Table 4.4. To make a clear comparison between the proposed COML and PA baselines, we show the variation of their cumulative error rates along the entire online learning process averaged over the 10 runs in Figure 4.4.

From these results, we first observe that the permutations of dataset have little influence on the performance of each method, as indicated by the small standard deviation values in Table 4.5. Overall, SVM has the lowest error rate, which is achieved at the expense of having the worst F1-measure on the positive class. Note that the majority of the dataset is negative, thus predicting more examples as the majority class decreases the overall error rate, but also degrades the accuracy of the minority positive class. In contrast, the collaborative online multitask learning (COML) and its confidence weighted learning counterpart (CW-COML) outperformed all online competitors in terms of error rate and F1 measures of both classes.

We note that confidence weighted online learning (AROW) actually performed slightly worse than the vanilla PA algorithms. However, among the four confidence weighted models, learning related tasks jointly still outperforms learning the tasks individually, as shown by the improvement of CW-COML model over the AROW-Global, AROW-Personal and AROW-Simple models.

4.3.5 Micro-blog Sentiment Analysis

With the growing popularity of micro-blogs like *Twitter*³ comes the demand to understand their users. We present an interesting problem, namely the micro-blog sentiment

³<http://twitter.com>

Table 4.5: Details of *Twitter* sentiment dataset

User Name	# Instance	# Pos Sentiment	# Neg Sentiment	% Sentiment
JamesKysonLee	1187	152	48	16.85
JimGaffigan	1161	98	280	32.56
lcapplegate	1185	134	196	27.85
elizadushku	1420	242	129	26.13
jessetyler	1040	131	168	28.75
peterfacinelli	1138	233	106	29.79
Total / Average	7131	990	927	26.88

analysis, which aims to automatically classify each micro-blog post into three categories: *positive sentiment*, *negative sentiment*, and *non-sentimental*. This problem is challenging because a micro-blog post is often very short and each person may have his/her unique way of expressing sentiments. Moreover, the proportion of sentimental posts is typically very small, and varies across individuals. Ideally, a personalized classifier should be created for each micro-blogger. However, there is a dearth of training data for each user, making the personalized sentiment model vastly inaccurate unless the model has been trained over hundreds of micro-blogs.

A post on *Twitter* is called a tweet. We crawled 7131 tweets written by six influential users according to *wefollow.com*⁴. The latest tweet in our dataset was published in May, 2010, and the oldest one was in February, 2008. Each tweet was converted into a word vector using the TF-IDF representation. A human annotator was asked to label tweets as one of three categories: positive sentiment (class 1), negative sentiment (class 2), and non-sentimental (class 3). Table 4.5 summarizes the labeling result. We can see that on average, 20-30% of a user tweets are sentimental (belonging to positive or negative sentiment). This result matches our conjecture that sentimental tweets are generally a minority among *Twitter* users. The experimental results are reported in Table 4.6. It can be seen that two of the models trained with PA and AROW — Global and Personal, match each other in strength, but loose out to the collaborative models. This result validates our previous concern that simply learning a single global model from all users' data is insufficient for solving the micro-blog sentiment classification task, since each user expresses emotions differently. The simple combination of Global model and Personal

⁴<http://wefollow.com>

Figure 4.4: Cumulative error rate on the MHC-I dataset along the entire online learning process.

model — Simple model is able to approach the best model between Global and Personal, but fails to outperform the best one. This is because the “simple” scheme selects between the global and personal models by comparing their *historical* cumulative error counts. It is possible for a model with few mistakes in the earlier learning rounds to misclassify an incoming sample. Similarly, a model with large historical cumulative errors may score a lucky hit on that sample. This makes sticking to the first model an unwise choice.

Table 4.6 shows that SVM achieves better error rate and F1-measure on the majority class (non-sentimental tweet) than the other competitors. On the other hand, by examining the minority classes (positive or negative sentiment tweet), we find that the F1-measure performance of SVM is worse than the others. This is because SVM classifies

most examples as non-sentimental, leading to high accuracy on the majority class at the expense of the minority class. We further find that the F1-measure performance of the collaborative online multitask learning methods (COML and CW-COML) is better than the other competitors for most cases, again verifying the effectiveness of the proposed technique for online micro-blog sentiment classification. However, despite the improvements, the overall F1-measure of both the positive and negative class tweets still have a long way to go, as both are extremely low at around 25%.

4.4 Discussion

In this chapter, we proposed a collaborative online multitask learning method that is able to take advantage of individual and global models to achieve an overall improvement in classification performance. We showed that it is able to outperform both the global and personal models by coherently integrating them in our unified collaborative learning framework. The experimental results demonstrate that our algorithms are both effective and efficient for three real-life applications, including online spam email filtering, MHC-I binding prediction, and micro-blog sentiment classification.

Although the collaborative online multitask learning algorithm was firstly designed to solve the UGC classification problem, it has potential applications outside of the domains studied here. We hope to be able to extend our experiments to a more substantial size dataset and also to more applications. Our methods assume uniform relations across tasks. However, it is more reasonable to take into account the degree of relatedness among tasks. How to incorporate hierarchies and clusters of tasks is also worthy of further study. In conclusion, our collaborative online multitask learning method is a significant first step towards a more effective online multitask classification approach.

Table 4.6: Cumulative error rate and F1-measure (%) on the *Twitter* sentiment dataset

(a) Cumulative error rate (%)

	Jam.	Jim.	lca.	eli.	jes.	pet.	Avg.
SVM	17.35	32.21	28.52	26.76	29.42	30.23	27.42
LINEAR	19.46	35.23	31.56	30.99	36.35	33.30	31.15
ROMMA	36.23	46.60	48.19	46.27	51.83	47.19	46.05
PA-G	48.44	50.04	53.16	53.73	56.44	51.85	52.28
PA-P	49.12	54.69	60.08	60.63	64.33	58.44	57.88
PA-S	48.44	50.90	53.50	54.01	57.21	51.67	52.62
COML	22.91	34.28	31.98	31.20	35.67	31.11	31.19
AROW-G	25.02	37.21	32.24	34.23	37.21	32.07	33.00
AROW-P	18.70	35.40	33.84	29.93	35.77	33.30	31.16
AROW-S	48.44	50.90	53.50	54.01	57.21	51.67	52.62
CW-COML	22.07	32.90	29.87	31.69	34.04	29.09	29.94

(b) Class 1 (positive sentiment) F1-measure (%)

	Jam.	Jim.	lca.	eli.	jes.	pet.	Avg.
SVM	11.21	2.60	1.17	2.06	3.77	18.87	6.61
LINEAR	20.62	9.71	6.97	16.35	7.80	28.63	15.01
ROMMA	18.25	9.98	11.20	21.32	14.31	26.94	17.00
PA-G	26.30	13.60	17.78	25.82	20.59	29.56	22.27
PA-P	26.89	13.39	11.19	27.98	16.48	30.71	21.11
PA-S	25.68	13.85	16.74	27.92	16.67	30.60	21.91
COML	27.27	14.22	17.07	25.30	19.87	31.13	22.48
AROW-G	24.43	13.60	16.59	21.10	17.83	28.01	20.26
AROW-P	25.50	13.11	10.69	25.31	16.22	31.46	20.38
AROW-S	28.44	13.90	15.50	24.01	17.21	31.67	20.12
CW-COML	22.02	16.23	18.43	21.60	15.71	29.42	20.57

(c) Class 2 (negative sentiment) F1-measure (%)

	Jam.	Jim.	lca.	eli.	jes.	pet.	Avg.
SVM	0.00	27.24	6.11	2.06	5.56	2.27	7.21
LINEAR	3.35	30.09	14.61	5.17	9.57	7.33	11.69
ROMMA	6.52	31.61	20.36	9.63	14.04	13.53	15.95
PA-G	3.88	30.55	21.58	12.80	15.63	17.90	17.05
PA-P	5.91	34.70	21.07	6.71	14.19	9.98	15.43
PA-S	5.98	27.44	20.08	5.99	12.44	11.85	13.96
COML	4.23	36.22	21.21	11.93	14.71	16.51	17.47
AROW-G	5.71	29.41	26.82	11.64	13.03	17.61	17.37
AROW-P	6.72	34.45	21.22	8.21	13.89	9.55	15.67
AROW-S	6.44	30.90	23.50	14.01	17.21	14.67	13.62
CW-COML	7.75	35.47	26.25	12.11	18.06	20.24	19.98

(d) Class 3 (non-sentimental) F1-measure (%)

