Sparse Learning Under Regularization Framework

YANG, Haiqin

A Thesis Submitted in Partial Fulfilment
of the Requirements for the Degree of
Doctor of Philosophy
in
Computer Science and Engineering

The Chinese University of Hong Kong
January 2011

Thesis/Assessment Committee

Professor LEUNG Kwong Sak (Chair)

Professor KING Kuo Chin Irwin (Thesis Supervisor)

Professor LYU Rung Tsong Michael (Thesis Supervisor)

Professor CHAN Lai Wan (Committee Member)

Professor YEUNG Dit-Yan (External Examiner)

Abstract of thesis entitled:
Sparse Learning Under Regularization Framework
Submitted by YANG, Haiqin
for the degree of Doctor of Philosophy
at The Chinese University of Hong Kong in January 2011

Regularization is a dominant theme in machine learning and statistics due to its prominent ability in providing an intuitive and principled tool for learning from high-dimensional data. As large-scale learning applications become popular, developing efficient algorithms and parsimonious models become promising and necessary for these applications. Aiming at solving large-scale learning problems, this thesis tackles the key research problems ranging from feature selection to learning with unlabeled data and learning data similarity representation. More specifically, we focus on the problems in three areas: online learning, semi-supervised learning, and multiple kernel learning.

The first part of this thesis develops a novel online learning framework to solve group lasso and multi-task feature selection. To the best our knowledge, the proposed online learning framework is the first framework for the corresponding models. The main advantages of the online learning algorithms are that 1) they can work on the applications where training data appear sequentially; consequently, the training procedure can be started at any time; 2) they can handle data up to any size with any number of features. The efficiency of the algorithms is attained because we derive closed-form solutions to update the weights of the corresponding models. At each iteration, the online learning algorithms just need $\mathcal{O}(d)$ time complexity and memory cost for

group lasso, while they need $\mathcal{O}(d \times Q)$ for multi-task feature selection, where d is the number of dimensions and Q is the number of tasks. Moreover, we provide theoretical analysis for the average regret of the online learning algorithms, which also guarantees the convergence rate of the algorithms. In addition, we extend the online learning framework to solve several related models which yield more sparse solutions.

The second part of this thesis addresses a general scenario of semi-supervised learning for the binary classification problem, where the unlabeled data may be a mixture of relevant and irrelevant data to the target binary classification task. Without specifying the relatedness in the unlabeled data, we develop a novel maximum margin classifier, named the *tri-class support vector machine* (3C-SVM), to seek an inductive rule that can separate these data into three categories: -1, $+1$, or 0. This is achieved by adopting a novel min loss function and following the maximum entropy principle. For the implementation, we approximate the problem and solve it by a standard concave-convex procedure (CCCP). The approach is very efficient and it is possible to solve large-scale datasets.

The third part of this thesis focuses on multiple kernel learning (MKL) to solve the insufficiency of the L_1-MKL and the L_p-MKL models. Hence, we propose a generalized MKL (GMKL) model by introducing an elastic net-type constraint on the kernel weights. More specifically, it is an MKL model with a constraint on a linear combination of the L_1-norm and the square of the L_2-norm on the kernel weights to seek the optimal kernel combination weights. Therefore, previous MKL problems based on the L_1-norm or the L_2-norm constraints can be regarded as its special cases. Moreover, our GMKL enjoys the favorable sparsity property on the solution and also facilitates the grouping effect. In addition, the optimization of our GMKL is a convex optimization problem, where a local solution is the globally op-

timal solution. We further derive the level method to efficiently solve the optimization problem.

學位論文摘要

學位論文題目：正則化框架下的稀疏學習
提交人：楊海欽
學位：哲學博士
香港中文大學，二零一一年

正則化學習(regularization)由于具有能够给高维数据學習提供直觀和准則性工具的顯著能力，因此它成為了机器学习和统计學的研究主轴。目前大规模數據学习的应用广泛存在，为这些应用量身度造而提出的有效算法和儉约的學習模型将很有前景，亦十分必要。为了解决大规模數據學習的問題，本論文着重解決幾個關鍵問題，范圍包括特征選擇，未標記數據學習和數據相似表示的學習。具體而言，我們重點考慮三個研究方向的問題，即在線學習(online learning)，半監督學習(semi-supervised learning)和多核學習(multiple kernel learning，簡稱 MKL)。