	Jam.	Jim.	lca.	eli.	jes.	pet.	Avg.
SVM	90.38	79.20	83.15	84.45	82.45	81.34	83.50
LINEAR	88.87	75.84	80.36	80.79	76.81	77.90	80.09
ROMMA	76.35	63.22	63.88	66.05	60.45	63.14	65.52
PA-G	84.84	73.53	76.91	77.57	75.36	77.72	78.09
PA-P	89.15	74.51	78.15	81.31	76.10	79.46	77.53
PA-S	89.34	74.69	77.79	81.85	75.80	79.58	77.99
COML	86.32	75.61	79.07	79.74	76.11	79.32	79.08
AROW-G	84.87	73.69	78.44	77.50	75.05	78.42	77.99
AROW-P	89.22	74.83	77.88	80.97	76.40	77.48	79.46
AROW-S	88.44	70.90	73.50	74.01	77.21	71.67	72.62
CW-COML	87.00	76.93	80.44	79.58	77.62	80.79	80.40

Chapter 5

Collaborative Online Multitask Ranking

The preceding chapter presented a task-centric collaborative learning method for solving the classification problem, which aims to assign instances to one of several non-ordered categories. Ranking resembles classification as both assign one of several possible labels to a new instance, but differs in that there is an order relation between the labels. In this chapter, we examine the problem of learning to rank instances belonging to multiple related tasks from the task-centric collaborative perspective. We begin with a brief introduction about motivation, followed by methodology and experimental evaluation.

5.1 Motivation

Ranking or ordering objects according to their degrees of relevance, preference, or importance has been extensively studied in data mining, information retrieval, and related areas for decades. It has been applied widely to real-world applications. Document retrieval, collaborative filtering, sentiment analysis and web spam detection are some examples of natural ranking problems or can be solved with ranking methods [91].

An active line of research that has emerged recently is to leverage machine learning techniques to solve ranking problems. The so-called *learning-to-rank* methods examine how a ranking model can be automatically constructed from training data, which consist of a set of input objects and their corresponding ranks, such that the model can sort new objects in a “reasonable” way. Learning-to-rank has become a very hot research topic. A large number of learning-to-rank algorithms have been proposed [42, 51, 28, 30, 32, 40].

When applied to multiple related ranking problems, existing learning-to-rank approaches based on a single ranking model trained from data of multiple ranking problems may fail to capture the peculiarity of each ranking problem. On the other hand, training each model per task using task-specific data ignores the potential commonness shared among related ranking problems. As an example, consider the problem of recommending movies to multiple users. The recommendation should rely on the user’s own data, e.g., user profile, history of movie enjoyed in the past, etc. Yet it is also reasonable to take other similar users’ preferences into consideration. It is thus desirable to combine individual traits and group characteristics to achieve a result that is greater than the sum of its parts.

The problem of jointly solving several related learning tasks by leveraging the commonality among tasks has been studied in the machine learning community under the guise of multitask learning. Empirical findings have demonstrated the usefulness of leveraging information learned from one task to influence the learning of other tasks in presence of data from different but related problems [33, 61, 39]. Moreover, it has shown that combining information from several tasks can help when training data for each task is relatively small.

We study the problem of learning to rank instances belonging to multiple related tasks. Specifically, we adopt the online learning framework for its simplicity and effectiveness. We propose a highly efficient and scalable online ranking method, namely collaborative online ranking, which is designed to rank data generated by a group of tasks by combining group and individual characteristics. The basic idea is to run K online ranking models in parallel for K related tasks. Each model corresponds to a specific task, and is updated by using each task data and leveraging the information provided by models of fellow tasks. We formulate this idea into an optimization problem, enabling multiple tasks with varying amounts of data to collaborate and achieve overall better ranking performance in an online learning manner. We show that learning multiple related tasks collaboratively outperforms learning each task independently by comparing our algorithm with state-of-the-art online learners in real-world problems. We also compare the run-time complexity of all algorithms, showing that ours are not only effective, but also equally fast as other online learners.

5.2 Methodology

In the multitask online ranking setting, we are faced with K different but related ranking problems, also known as tasks. It is assumed that objects under investigation across all tasks are represented by feature vectors drawn from a single feature space. At the beginning of learning round t , the algorithm observes a set of K instances, one for each of the ranking problems. The algorithm predicts the rank for each of the instances it has observed using a ranking mechanism. The algorithm then receives a feedback indicating the correct rank, and may modify the ranking mechanism, presumably improving the chances of making an accurate prediction on subsequent rounds.

The idea of our multitask online ranking approach is to maintain several ranking models for each of the related tasks simultaneously, and to update them by leveraging both the task specific training data and the shared knowledge across all tasks. The ranking model we consider in this paper comprises a set of weights, which are used to form a linear combination of the instance feature values. We average the parameters across several ranking models into a single combined model to represent the shared knowledge across all tasks, and adopt the Perceptron ranking (PRank) update strategy [51] to learn these ranking models. In the following, we first define the ranking problem and review the PRank algorithm. We then describe the proposed collaborative online ranking algorithm formally, followed by a minor variant.

5.2.1 Preliminaries

The Perceptron Ranking (PRank) algorithm [51] forms the basis of our approach. PRank works on a collection of instance-rank pairs, i.e., $\{(\mathbf{x}_t, y_t)\}$, $t = 1, \dots, T$, where $\mathbf{x}_t \in \mathbb{R}^d$ is a d -dimensional feature vector representing the instance at round t and its corresponding rank y_t is an element from a finite set $\mathcal{Y} = \{1, \dots, k\}$ with a total order relation. The goal is to learn a mapping from instances to rankings: $\mathbb{R}^d \rightarrow \mathcal{Y}$.

To learn the mapping, PRank maintains a weight vector $\mathbf{w} \in \mathbb{R}^d$ and a vector $\mathbf{b}$ of increasing thresholds $b[1] \leq \dots \leq b[k-1] \leq b[k] = \infty^1$, which divide the real line into k segments, one for each possible rank. Given a new instance $\mathbf{x}$, PRank computes a score

¹We use $b[i]$ to denote the i -th component of the vector $\mathbf{b}$.

as the inner-product between $\mathbf{x}$ and $\mathbf{w}$. Then it locates the score on the real line and returns the predicted rank as indicated by the thresholds. Formally, the predicted rank is defined to be the index of the first (smallest) threshold $b[r]$ for which $\mathbf{w} \cdot \mathbf{x} < b[r]$, i.e., $\hat{y} = \min_{r \in \{1, \dots, k\}} \{r : \mathbf{w} \cdot \mathbf{x} < b[r]\}$. After prediction, the true (ground truth) rank $y \in \mathcal{Y}$ is revealed. An error is said to occur if the ground truth and the predicted rank mismatches, i.e., $y \neq \hat{y}$.

To make a correct prediction, it is required that $\mathbf{w} \cdot \mathbf{x} > b[r]$ for $r = 1, \dots, y - 1$ and $\mathbf{w} \cdot \mathbf{x} < b[r]$ for $r = y, \dots, k - 1$. By allocating $k - 1$ labels, denoted as $l[1], \dots, l[k - 1]$, to each interval represented by the threshold vector $\mathbf{b}$, we have the following².

$$l[r] = \begin{cases} 1 & \text{for } \mathbf{w} \cdot \mathbf{x} > b[r] \\ -1 & \text{for } \mathbf{w} \cdot \mathbf{x} < b[r] \end{cases}$$

In other words, a predicted rank is correct if $l[r](\mathbf{w} \cdot \mathbf{x} - b[r]) > 0$ for all $r = 1, \dots, k - 1$. If PRank makes a mistake, there is at least one threshold, indexed r , for which the value of $\mathbf{w} \cdot \mathbf{x}$ is on the wrong side of $b[r]$, i.e., $l[r](\mathbf{w} \cdot \mathbf{x} - b[r]) \leq 0$. To correct the mistake, PRank moves the values of $\mathbf{w} \cdot \mathbf{x}$ and $b[r]$ toward each other. To do so, PRank modifies the values of $b[r]$ for which $l[r](\mathbf{w} \cdot \mathbf{x} - b[r]) \leq 0$ by replacing them with $b[r] - l[r]$. The value of $\mathbf{w}$ is also replaced with $\mathbf{w} + (\sum l[r])\mathbf{x}$, where the sum is taken over all indices r for which there is a prediction error, i.e., $l[r](\mathbf{w} \cdot \mathbf{x} - b[r]) \leq 0$.

5.2.2 Collaborative Online Ranking

The proposed collaborative online ranking method shares the same settings as PRank, except that it processes several related ranking tasks in parallel. Formally speaking, there are K related tasks whose data arrive in sequence. For simplicity, we assume that every sequence contains up to T trials. Denote by $(\mathbf{x}_t^i, y_t^i)$ an instance-rank pair belonging to the i -th task at round t . We adopt a weight vector $\mathbf{w}_t^i \in \mathbb{R}^d$ and a set of k thresholds $b^i[1] \leq \dots \leq b^i[k - 1] \leq b^i[k] = \infty$ to represent the ranking rule of the i -th task. Our goal is to learn K ranking rules for each of the K tasks by using each task's training data, while leveraging the information provided by the other tasks.

As the tasks are related, we assume that there exists common knowledge shared among tasks, which can be used to enhance the learning of individual tasks. Using the

²We exclude the k -th threshold because $b[k]$ is fixed at infinity. It is clear that $\mathbf{w} \cdot \mathbf{x} < b[k]$.

same terminology from online collaborative classification, we call the ranking model for a specific task the *unique* model, and the ranking models of other tasks the *fellow* models. We use a *global* model with weight vector $\bar{\mathbf{w}}$ to represent the common knowledge. For the i -th task, one way to define the global model weight vector is to set it to an equally weighted sum of the fellow models' weight vectors, that is

$$\bar{\mathbf{w}} = \sum_{j \in K, j \neq i} \mathbf{w}^j / (K - 1) \quad (5.1)$$

We need to update the ranking rule when a prediction error occurs. Our aim is to achieve a balance between two factors. On one hand, the new ranking rule $\mathbf{w}_{t+1}^i$ should be as similar as possible to the previous one $\mathbf{w}_t^i$, which encompasses the knowledge of past training examples. On the other hand, we require the updated weight vector to be similar to the global model's weight vector, in order to incorporate common knowledge shared among tasks.

Specifically, assuming that the i -th task makes an error on round t , to update its weight vector, we first minimize the deviation of the new weight from the prior weight and the global weight, as follows

$$\mathbf{w}_{t+1}^i = \underset{\mathbf{w}^i \in \mathbb{R}^d}{\operatorname{argmin}} \frac{\eta_1}{2} \|\mathbf{w}^i - \mathbf{w}_t^i\|^2 + \frac{\eta_2}{2} \|\mathbf{w}^i - \bar{\mathbf{w}}_t\|^2 \quad (5.2)$$

where η_1 and η_2 are parameters that balance the tradeoff between the task specific model $\mathbf{w}_t^i$ and the global model $\bar{\mathbf{w}}_t$.

We then follow the PRank update rule by adding $(\sum_r l[r])\mathbf{x}^i$ to $\mathbf{w}^i$, where the summation applies to all indices r that causes the prediction error, i.e., $l[r](\mathbf{w}^i \cdot \mathbf{x}^i - b[r]) \leq 0$. Solving for problem (5.2), and adding $(\sum_r l[r])\mathbf{x}^i$ to the solution, we obtain the update rule for our collaborative online ranking algorithm. That is, on round t , we set the new weight vector of i -th task $\mathbf{w}_{t+1}^i$ as

$$\mathbf{w}_{t+1}^i = \frac{\eta_1 \mathbf{w}_t^i + \eta_2 \bar{\mathbf{w}}_t}{\eta_1 + \eta_2} + \sum_r l[r] \mathbf{x}_t^i \quad (5.3)$$

The above formulation aims to achieve a balance between the individual and global models, i.e., in spite of its uniqueness, each individual also shares some commonness with other tasks. It coherently combines the individual model with the global one. In

particular, if we set $\eta_2 = 0$, the optimization reduces to the approach of learning an individual PRank model without engaging the global model; if we set $\eta_1 = 0$, it reduces to the average of all models. Accordingly, we can fine tune the contribution of each part by setting appropriate parameters.

It is worth noting that the rationale of averaging the fellow models' weight vectors in equation (5.1) is similar to that described in the OAP-BPM (Online Aggregate PRank-Bayes Point Machine) [71]. OAP-BPM is an extension of PRank by approximating the Bayes point, which is the single hypothesis chosen from a fixed class of classifiers that achieves the minimum probability error [73]. To estimate the Bayes point in an online learning fashion, OAP-BPM trains K PRank models in parallel by randomly choosing to present a given sample to the i -th model only if $p_i = 1$, where $p_i, i = 1, \dots, K$ are independent Bernoulli random variables with $Pr\{p_i = 1\} = \tau$. After that, the weights and thresholds associated with the models are averaged to produce the final model. It has been proven that this improves the generalization performance compared to PRank. The proposed collaborative online learning algorithm trains several ranking model for each task using the task specific data. This is in fact analogous to generating diverse solutions by training each ranking model with a random sampling of the original dataset.

5.2.3 Collaborative Online Ranking with Large Margin

In this section, we present two variants of the proposed online collaborative ranking algorithm. The motivation for the variants stems from the work of Crammer et al. [53], who proposed a large margin version of PRank. Recall that PRank makes a prediction by positioning the value of $\mathbf{w} \cdot \mathbf{x}$ to an interval defined by thresholding a continuous real number line. As stated in [53], the difference between the value of $\mathbf{w} \cdot \mathbf{x}$ and the closest threshold plays a similar role to the notion of margin in classification problems. Even when PRank gives a correct prediction, the value of $\mathbf{w} \cdot \mathbf{x}$ might lie very close to one of the thresholds, thus leading to a trivial margin. To ensure a large enough margin after update, Crammer et al. [53] constrained the updated rule $(\mathbf{w}_{t+1}, \mathbf{b}_{t+1})$ to satisfy $\min_r \{(\mathbf{w}_{t+1} \cdot \mathbf{x}_{t+1} - b[r]_{t+1})l[r]_t\} \geq \beta$, where β is a positive constant denoting the margin width.

Following this idea, we modify the basic collaborative online ranking algorithm by setting the i -th task's new weight vector $\mathbf{w}_{t+1}^i$ and threshold vector $\mathbf{b}_{t+1}^i$ to be the solution to the following optimization problem, with the goal of achieving a margin of at least 1.

$$\min_{\mathbf{w}^i, \mathbf{b}^i} \quad \frac{\eta_1}{2} \|(\mathbf{w}^i, \mathbf{b}^i) - (\mathbf{w}_t^i, \mathbf{b}_t^i)\|^2 + \frac{\eta_2}{2} \|(\mathbf{w}^i, \mathbf{b}^i) - (\bar{\mathbf{w}}_t, \bar{\mathbf{b}}_t)\|^2 \quad (5.4a)$$

$$\text{s.t.} \quad (\mathbf{w}^i \cdot \mathbf{x}_t^i - b^i[r])l[r] \geq 1 \quad \text{for } r = 1, \dots, k-1 \quad (5.4b)$$

To solve the above optimization problem, we first write its Lagrangian function as

$$\mathcal{L} = \frac{\eta_1}{2} \|(\mathbf{w}^i, \mathbf{b}^i) - (\mathbf{w}_t^i, \mathbf{b}_t^i)\|^2 + \frac{\eta_2}{2} \|(\mathbf{w}^i, \mathbf{b}^i) - (\bar{\mathbf{w}}_t, \bar{\mathbf{b}}_t)\|^2 - \sum_{r=1}^{k-1} \tau_r ((\mathbf{w}^i \cdot \mathbf{x}_t^i - b^i[r])l[r] - 1) \quad (5.5)$$

where $\tau_r, r = 1, \dots, k-1$ are non-negative Lagrange multipliers.