本論文的第一部分提出了一個新穎的在線學習框架。在該框架下，我們解決了在線的組套索(group lasso)和多任務特征選擇(multi-task feature selection)兩個問題。據我們所知，該在線學習框架是第一個用于解決這兩個問題的在線學習框架。這里提出的在線學習算法最大的優勢包括：1）他們可以應用于訓練數據順序獲得的情形。在這種情況下，訓練過程都可以在任何時刻執行；2）他們可以處理任何數目和任何維數的數據。由于我們推導出更新相應模型權值的解析解，因此相應模型的訓練十分高

效。對于在線組套索模型，在每次迭代中，該算法最差時間複雜度和最差存儲成本是$O(d)$。相應的，對于在線多任務特征選擇模型，其每次迭代的最差時間複雜度和最差存儲成本是$O(d \times Q)$。其中d是數據的維數，Q是任務的個數。此外，我們亦提供了在線學習算法的平均后悔度(regret)的理論分析。這保證在線學習算法的收斂速度。同時，我們還對該在線學習框架進行擴展，解決幾個相關的模型，并得到更稀疏的解。

本論文第二部分解決了一般情況下二元分類的半監督學習的問題。這里“一般情況”是指未標記的數據可能是由跟期望分類的兩類數據相關或者不相關的數據組成。這里我們提出了一個新穎的最大間隔分類器，叫三類支持向量機(tri-class support vector machine, 簡稱 3C-SVM)。該分類器在不需要指定未標記數據跟目標分類數據是否具有關聯性的情況下，可以找到一條歸納法則把給定數據分成-1, +1, 或者 0 三類。該模型是基于最大熵原則和我們提出的一個新穎的最小損失函數實現其目的的。在算法實現上，我們通過對該模型進行近似并用標準的凹凸過程(concave-convex procedure，簡稱 CCCP)求解。因此，該模型的實現十分有效，并使之可能應用于大規模數據集上。

本論文第三部分重點在于解決目前已提出的典型的多核學習模型的不足，即一階多核學習模型(L_1 – MKL)和$_p$階多核學習模型(L_p – MKL)的不足。因此，我們通過對核權值引入彈性網絡型的限制，提出了一個廣義的多核學習模型(generalized MKL，簡稱 GMKL)。具體而言，該多核學習模型在尋求最佳核權值組合時加入對核權值的一階范式和二階范式的平方的線性加權的

它的特殊情況。此外，我們提出的廣義多核學習模型具有喜人的性質，即其解具有稀疏性和組效應。而且該模型的優化是一個凸優化問題，即局部最優解即是全局最優解。我們通過用水平(level)方法有效地解決其優化問題。

Acknowledgement

First and foremost, I would like to express my sincere appreciation to my supervisors, Prof. Irwin King and Prof. Michael R. Lyu, for their guidance, encouragement, and support during my Ph. D. study. Their meticulosity and patience in teaching and research influence me a lot in my education. I am indebted to them for their priceless and copious advice on conducting research, especially the skills in presentation, communication, and English writing. Their efforts have helped me much in finishing this thesis.

I would like to express my gratitude to Prof. Stephen Boyd for his helpful discussions when I visited his group at Stanford University. His excellent comments motivate me to develop efficient and interpretable learning models and make me turn my research focus to the online learning algorithms after visiting his group. Prof. Boyd's book and the CVX toolbox on convex optimization provide me many insights and facilitate in conducting the work in this thesis. I would also like to thank Prof. Jieping Ye for his helpful comments and insights on improving the quality of my research work. The toolbox, SLEP, developed in their group, also provides me much help in conducting experiments on testing sparse models. I am grateful to my thesis committee, Prof. Kwong Sak Leung, Prof. Laiwan Chan, and the external marker, Prof. Dit-Yan Yeung, as well as other professors or collaborators, Robert Tibshirani, Daphne Koller, Shenghuo Zhu, Kai Yu, Shuiwang Ji, for their helpful suggestions and discus-

sions on research work.

I am also grateful to my colleagues in our group and my friends. I like to thank Zenglin Xu and Kaizhu Huang for the constructive discussions on the thesis work and the comments or helps from Xinyu Chen, Hongbo Deng, Patrick Lau, Zhengjiang Lin, Hao Ma, Xin Xin, Haixuan Yang, Wujie Zheng, Zibin Zheng, Chao Zhou, Yangfan Zhou, Jianke Zhu, ect.

Finally, I am deeply grateful to my parents for their unbending support and constant encouragement in my study. Without their encouragement, my Ph. D. study would not be started and the thesis would not be completed.