Taking the derivative of equation (5.5) with respect to $\mathbf{w}^i$ and $b^i[r]$, and comparing them to zero respectively, we obtain

$$\begin{aligned} \mathbf{w}_{t+1}^i &= \frac{\eta_1 \mathbf{w}_t^i + \eta_2 \bar{\mathbf{w}}_t + (\sum_r \tau_r l[r]) \mathbf{x}_t^i}{\eta_1 + \eta_2} \\ b_{t+1}^i[r] &= \frac{\eta_1 b_t^i[r] + \eta_2 \bar{b}_t[r] - \tau_r l[r]}{\eta_1 + \eta_2} \end{aligned}$$

Plugging the value of $\mathbf{w}^i$ and $\mathbf{b}^i$ into the Lagrangian function (5.5), we get the following dual problem, which can be solved using standard optimization tools.

$$\min_{\tau} \quad \frac{1}{2} \|\mathbf{x}_t\|^2 \left(\sum_r \tau_r l_t[r] \right)^2 + \frac{1}{2} \sum_r \tau_r^2 + \quad (5.6a)$$

$$\begin{aligned} &\sum_r \tau_r (y_r (\eta_1 (\mathbf{w}_t^i \cdot \mathbf{x}_t - b_t^i[r]) + \eta_2 (\bar{\mathbf{w}}_t \cdot \mathbf{x}_t - \bar{b}_t[r])) - \eta_1 - \eta_2) \\ \text{s.t.} \quad &\tau_r \geq 0 \quad \text{for } r = 1, \dots, k-1 \end{aligned} \quad (5.6b)$$

Requiring every update to achieve a margin of at least 1 may be overly strict. To relax this constraint, we adopt the technique used by the soft-margin classifiers. Our goal is to achieve a margin of at least 1 as often as possible. If the algorithm attains a margin less than 1, it suffers a loss defined by the following loss function.

$$\ell(\mathbf{w}^i, b^i[r], \mathbf{x}_t^i) = \begin{cases} 0 & (\mathbf{w}^i \cdot \mathbf{x}_t^i - b^i[r])l[r] \geq 1 \\ 1 - (\mathbf{w}^i \cdot \mathbf{x}_t^i - b^i[r])l[r] & \text{otherwise} \end{cases}$$

We introduce a non-negative slack variable ξ into the optimization problem (5.4), which scales quadratically with the objective function. This leads to the following optimization problem.

$$\min_{\mathbf{w}^i, \mathbf{b}^i} \quad \frac{\eta_1}{2} \|(\mathbf{w}^i, \mathbf{b}^i) - (\mathbf{w}_t^i, \mathbf{b}_t^i)\|^2 + \frac{\eta_2}{2} \|(\mathbf{w}^i, \mathbf{b}^i) - (\bar{\mathbf{w}}_t, \bar{\mathbf{b}}_t)\|^2 + C \sum_{r=1}^{k-1} \xi_r^2 \quad (5.7a)$$

$$\text{s.t.} \quad \ell(\mathbf{w}^i, b^i[r], \mathbf{x}_t^i) \leq \xi_r \quad \text{for } r = 1, \dots, k-1 \quad (5.7b)$$

Using the Lagrange multiplier technique, the dual problem of (5.7) can be derived as

$$\min_{\tau} \quad \frac{1}{2} \|\mathbf{x}_t\|^2 \left(\sum_r \tau_r l_t[r] \right)^2 + \frac{1}{2} \sum_r \tau_r^2 + \frac{\eta_1 + \eta_2}{4C} \sum_r \tau_r^2 + \quad (5.8a)$$

$$\sum_r \tau_r (y_r (\eta_1 (\mathbf{w}_t^i \cdot \mathbf{x}_t - b_t^i[r]) + \eta_2 (\bar{\mathbf{w}}_t \cdot \mathbf{x}_t - \bar{b}_t[r])) - \eta_1 - \eta_2) \\ \text{s.t.} \quad \tau_r \geq 0 \quad \text{for } r = 1, \dots, k-1 \quad (5.8b)$$

We term the basic collaborative online ranking algorithm COR, and its large margin version CORM-I (based on optimization problem (5.4)), CORM-II (based on optimization problem (5.8)). Algorithm 5 summarizes the three methods. It is clear that COR generally enjoys linear time complexity with respect to the number of instances and dimensions, which is not different from typical online learners. CORM-I and CORM-II involve a simple quadratic optimization problem with a constraint, which can be solved using standard optimization tools effectively. Compared to learning several tasks separately, the proposed methods have negligible additional complexity, which is caused by averaging weight vectors. It is also feasible to explore parallel computing methods to significantly reduce the time complexity in the implementation stage. Therefore, our algorithm is both efficient and scalable for large-scale online learning tasks.

5.3 Experimental Results

5.3.1 Benchmark Setup

We compare the proposed collaborative online ranking algorithms (COR, CORM-I, and CORM-II) with PRank [51], NoPRank [53] (abbreviation for Norm-Optimized Prank, a

margin based version of PRank), and OAP-BPM [71]. We choose parameters η_1 and η_2 from a small range of values: $\{0.001, 0.01, 0.1, 1, 10, 100\}$ for COR, CORM-I, and CORM-II. We set β , the width of margin, to 1 for NoPRank. For OAP-BPM, the parameter K is set to 100 and τ is chosen from $\{0.3, 0.6, 0.9\}$. We use data from all tasks to train the NoPRank and OAP-BPM models, and evaluate them on each task specific data, respectively.

To examine the effectiveness of ranking multiple related tasks together, we compare our methods with two variants of the PRank algorithm as described below.

- (i) **PRank-Global** It learns a single ranking model from *all* tasks' data by applying the PRank algorithm. At each learning round, the algorithm receives a training instance from each task, and uses that instance to update its weight vector.
- (ii) **PRank-Unique** It employs the PRank algorithm to train a specific rank model for *each* task using only its own data. In other words, every task is associated with a unique ranking model.

We adopt the *cumulative error rate*, i.e., the ratio of the number of mistakes made by an online learner over the total number of examples received up-to-date as a metric for comparing different algorithms. Despite its extensive usage in online learning studies, the cumulative error rate does not consider the ordering preference of the ranks. For example, if the true rank is 1, then a predicted rank 2 is preferred over 3. To qualify the preference, we compute for each algorithm the *cumulative rank-loss rate* as the ratio of the absolute difference between the true and predicted ranks over the total number of examples received to date. Formally, the cumulative rank-loss rate at round T is $(1/T) \sum_t^T (|y_t - \hat{y}_t|)$. For both measures, a lower value corresponds to a better performance.

5.3.2 EachMovie Dataset

The EachMovie dataset has been used extensively for collaborative filtering research. It contains ratings entered by 72916 users on 1628 movies. A user rating has one of six possible values: 0, 0.2, 0.4, 0.6, 0.8, 1. Following Crammer et al. [51], we subtract 0.5 from each rating to polarize the range to $-0.5, -0.3, -0.1, -0.1, 0.3, 0.5$. In order to generate multiple related ranking problems, we first choose 1813 users who have viewed at least

Table 5.1: Run-time (in seconds) for different algorithms

	EachMovie	MovieLens	Amazon Review
COR	12.76	2.81	67.91
CORM-I	37.5	27.03	156.25
CORM-II	44.6	26.91	160.4
PRank-Global	5.99	0.87	62.47
PRank-Unique	4.03	0.82	44.41
NoPRank	34.72	32.83	152.47
OAP-BPM	99.83	105.72	78.02

200 movies. These 1813 viewers are then divided into two sets: the first set contains 30 users who have viewed exactly 200 movies, and the second set comprises 1783 users who have viewed more than 200 movies. We choose one person among the viewers in the first set and treat his/her ratings as the target ranks. We use the ratings of all the people in the second set who viewed the same movie as features. A value of zero is assigned to movies that have not been rated by a particular user. Finally, we obtain 30 related tasks, each of which has 200 instances (feature-rank pairs).

The training instances are shuffled into a random sequence and fed to all algorithms. Figure 5.1 depicts the average variation of the cumulative rank loss rate and rank loss rate over 20 tasks for the entire online learning process. We also report the run-time for each algorithm, i.e., the total time consumed by updating the model and making predictions in Table 5.1. From these results, we can make several observations.

First, the proposed collaborative online ranking methods (COR, CORM-I, and CORM-II) consistently outperform the other four online learners in terms of error rate and rank loss rate. CORM-II has the overall lowest error rate of 60%, which is almost 15% better than PRank-Unique. Note that CORM-I is the second best performer, and COR is the second runner-up. This shows that learning tasks collaboratively outperforms the individually trained baselines.

Second, by comparing the basic algorithms with variants that employ the large margin principle (PRank-Global vs. NoPrank, COR vs. CORM), we see that the latter is always better than the former. This indicates that imposing large margin constraint has positive effects on solving the ranking problem.

Figure 5.1: Experimental results of the EachMovie dataset.

Third, Table 5.1 shows that COR is slightly slower than the original PRank algorithm. This is expected since COR has to average the weight vectors of several models at each learning round. However, this extra computational cost is trivial compared to the cost of updating every task model. Both maximum margin based algorithms, i.e., CORM and NoPRank incurred considerable extra time compared to PRank based methods. This is because CORM and NoPRank have to invoke an optimization routine on each round. However, by comparing the two margin based algorithms, we see that CORM vastly outperformed NoPRank on both the cumulative error rate and rank loss rate with negligible additional computational cost. This again validates the efficiency of the proposed collaborative online ranking methods.