Contents

List of Figures

List of Tables

Chapter 1

Introduction

Machine learning is a research area that focuses on designing and developing algorithms for computers to evolve behaviors based on empirical data [114]. A center problem in machine learning is to automatically learn to recognize complex patterns and make intelligent decisions based on data. Due to their effectiveness and good performance in practice, many machine learning algorithms [67, 132, 160] have been developed and largely employed to solving problems in applications of computer vision [70, 123], pattern recognition [53, 144], search engines [85, 128], medical diagnosis [75, 76], bioinformatics [67, 87, 93], etc.

Recently, as large-scale datasets become popular, building parsimonious models and developing efficient algorithms are essentially important to machine learning applications. Hence, in this thesis, we develop several models to solve the large-scale learning problems. The work ranges from feature selection to learning from unlabeled data and learning data similarity representation. More specially, we focus on the key research problems in three areas:

Online learning: In some applications, e.g., web applications, training data are large in volume and may appear sequentially. Developing efficient algorithms to learn these kinds of data becomes essentially important. Online learning is a

very suitable learning paradigm for these applications due to its well-scaling capability and good general performance. The center problem of online learning algorithms is to make decisions about the present based only on the knowledge of the past when samples sequentially become available, or to update the function weights as samples come sequentially. How to design online learning algorithms for parsimonious models is very critical for large-scale learning.

Semi-supervised learning: Another problem in learning large-scale datasets lies in the deficiency of labeled data and the abundance of unlabeled data. This is a common occurrence in many real-world applications since labeling data is costly and needs experts' efforts. Meanwhile, unlabeled data are easily collected and they are usually embedded with useful information, e.g., data distribution or structure information. How to utilize the unlabeled data and the precious labeled data to seek effective rules is very important.

Multiple kernel learning: A center problem in machine learning is to design a suitable measurement to represent the data similarity, so as to perform the tasks of classification, regression, etc. This is especially essential in kernel machines, e.g., SVMs [144]. To achieve this goal, learning a good kernel representation provides an effective tool. A successful paradigm of learning kernels is to seek a combination of base kernel functions/matrices that maximizes a generalized performance measure [137]. As the real-world applications become more and more complicated, how to design more accurate kernel representation for the applications is crucial.

1.1 Contributions

This thesis aims at developing efficient and effective machine learning algorithms for solving large-scale learning applications. To this purpose, we focus on the challenging problems in the above-mentioned three areas and develop several novel algorithms. The main contributions of this thesis include:

- **Proposing online learning algorithms**
  - In this part, we have proposed the first online learning framework for two kinds of feature selection models. The first one is the online learning framework to solve the group lasso model, which can fit the data while selecting the explanatory factors in a group manner [184]. The second one is the online learning framework to solve the multi-task feature selection problem. Both algorithms update the weights of the models as data come sequentially.
  - A main advantage of the proposed algorithms is its efficiency due to our derived closed-form solutions in updating the corresponding model weights. At each iteration, the online learning algorithms just need $\mathcal{O}(d)$ time complexity and memory cost for group lasso and need $\mathcal{O}(d \times Q)$ complexity for multi-task feature selection, where d is the number of dimensions and Q is the number of tasks.
  - We also provide the regret bounds for the algorithms to reveal the property of the online learning algorithms from the theoretical perspective.
- **Proposing a general semi-supervised learning framework**

  - We propose a maximum margin semi-supervised learning framework to seek an inductive rule from both labeled and unlabeled data. In our framework, the unlabeled data may be a mixture of relevant and irrelevant data. The relevant data are drawn from the same distribution as the labeled data, while the irrelevant data are drawn from distribution(s) different from that of the labeled data. In this setting, we develop a *tri-class support vector machine* (3C-SVM) to distinguish the irrelevant data while classifying the relevant data. By adopting a novel min-loss function and following the maximum entropy principle, we can achieve this goal.
  - The proposed 3C-SVM generalizes several popular maximum margin classifiers, including standard SVMs, Semi-supervised SVMs (S^3VMs) [15, 84], and SVMs learned from universum ($\mathcal{U}$-SVMs) [161, 165]. The theoretical analysis on the 3C-SVM has been provided to indicate how the irrelevant data play the role of seeking a good subspace and to explain why the 3C-SVM works.
  - An efficient algorithm is proposed to solve the 3C-SVM by the concave-convex procedure (CCCP) [185]. Hence, the 3C-SVM just needs to solve several quadratic programming (QP) problems, which has the same worst case time complexity as that of S^3VMs [36].

- **Proposing a sparse generalized multiple kernel learning (GMKL) model**
  - We propose a generalized MKL model which includes the L_1-MKL [96] and the L_p-MKL [89, 90] as its special cases and conquers the insufficiency of these previously proposed MKL models.
  - We provide theoretical analysis of the GMKL on why

it contains sparse solutions with the grouping effect. This guarantees the favorite properties of the GMKL.

– Since the GMKL can be transformed into a convex-concave optimization problem, which the global optimal solution is guaranteed. We propose a very efficient method, the level method, to solve it and analyze its convergence rate. Consequently, the GMKL is enabled with its potential to solve large scale datasets.

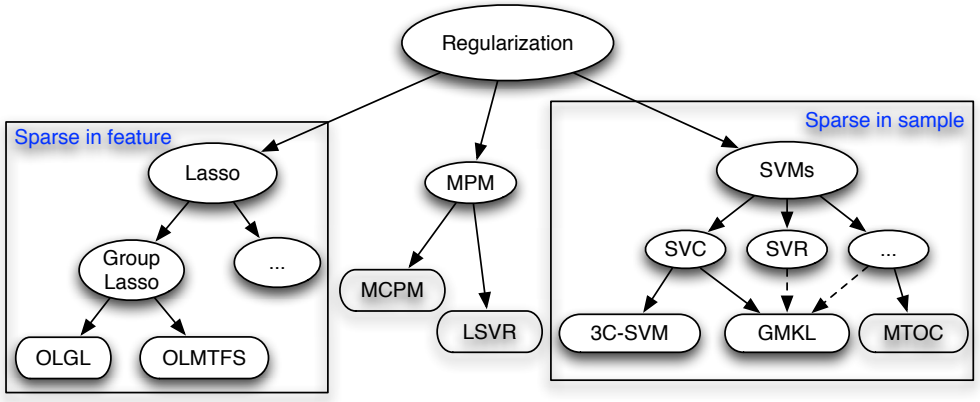


Figure 1.1: Illustration of the sparse models and our derived models.

In summary, Figure 1.1 gives an overview of the sparse models and seven models that we have developed in recent years. In the thesis, we only present four of them, which are especially promising for solving large-scale learning applications. They are online learning for group lasso (OLGL), online learning for multi-task feature selection (OLMTFS), the 3C-SVM, and GMKL. For other three models, they are multi-task for one-class classification (MTOC) [179], Minimax clustering probability machine (MCPM) [175], Localized support vector regression (LSVR) [176]. We do not include them in the thesis because they are not sparse models or not efficient for solving large-scale applica-

tions. The thesis also omits parts of our work in web applications [177, 178], and some other collaborated work on multiple kernel learning [171, 172].

1.2 Notation

In order to make the notations in the whole thesis consistent, we define the mathematical symbols in the following table.

Table 1.1: Symbols used in the thesis.

Symbol	Description
$\mathbf{K}$, $\mathbf{W}$	Bold capital letters indicate matrices.
$\mathbf{w}$, $\boldsymbol{\alpha}$	Bold small letters indicate vectors.
$\mathbf{1}_m$ ($\mathbf{0}_m$)	An m-dimensional vector with each element being 1 (0)
$\mathcal{X}$, $\mathbb{R}$	Calligraphic or blackboard bold fonts Letters indicate sets.
$\mathbb{R}^n$	An n-dimensional real space
$\mathbf{z} \in \mathbb{R}^n_+$	An n-dimensional vector with $z_i \geq 0$, for $i = 1, \ldots, n$.
$^\top$	Transpose operator
$\langle \mathbf{x}, \mathbf{y} \rangle_{\mathcal{H}}$	The inner product of $\mathbf{x}$ and $\mathbf{y}$ in the space $\mathcal{H}$.
$d(\mathcal{H})$	The dimension of the space $\mathcal{H}$.
$\mathbf{X} \succeq \mathbf{0}$	A positive semi-definite matrix
$\circ$	The Hadamard product or elementwise product
$\mathbf{W}_{i\bullet}^\top \in \mathbb{R}^Q$	The i-th row of $\mathbf{W}$ consists of Q elements.
$\mathbf{W}_{\bullet j} \in \mathbb{R}^d$	The j-th column of $\mathbf{W}$ consists of d elements.
$\|\mathbf{W}\|_F$	The Frobenius norm on $\mathbf{W}$, $\|\mathbf{W}\|_F = \sqrt{\sum_{i=1}^d \sum_{j=1}^Q W_{ij}^2}$
$[a]_+$	$\max\{0, a\}$

1.3 Scope

This thesis states and refers to the learning first as statistical learning. Especially, the center is sparse learning under the regularization framework, which appears to be the current main trend of learning approaches. The corresponding discussions on our work include online learning for feature selection, semi-supervised learning in a general unlabeled data assumption, and multiple kernel learning for data similarity measurement.