5.3.3 MovieLens Dataset

The MovieLens dataset³ consists of ten thousand ratings (integers ranging from 1 to 5) from 943 users on 1682 movies. We build a subset of the MovieLens dataset by excluding users who gave less than 300 ratings. The same features as for the EachMovie dataset were extracted, i.e., a user's ratings serve as the target ranks while the ratings of fellow viewers serve as features. Specifically, we selected 15 user ratings as target ranks, and used the remaining 129 user ratings as features. Finally, the data used in the experiment

³<http://www.grouplens.org/node/73>

contained 20 tasks corresponding to the preferences of 20 different users, who have rated 388 to 687 movies.

Figure 5.2: Experimental results of the MovieLens dataset.

The cumulative error rate and rank loss rate results averaged over 15 tasks are plotted in Figure 5.2. Note that the perturbations appearing near the tails of curves are due to the varying number of instances belonging to each task. Some learning tasks with fewer samples ended earlier than others, which affected the averages, thus causing the perturbations. In spite of the perturbations, the COR and CORM curves still lie way below the others, indicating that the proposed methods made far fewer errors compared to the others. In particular, CORM is significantly better than the NoPRank baseline, as the overall error rate has been reduced by 20%.

5.3.4 Multi-Domain Sentiment Dataset

The data constructed by Blitzer et al. [21] has been previously used in transfer learning studies. It comprises reviews of four types of products from Amazon.com: books (5501 reviews), DVDs (5118 reviews), electronics (5901 reviews), and kitchen appliances (5149 reviews). Each review comes with an integer rating of 0 to 4 stars. The review text is converted to a word vector using the TF-IDF (term frequency inverse document frequency) representation.

Figure 5.3: Cumulative error rate of the multi-domain sentiment dataset.

We show the variation of cumulative error rates for the entire online learning process in Figure 5.3, and that of cumulative rank loss rates in Figure 5.4. Similar to the results of the previous two datasets, the results of this dataset again show that it is possible to outperform both the global and unique models by solving related ranking problems collaboratively. We note that the basic form of our collaborative online ranking (COR) actually performed slightly worse than the NoPRank algorithm. This is expected since COR does not impose the large margin constraint, which is useful for rank prediction as shown in the previous experiments. By requiring the algorithm to rank samples correctly with a large enough margin, our CORM-I and CORM-II consistently beat the large

margin single task learning algorithm NoPRank. Taking into consideration the minimal additional time cost incurred by COR, CORM-I, and CORM-II as shown in Table 5.1, we can conclude that our algorithm is both effective and efficient for solving ranking problems in real world applications.

Figure 5.4: Cumulative rank loss rate of the multi-domain sentiment dataset.

5.4 Discussion

In this chapter, we show a collaborative online ranking method that is able to take advantage of individual and global models to achieve an overall improvement in ranking

performance for jointly solving multiple related ranking problems. To the best of our knowledge, there has been few attempts at exploring ranking problems in the online multitask learning setting. We show that the coherent integration of the global and unique models in a unified collaborative learning framework led to performances that are better than either one alone. The experimental results demonstrate that our algorithms are both effective and efficient for real-world applications.

Algorithm 5: Collaborative Online Ranking

Input: sequences of instance-rank pairs of K tasks $(\mathbf{x}_t^i, y_t^i)$, $i = 1, \dots, K$,
 $t = 1, \dots, T$, parameters $\eta_1 \geq 0$, $\eta_2 \geq 0$, $C > 0$ (for CORM-I)
Initialize: set $\mathbf{w}_0^i = \mathbf{0}$, $b_0^i[1], \dots, b_0^i[k-1] = 0$, $b_0^i[k] = \infty$
for $t \leftarrow 1$ **to** T **do**
 for $i \leftarrow 1$ **to** N **do**
 compute the global model $\bar{\mathbf{w}}_t = \sum_{j \in N, j \neq i} \mathbf{w}_t^j / (N - 1)$;
 receive a new instance of the i -th task $\mathbf{x}_t^i$;
 predict $\hat{y}_t^i = \min_{r \in \{1, \dots, k\}} \{r : \mathbf{w}_t^i \cdot \mathbf{x}_t^i - b_t^i[r] < 0\}$;
 receive a new rank value of the i -th task y_t^i ;
 if $y_t^i \neq \hat{y}_t^i$ **then**
 for $r \leftarrow 1$ **to** $k - 1$ **do**
 if $y_t^i \leq r$ **then** $l[r] = -1$;
 else $l[r] = 1$;
 end
 // Update rule for COR
 for $r \leftarrow 1$ **to** $k - 1$ **do**
 if $(\mathbf{w}_t^i \cdot \mathbf{x}_t^i - b_t^i[r])l[r] \leq 0$ **then** $\tau_r = l[r]$;
 else $\tau_r = 0$;
 end
 update $\mathbf{w}_{t+1}^i = \frac{\eta_1 \mathbf{w}_t^i + \eta_2 \bar{\mathbf{w}}_t}{\eta_1 + \eta_2} + \sum_r \tau_r \mathbf{x}_t^i$;
 for $r \leftarrow 1$ **to** $k - 1$ **do**
 update $b_{t+1}^i[r] = b_t^i[r] - \tau_r$;
 end
 // Update rule for CORM-I and CORM-II
 obtain τ by solving problem (5.6) (for CORM-I) or problem (5.8) (for CORM-II);
 update $\mathbf{w}_{t+1}^i = \frac{\eta_1 \mathbf{w}_t^i + \eta_2 \bar{\mathbf{w}}_t + (\sum_r \tau_r l[r]) \mathbf{x}_t^i}{\eta_1 + \eta_2}$;
 for $r \leftarrow 1$ **to** $k - 1$ **do**
 update $b_{t+1}^i[r] = \frac{\eta_1 b_t^i[r] + \eta_2 \bar{b}_t[r] - \tau_r l[r]}{\eta_1 + \eta_2}$;
 end
 end
 end
end

Chapter 6

Conclusion and Future Work

This chapter summarizes the thesis and presents several future directions.

6.1 Summary of the Thesis

This thesis investigates how machine learning classifiers can work collaboratively from two perspectives: data-centric and task-centric. For the data centric collaborative learning, we present a two-view transductive SVM that is able to take advantage of multiple representations of the same problem to achieve an overall improvement in classification performance for problems lacking in labeled data. It is an extension of the existing two-view supervised learning algorithm into a semi-supervised setting. We also incorporated graph-based semi-supervised learning into our algorithm by exploring the intrinsic manifold structure of training data. We formulated our learning framework into an optimization problem and presented an efficient way to solve it. Experimental results on synthetic and real-life datasets justify the utility of our two-view transductive learning algorithm. In particular, requiring the two views to maintain a maximum consensus on both labeled and unlabeled data always makes the classifier more accurate and stable. Moreover, incorporating manifold information can also help to improve the classification performance.

Our two-view transductive SVM was somewhat motivated by our need to detect spam product reviews from online forums. Experimental results were promising on the review spam detection task: a model trained with a few labeled data using our algorithm is comparable to one trained on a significantly larger amount of labeled data using supervised

learning. The task of product review mining can be enhanced by applying our method to detect and filter spam reviews. In fact, for a general classification problem, our algorithm is more accurate and stable compared to traditional transductive SVM trained from data in a single view.

The second half of this thesis examined the task centric collaborative learning problem under the online learning setting from two aspects: classification and ranking. We first studied online multitask learning for solving multiple related classification tasks in parallel, with the goal of accurately and efficiently classifying data received by each task in sequence. Our approach is an extension of Crammer’s [47] online learning algorithm. In particular, it was motivated by our need to detect if a user is expressing emotions in micro-blogging as early and accurately as possible. We evaluated our algorithm on a real-life micro-blogging dataset collected from *Twitter*. Experimental results were promising on the sentiment classification task: a model trained with our algorithm achieved fewer cumulative errors than models trained using other approaches.

We then extended collaborative online learning to solve the ranking problem. The proposed collaborative online ranking algorithm can rank data generated by a group of tasks by combining group and individual characteristics. It is so far the first attempt at exploring ranking problem in an online multitask learning setting. We showed that it is able to outperform both the global and unique models by coherently integrating them in a unified collaborative learning framework. The experimental results demonstrate that our algorithms are both effective and efficient for real-life applications.

To sum up, both the proposed collaborative online classification and ranking algorithms are able to take advantage of individual and global models to achieve an overall improvement in classification/ranking performance for jointly learning multiple correlated tasks. Experimental results show that it is able to outperform both the global and personal models by coherently integrating them in a unified collaborative learning framework. In addition, thanks to the simplicity of online learning, the proposed algorithms are highly efficient and scalable, and are especially suitable for problems whose data are generated in a sequential manner, or massive datasets that are too large to fit in memory.

We highlight the contributions of our research as follows.

- (i) We devised a systematic methodology to collect two-view data from online product review forums for spam review detection.
- (ii) We extended the existing two-view supervised learning algorithm to a semi-supervised setting, and created a new multi-view semi-supervised learning algorithm, namely the two-view transductive SVM, which takes advantage of both the abundant amount of unlabeled data and their multiple representations to improve classification performance.
- (iii) We incorporated manifold regularization, which is a type of graph-based semi-supervised learning into our two-view transductive SVM framework. Experimental results indicate that incorporating additional manifold information can also help to improve the classification results.
- (iv) We studied the multitask learning problem under the online learning setting, and presented several collaborative online learning algorithms for classifying or ranking data generated by a group of different but related tasks. The proposed collaborative online ranking algorithm is the first attempt at exploring ranking problems in an online multitask learning setting.
- (v) We investigated a new micro-blogging data mining problem, “micro-blogging sentiment detection”, which aims to identify and classify emotional micro-blogging posts.
- (vi) We evaluated the empirical performances of our algorithms on various datasets. The encouraging results validated the effectiveness of our methods.

6.2 Future Work

The future directions of our research on multi-view semi-supervised learning are listed as follows.

- We plan to seek alternative ways to represent the consensus between the two views and examine the performance of our algorithm on more datasets.

- Further, we are interested to derive the theoretical error bounds of our algorithm, and also study its convergence properties and conditions as well as other efficiency issues. One possible way is to apply the theory of Rademacher complexity [12] to analyze the theoretical risk of our algorithm.
- Finally, we could also investigate some state-of-the-art kernel learning techniques [75] [74] [114] for optimizing the kernel functions in building more effective two-view transductive SVM models.

We also plan to extend our collaborative online learning framework in the following ways.

- We have not explicitly utilized content similarity between successive micro-blog posts, though this is implicitly considered in a classification context when the current model weight is compared to the previous one. Thus, both the global and temporal models can be further refined based on content and temporal proximity of successive tweets.
- Further, we will take into consideration the time element, which for now has been left out as we simply consider sequential presentation of each micro-blog regardless of the publication time-stamp. Intuitively, micro-blogs that are published in quick succession (minutes, hours) should be more relevant than those that are published days, weeks, or months apart. A possible solution is to take the most recent k tweets into consideration by introducing a factor decreases with increasing time intervals. Specifically, we can amend the collaborative online learning framework as follows

$$\mathbf{w}_{t+1} = \operatorname{argmin}_{\mathbf{w} \in \mathbb{R}^d} \quad \frac{\eta}{2} \|\mathbf{w} - \mathbf{w}_t\|^2 + \frac{1-\eta}{2} \|\mathbf{w} - \mathbf{u}_t\|^2 + C\xi \quad (6.1a)$$

$$\text{s.t.} \quad \sum_{k=1}^t \gamma^{t-k} \ell(\mathbf{w}; (\mathbf{x}_k, y_k)) \leq \xi \quad (6.1b)$$

$$\xi \geq 0 \quad (6.1c)$$

where t is the number of previous training instances that we wish to take into consideration, and $0 \leq \gamma \leq 1$ is a damping parameter. Clearly, as the time interval

increases, the influence of old training examples on the loss function will decrease. This formulation satisfies our assumption about the sequential similarity of micro-blog posts.

- We hope to be able to extend our experiments to a more substantial size dataset and also to more applications. Our methods assume uniform relations across tasks. However, it is more reasonable to take into account the degree of relatedness among tasks. How to incorporate hierarchies and clusters of tasks is also worthy of further study.

Publications

- (i) Guangxia Li, Kuiyu Chang, Steven C. H. Hoi: Multiview Semi-Supervised Learning with Consensus. *IEEE Trans. Knowl. Data Eng.* 24(11): 2040-2051 (2012)
- (ii) Guangxia Li, Kuiyu Chang, Steven C. H. Hoi, Wenting Liu, Ramesh Jain: Collaborative Online Learning of User Generated Content. *CIKM 2011*: 285-290
- (iii) Guangxia Li, Steven C. H. Hoi, Kuiyu Chang, Ramesh Jain: Micro-blogging Sentiment Detection by Collaborative Online Learning. *ICDM 2010*: 893-898
- (iv) Guangxia Li, Steven C. H. Hoi, Kuiyu Chang: Two-View Transductive Support Vector Machines. *SDM 2010*: 235-244
- (v) Guangxia Li, Steven C. H. Hoi, Kuiyu Chang, Wenting Liu, Ramesh Jain: Micro-blogging Sentiment Detection by Collaborative Online Learning. *IEEE Trans. Knowl. Data Eng.* In Print
- (vi) Wenting Liu, Guangxia Li, James Cheng: Fast Approximation of PageRank by Adaptive Sampling. *Knowl. Inf. Syst.* In Print

References

- [1] J. Abernethy, P. L. Bartlett, and A. Rakhlin. Multitask learning with expert advice. In *COLT*, pages 484–498, 2007.
- [2] A. Alekh, R. Alexander, and B. Peter. Matrix regularization techniques for online multitask learning. Technical report, EECS Department, University of California, Berkeley, 2008.
- [3] G. M. Allenby and P. E. Rossi. Marketing models of consumer heterogeneity. *Journal of Econometrics*, 1999.
- [4] R. K. Ando and T. Zhang. A framework for learning predictive structures from multiple tasks and unlabeled data. *Journal of Machine Learning Research*, 6:1817–1853, 2005.
- [5] R. K. Ando and T. Zhang. Two-view feature generation model for semi-supervised learning. In *ICML*, pages 25–32, 2007.
- [6] A. Argyriou, C.A. Micchelli, M. Pontil, and Y. Ying. A spectral regularization framework for multitask structure learning. In *NIPS*, pages 25–32, 2008.
- [7] A. Argyriou, T. Evgeniou, and M. Pontil. Multi-task feature learning. In *NIPS*, pages 41–48, 2006.
- [8] N. Arora, G. Allenby, and J. Ginter. A hierarchical bayes model of primary and secondary demand. *Marketing Science*, 1998.
- [9] B. Bakker and T. Heskes. Task clustering and gating for bayesian multi-tasking learning. *JMLR*, 4, 2003.

REFERENCES

- [10] L. Barbosa and J. Feng. Robust sentiment detection on twitter from biased and noisy data. In *COLING (Posters)*, pages 36–44, 2010.
- [11] T. Bartell and G. W. Cottrell. learning to retrieval information. *SCC*, 1995.
- [12] P. L. Bartlett and S. Mendelson. Rademacher and gaussian complexities: Risk bounds and structural results. *Journal of Machine Learning Research*, 3:463–482, 2002.
- [13] J. Baxter. A model for inductive bias learning. *Journal of Artificial Intelligence Research*, 2000.
- [14] M. Belkin and P. Niyogi. Using manifold structure for partially labeled classification. In *NIPS*, pages 929–936, 2002.
- [15] M. Belkin, P. Niyogi, and V. Sindhwani. On manifold regularization. In *AISTAT*, 2005.
- [16] M. Belkin, P. Niyogi, and V. Sindhwani. Manifold regularization: A geometric framework for learning from labeled and unlabeled examples. *Journal of Machine Learning Research*, 7:2399–2434, 2006.
- [17] S. Ben-David, J. Gehrke, and R. Schuller. Exploiting task relatedness for multiple task learning. *COLT*, 2003.
- [18] M. Bertero, T. A. Poggio, and V. Torre. Ill-posed problems in early vision. In *Proceedings of the IEEE*, pages 869–889, 1988.
- [19] D. P. Bertsekas. *Constrained Optimization and Lagrange Multiplier Methods (Optimization and Neural Computation Series)*. Athena Scientific, 1996.
- [20] A. Bifet and E. Frank. Sentiment knowledge discovery in twitter streaming data. In *Discovery Science*, pages 1–15, 2010.
- [21] J. Blitzer, M. Dredze, and F. Pereira. Biographies, bollywood, boom-boxes and blenders: Domain adaptation for sentiment classification. In *ACL*, 2007.