1.4 Organization

The rest of this thesis is organized as follows.

- Chapter 2

  We review the background of supervised learning. Especially, we focus on elaborating two families of well-known sparse learning models under the regularization framework and three hot topics in current machine learning area.

- Chapter 3

  We propose the first online learning framework for group lasso, which can fit the data and select the important explanatory in a group manner. We present the algorithmic framework, the closed-form solutions for variant group lasso models, and the average regret bound. Experiments on both synthetic and real-world datasets are performed to demonstrate the merits of the proposed learning framework.

- Chapter 4

  We propose the first online learning framework for multi-task feature selection models, which can select the important features across related tasks and important tasks that dominate the selected features. We present the algorithm

framework, the closed-form solutions for variant group lasso models, and the average regret bound. A series of detailed experiments is conducted to show the merits of the proposed online learning algorithms.

- Chapter 5

  We develop a novel maximum margin classifier, named the tri-class support vector machine (3C-SVM), to learn the inductive rule from both labeled and unlabeled data. We present the insight of the model in theoretical perspective, the properties of the model, and a series of experiments to demonstrate the advantages of the proposed 3C-SVM.

- Chapter 6

  We propose a generalized MKL (GMKL) model by introducing an elastic net-type constraint on the kernel weights. We provide the theoretical analysis on the properties of GMKL and the convergence rate of the proposing level method to solve it. Results on a series of experiments on both synthetic and real-world datasets are reported to show the effectiveness and efficiency of the proposed GMKL.

- Chapter 7

  We summarize this thesis and discuss some future work.

We try to make each of these chapters self-contained. Therefore, in several chapters, some critical contents, e.g., model definitions, algorithm framework, or illustrative figures, having appeared in previous chapters, may be briefly reiterated.

□ **End of chapter.**

Chapter 2

Background

In this chapter, we first review supervised learning and the regularization framework in Sec. 2.1. After that, in Sec. 2.2, we review two families of well-known sparse models, Lasso [155] and Support Vector Machines (SVMs) [160], under this regularization framework. Finally, we review three key learning paradigms, online learning, semi-supervised learning, and multiple kernel learning in Sec. 2.3, Sec. 2.4, and Sec. 2.5, respectively.

2.1 Supervised Learning Under Regularization

In supervised learning, we are given a set of N independent and identically distributed (i.i.d.) paired data sampled from a fixed but unknown distribution $\mathcal{P}$ over $\mathcal{X} \times \mathcal{Y}$ as

$$\mathcal{D} = \{(\mathbf{x}_i, y_i)\}_{i=1}^{N}, \quad \mathbf{x}_i \in \mathcal{X}, \quad y_i \in \mathcal{Y}. \tag{2.1}$$

The input space $\mathcal{X}$ is an arbitrary set, usually $\mathcal{X} = \mathbb{R}^d$, while the output space $\mathcal{Y}$ is a small number of discrete classes for classification problems and $\mathcal{Y} \in \mathbb{R}$ for regression problem. This is very common in many real-world applications. For example, in handwritten character recognition [69, 137], $\mathcal{X}$ is the set of

images of letters and $\mathcal{Y}$ is the alphabet; in predicting housing values [66], $\mathcal{X}$ is a set of features about the region and $\mathcal{Y}$ is the median value of the house.

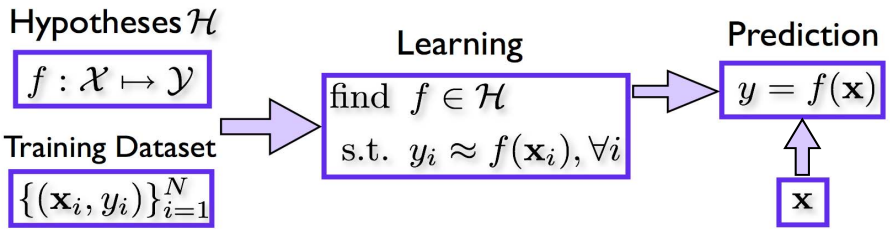


Figure 2.1: Demonstration of supervised learning procedure.