- [22] A. Blum. On-line algorithms in machine learning (a survey). In *Online Algorithms: The State of the Art*, volume Chapter 14, 1998.
- [23] A. Blum and S. Chawla. Learning from labeled and unlabeled data using graph mincuts. In *ICML*, pages 19–26, 2001.
- [24] A. Blum and T. Mitchell. Combining labeled and unlabeled data with co-training. In *COLT*, pages 92–100, 1998.
- [25] E. V. Bonilla, K. M. Chai, and C. K. I. Williams. Multi-task gaussian process prediction. In *NIPS*, 2007.
- [26] C. Burges. From ranknet to lambdarank to lambdamart: An overview. *Microsoft Research Technical Report*, 2010.
- [27] C. J. C. Burges and R. Ragno. Learning to rank with nonsmoothcost functions. *NIPS*, 2006.
- [28] C. J. C. Burges, T. Shaked, E. Renshaw, A. Lazier, M. Deeds, N. Hamilton, and G. N. Hullender. Learning to rank using gradient descent. In *ICML*, pages 89–96, 2005.
- [29] Y. Cao and J. Xu. Adapting ranking svm to information retrieval. *SIGIR*, 2006.
- [30] Y. Cao, J. Xu, T.-Y. Liu, H. Li, Y. Huang, and H.-W. Hon. Adapting ranking svm to document retrieval. In *SIGIR*, pages 186–193, 2006.
- [31] Z. Cao and T. Qin. Learning to rank: From pairwise to listwise approach. *ICML*, 2007.
- [32] Z. Cao, T. Qin, T.-Y. Liu, M.-F. Tsai, and H. Li. Learning to rank: from pairwise approach to listwise approach. In *ICML*, pages 129–136, 2007.
- [33] R. Caruana. Multitask learning. *Machine Learning*, 28(1):41–75, 1997.
- [34] G. Cavallanti, N. Cesa-Bianchi, and C. Gentile. Linear algorithms for online multitask classification. *Journal of Machine Learning Research*, 11:2901–2934, 2010.

- [35] N. Cesa-Bianchi, A. Conconi, and C. Gentile. On the generalization ability of on-line learning algorithms. *IEEE Transactions on Information Theory*, 50(9):2050–2057, 2004.
- [36] N. Cesa-Bianchi, A. Conconi, and C. Gentile. A second-order perceptron algorithm. *SIAM J. Comput.*, 34(3):640–668, 2005.
- [37] C.-C. Chang and C.-J. Lin. *LIBSVM: a library for support vector machines*, 2001.
- [38] O. Chapelle and A. Zien. Semi-supervised classification by low density separation. In *AISTAT*, 2005.
- [39] W. Christian, A. Yasemin, T. Nora, and R. Gunnar. Inferring latent task structure for multi-task learning via multiple kernel learning. *BMC Bioinformatics*, 11(Suppl 8):S5, 2010.
- [40] W. Chu and S. S. Keerthi. New approaches to support vector ordinal regression. In *ICML*, pages 145–152, 2005.
- [41] F. R. K. Chung. *Spectral Graph Theory*. CBMS-AMS, 1994.
- [42] W. W. Cohen, R. E. Schapire, and Y. Singer. Learning to order things. *J. Artif. Intell. Res. (JAIR)*, 10:243–270, 1999.
- [43] R. Collobert, F. H. Sinz, J. Weston, and L. Bottou. Large scale transductive svms. *Journal of Machine Learning Research*, 7:1687–1712, 2006.
- [44] R. Collobert and J. Weston. A unified architecture for natural language processing: deep neural networks with multitask learning. In *ICML*, pages 160–167, 2008.
- [45] C. Cortes and M. Mohri. Magnitude-preserving ranking algorithms. *ICML*, 2007.
- [46] D. Cossock and T. Zhang. Subset ranking using regression. In *COLT*, pages 605–619, 2006.
- [47] K. Crammer, O. Dekel, J. Keshet, S. Shalev-Shwartz, and Y. Singer. Online passive-aggressive algorithms. *Journal of Machine Learning Research*, 7:551–585, 2006.

- [48] K. Crammer, M. Dredze, and A. Kulesza. Multi-class confidence weighted algorithms. In *EMNLP*, pages 496–504, 2009.
- [49] K. Crammer, M. Dredze, and F. Pereira. Exact convex confidence-weighted learning. In *NIPS*, pages 345–352, 2008.
- [50] K. Crammer, A. Kulesza, and M. Dredze. Adaptive regularization of weight vectors. In *Advances in Neural Information Processing Systems 22*, pages 414–422, 2009.
- [51] K. Crammer and Y. Singer. Pranking with ranking. In *NIPS*, pages 641–647, 2001.
- [52] K. Crammer and Y. Singer. Ultraconservative online algorithms for multiclass problems. *Journal of Machine Learning Research*, 3:951–991, 2001.
- [53] K. Crammer and Y. Singer. Online ranking by projecting. *Neural Computation*, 17(1):145–175, 2005.
- [54] S. Dasgupta, M. L. Littman, and D. A. McAllester. Pac generalization bounds for co-training. In *NIPS*, pages 375–382, 2001.
- [55] K. Dave, S. Lawrence, and D. M. Pennock. Mining the peanut gallery: opinion extraction and semantic classification of product reviews. In *WWW*, pages 519–528, 2003.
- [56] D. Davidov, O. Tsur, and A. Rappoport. Enhanced sentiment learning using twitter hashtags and smileys. In *COLING (Posters)*, pages 241–249, 2010.
- [57] D. Davidov, O. Tsur, and A. Rappoport. Semi-Supervised Recognition of Sarcastic Sentences in Twitter and Amazon. In *Proceeding of the 23rd international conference on Computational Linguistics (COLING)*, July 2010.
- [58] V. R. de Sa. Learning classification with unlabeled data. In *NIPS*, pages 112–119, 1993.
- [59] O. Dekel, P. M. Long, and Y. Singer. Online multitask learning. In *COLT*, pages 453–467, 2006.

- [60] M. Dredze, K. Crammer, and F. Pereira. Confidence-weighted linear classification. In *ICML*, pages 264–271, 2008.
- [61] T. Evgeniou, C. A. Micchelli, and M. Pontil. Learning multiple tasks with kernel methods. *J. Mach. Learn. Res.*, 6:615–637, 2005.
- [62] T. Evgeniou and M. Pontil. Regularized multi-task learning. In *KDD*, pages 109–117, 2004.
- [63] R.-E. Fan, K.-W. Chang, C.-J. Hsieh, X.-R. Wang, and C.-J. Lin. LIBLINEAR: A library for large linear classification. *Journal of Machine Learning Research*, 9:1871–1874, 2008.
- [64] J. D. R. Farquhar, D. R. Hardoon, H. Meng, J. Shawe-Taylor, and S. Szedmák. Two view learning: Svm-2k, theory and practice. In *NIPS*, 2005.
- [65] Y. Freund and R. Iyer. An efficient boosting algorithm for combining preferences. *JMLR*, 2003.
- [66] J. Gao and H. Qi. Linear discriminant model for information retrieval. *SIGIR*, 2005.
- [67] C. Gentile. A new approximate maximal margin classification algorithm. *Journal of Machine Learning Research*, 2:213–242, 2001.
- [68] S. A. Goldman and Y. Zhou. Enhancing supervised learning with unlabeled data. In *ICML*, pages 327–334, 2000.
- [69] A. J. Grove, N. Littlestone, and D. Schuurmans. General convergence results for linear discriminant updates. *Machine Learning*, pages 173–210, 2001.
- [70] D. R. Hardoon, S. Szedmák, and J. Shawe-Taylor. Canonical correlation analysis: An overview with application to learning methods. *Neural Computation*, 16(12):2639–2664, 2004.
- [71] E. F. Harrington. Online ranking/collaborative filtering using the perceptron algorithm. In *ICML*, pages 250–257, 2003.

- [72] R. Herbrich and T. Graepel. Support vector learning for ordinal regression. *ICANN*, 1999.
- [73] R. Herbrich, T. Graepel, and C. Campbell. Bayes point machines. *Journal of Machine Learning Research*, 1:245–279, 2001.
- [74] S. C. H. Hoi, R. Jin, and M. R. Lyu. Learning nonparametric kernel matrices from pairwise constraints. In *ICML*, pages 361–368, 2007.
- [75] S. C. H. Hoi, M. R. Lyu, and E. Y. Chang. Learning the unified kernel machines for classification. In *KDD*, pages 187–196, 2006.
- [76] H. D. III. Bayesian multitask learning with latent hierarchies. *UAI*, 2009.
- [77] L. Jacob, F. Bach, and J. Vert. Clustered multi-task learning. In *NIPS*, pages 745–752, 2009.
- [78] L. Jacob and J.-P. Vert. Efficient peptide-mhc-i binding prediction for alleles with few known binders. *Bioinformatics*, 24(3):358–366, 2008.
- [79] A. Java, X. Song, T. Finin, and B. L. Tseng. Why we twitter: An analysis of a microblogging community. In *WebKDD/SNA-KDD*, pages 118–138, 2007.
- [80] N. Jindal and B. Liu. Review spam detection. In *WWW*, pages 1189–1190, 2007.
- [81] T. Joachims. Text categorization with support vector machines: Learning with many relevant features. In *ECML*, pages 137–142, 1998.
- [82] Z. Kang, K. Grauman, and F. sha. Learning with whom to share in multi-task feature learning. In *ICML*, 2011.
- [83] J. Kivinen and M. K. Warmuth. Additive versus exponentiated gradient updates for linear prediction. In *Information and Computation*, pages 1–64, 1997.
- [84] E. Kouloumpis, T. Wilson, and J. Moore. Twitter sentiment analysis: The good the bad and the omg! In *ICWSM*, 2011.

- [85] N. Kushmerick. Learning to remove internet advertisements. In *AGENTS '99: Proceedings of the third annual conference on Autonomous Agents*, pages 175–181, New York, NY, USA, 1999. ACM.
- [86] S. Lee, V. Chatalbashev, D. Vickrey, and D. Koller. Learning a meta-level prior for feature relevance from multiple related tasks. *ICML*, 2007.
- [87] P. Li, C. J. C. Burges, and Q. Wu. Mcrank: Learning to rank using multiple classification and gradient boosting. In *NIPS*, 2007.
- [88] Y. Li and P. M. Long. The relaxed online maximum margin algorithm. In *NIPS*, pages 498–504, 1999.
- [89] J. Liu, Y. Cao, C.-Y. Lin, Y. Huang, and M. Zhou. Low-quality product review detection in opinion summarization. In *Proceedings of the Joint Conference on Empirical Methods in Natural Language Processing and Computational Natural Language Learning (EMNLP-CoNLL)*, pages 334–342, 2007. Poster paper.
- [90] J. Liu, S. Ji, and J. Ye. Multi-task feature learning via efficient l_2 , l_1 -norm minimization. *UAI*, pages 339–348, 2009.
- [91] T.-Y. Liu. *Learning to Rank for Information Retrieval*. Springer, 2011.
- [92] G. Lugosi, O. Papaspiliopoulos, and G. Stoltz. Online multi-task learning with hard constraints. *COLT*, 2009.
- [93] C. A. Micchelli and M. Pontil. On learning vector-valued functions. *Neural Computation*, pages 177–204, 2005.
- [94] S. min Kim, P. Pantel, T. Chklovski, and M. Pennacchiotti. Automatically assessing review helpfulness. In *In Proceedings of the Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 423–430, 2006.
- [95] T. T. Nguyen, K. Chang, and S. C. Hui. Distribution-aware online classifiers. In *IJCAI*, pages 1427–1432, 2011.

- [96] K. Nigam and R. Ghani. Analyzing the effectiveness and applicability of co-training. In *CIKM*, pages 86–93, 2000.
- [97] J. Nocedal and S. J. Wright. *Numerical Optimization*. Springer, 2000.
- [98] A. Pak and P. Paroubek. Twitter as a corpus for sentiment analysis and opinion mining. In *LREC*, 2010.
- [99] B. Pang and L. Lee. Opinion mining and sentiment analysis. *Foundations and Trends in Information Retrieval*, 2(1-2):1–135, 2007.
- [100] B. Pang, L. Lee, and S. Vaithyanathan. Thumbs up? sentiment classification using machine learning techniques. In *EMNLP '02: Proceedings of the ACL-02 conference on Empirical methods in natural language processing*, pages 79–86, 2002.
- [101] S. Parameswaran and K. Weinberger. Large margin multi-task metric learning. *NIPS*, pages 1867–1875, 2010.
- [102] B. Peters, H.-H. Bui, S. Frankild, M. Nielsen, C. Lundegaard, and E. Kostem. A community resource benchmarking predictions of peptide binding to mhc-i molecules. *PLoS Computational Biology*, 2(6), 2006.
- [103] T. Qin and T. Liu. Multiple hyperplane ranker. *SIGIR*, 2007.
- [104] P. Rai and H. D. III. Infinite predictor subspace models for multitask learning. *TATS*, 2010.
- [105] S. Rajaram and A. Garg. Classification approach towards ranking and sorting problems. *ECML*, 2003.
- [106] H. Ralf, G. Thore, and O. Klaus. *Large margin rank boundaries for ordinal regression*. MIT Press, Cambridge, MA, 2000.
- [107] F. Rosenblatt. Neurocomputing: foundations of research. chapter The perception: a probabilistic model for information storage and organization in the brain, pages 89–114. MIT Press, Cambridge, MA, USA, 1988.

- [108] F. Sebastiani. Machine learning in automated text categorization. *ACM Comput. Surv.*, 34(1):1–47, 2002.
- [109] T. Serre, M. Kouh, C. Cadieu, U. Knoblich, G. Kreiman, and T. Poggio. Theory of object recognition: computations and circuits in the feedforward path of the ventral stream in primate visual cortex. *MIT*, 2005.
- [110] A. Shashua and A. Levin. Ranking with large margin principle: Two approaches. In *NIPS*, pages 937–944, 2002.
- [111] D. Sheldon. Graphical multi-task learning. In *NIPS’08 Workshop on Structured Input and Structured Output*, 2008.
- [112] V. Sindhwani and P. Niyogi. A co-regularized approach to semi-supervised learning with multiple views. In *Proceedings of the ICML Workshop on Learning with Multiple Views*, 2005.
- [113] V. Sindhwani, P. Niyogi, and M. Belkin. Beyond the point cloud: from transductive to semi-supervised learning. In *ICML*, pages 824–831, 2005.
- [114] S. Sonnenburg, G. Rätsch, C. Schäfer, and B. Schölkopf. Large scale multiple kernel learning. *Journal of Machine Learning Research*, 7:1531–1565, 2006.
- [115] S. Szedmák and J. Shawe-Taylor. Synthesis of maximum margin and multiview learning using unlabeled data. *Neurocomputing*, 70(7-9):1254–1264, 2007.
- [116] S. Thrun and J. O’Sullivan. Clustering learning tasks and the selective cross-task transfer of knowledge. *Kluwer Academic Publishers*, pages 235–257, 1998.
- [117] A. N. Tikhonov and V. Y. Arsenin. *Solutions of Ill-Posed Problems*. Vh Winston, 1977.
- [118] M. Tsai and T. Liu. Frank: A ranking method with fidelity loss. *SIGIR*, 2007.
- [119] V. N. Vapnik. *Statistical Learning Theory*. Wiley-Interscience, 1998.
- [120] M. N. Volkovs and R. S. Zemel. Boltzrank: Learning to maximize expected ranking gain. *ICML*, 2009.

- [121] J. Xu and H. Li. A boosting algorithm for information retrieval. *SIGIR*, 2007.
- [122] L. Yang, R. Jin, and J. Ye. Online learning by ellipsoid method. In *ICML*, page 145, 2009.
- [123] K. Yu, V. Tresp, and A. Schwaighofer. Learning gaussian processes from multiple tasks. In *ICML*, pages 1012–1019, 2005.
- [124] A. L. Yuille and A. Rangarajan. The concave-convex procedure (cccp). In *NIPS*, pages 1033–1040, 2001.
- [125] Y. Zhang and D. Yeung. A convex formulation for learning task relationships in multi-task learning. *UAI*, 2010.
- [126] Z. Zhang and B. Varadarajan. Utility scoring of product reviews. In *CIKM*, pages 51–57, 2006.
- [127] P. Zhao, S. C.-H. Hoi, and R. Jin. Duol: A double updating approach for online learning. In *Advances in Neural Information Processing Systems 22 (NIPS)*, pages 2259–2267, 2009.
- [128] Z. Zheng and H. Zha. A general boosting method and its application to learning ranking functions for web search. *NIPS*, 2007.
- [129] D. Zhou, O. Bousquet, T. N. Lal, J. Weston, and B. Schölkopf. Learning with local and global consistency. In *NIPS*, 2003.
- [130] Z.-H. Zhou, K.-J. Chen, and H.-B. Dai. Enhancing relevance feedback in image retrieval using unlabeled data. *ACM Trans. Inf. Syst.*, 24(2):219–244, 2006.
- [131] Z.-H. Zhou and M. Li. Tri-training: Exploiting unlabeled data using three classifiers. *IEEE Trans. Knowl. Data Eng.*, 17(11):1529–1541, 2005.
- [132] X. Zhu. Semi-supervised learning literature survey. Technical Report 1530, Computer Sciences, University of Wisconsin-Madison, 2005.
- [133] X. Zhu, Z. Ghahramani, and J. D. Lafferty. Semi-supervised learning using gaussian fields and harmonic functions. In *ICML*, pages 912–919, 2003.