The objective of supervised learning is to seek a function $f : \mathcal{X} \mapsto \mathcal{Y}$ that maps inputs $\mathbf{x} \in \mathcal{X}$ to outputs $y \in \mathcal{Y}$ while $f(\mathbf{x})$ approximates y on new samples from the distribution $(\mathbf{x}, y) \sim \mathcal{P}$. The diagram in Fig. 2.1 summarizes the flow of the supervised learning setting.

The problem of supervised learning has a long history and highly developed theory and practice; see for example [53, 67, 114, 160]. The two most important factors of variation of supervised learning algorithms are the hypothesis class $\mathcal{H}$ and the criterion for selection of f from $\mathcal{H}$ given the training data. A standard criterion is to quantify what it means for $f(\mathbf{x})$ to approximate y, which is measured by the expected error of the approximation by the risk functional $\mathcal{R}^{\ell}_{\mathcal{P}}[f]$ defined as

$$\mathcal{R}^{\ell}_{\mathcal{P}}[f] = \mathbf{E}_{(\mathbf{x},y)\sim\mathcal{P}}[\ell(\mathbf{x}, y, f(\mathbf{x}))], \tag{2.2}$$

where the loss function $\ell : \mathcal{X} \times \mathcal{Y} \times \mathcal{Y} \to \mathbb{R}^+$ measures the penalty for predicting $f(\mathbf{x})$ on the sample $(\mathbf{x}, y)$. In general, $\ell(\mathbf{x}, y, \hat{y}) = 0$ if $y = \hat{y}$. Many different loss functions can be adopted for the supervised learning. We will elaborate them in Section 2.1.2.

Since the distribution $\mathcal{P}$ is generally unknown, one estimates

the risk of f using its empirical risk $\mathcal{R}^{\ell}_{\mathcal{D}}$, computed on the training set $\mathcal{D}$ by

$$\mathcal{R}^{\ell}_{\mathcal{D}}[f] = \frac{1}{N}\sum_{i=1}^{N} \ell(\mathbf{x}_i, y_i, f(\mathbf{x}_i)), \tag{2.3}$$

$\mathcal{R}^{\ell}_{\mathcal{D}}[f]$ is called the training error or training loss.

A criterion to find f may be simply selecting a hypothesis with the lowest risk

$$f^* = \arg\min_{f\in\mathcal{H}} \mathcal{R}^{\ell}_{\mathcal{D}}[f].$$

However, this is generally not a good idea. For example, if $\mathcal{X} = \mathbb{R}, \mathcal{Y} = \mathbb{R}$ and $\mathcal{H}$ includes all polynomials of degree $N-1$, we can always find a polynomial f that passes through all the sample points $(\mathbf{x}_i, y_i)$, $i = 1\ldots, N$, assuming that all the $\mathbf{x}_i$ are unique. This polynomial is very likely to overfit the training data. That is, it has zero empirical risk, but high actual risk.

The key to selecting a good hypothesis is to trade-off complexity of class $\mathcal{H}$ (e.g. the degree of the polynomial) with the error on the training data as measured by empirical risk $\mathcal{R}^{\ell}_{\mathcal{D}}[f]$. For a vast majority of supervised learning algorithms, this fundamental balance is achieved by minimizing the weighted combination of the two criteria:

$$f^* = \arg\min_{f\in\mathcal{H}} \left(R[f] + C\mathcal{R}^{\ell}_{\mathcal{D}}[f]\right), \tag{2.4}$$

where $R[f]$, is often called regularization, measuring the inherent dimension or complexity of f, and $C \geq 0$ is a trade-off parameter. The setting of complexity measure $R[f]$ can be referred to [67, 160], and we will discuss it in the following section.

2.1.1 Regularization

The regularization framework is common in machine learning, including Support Vector Machines [160], Logistic Regression [67],

and Lasso [155], etc. In (2.4), the first term is the regularization term, which measures the inherent dimension or complexity of f. Usually, one chooses the (generalized) linear model as the function class by taking consideration of accuracy, efficiency, and extensibility. Typically, the decision function is defined as

$$f_{\boldsymbol{\vartheta}}(\mathbf{x}) = \mathbf{w}^\top \mathbf{x} + b, \tag{2.5}$$

where the parameter $\boldsymbol{\vartheta} = (\mathbf{w}, b)$ consists of the function weight $\mathbf{w} \in \mathbb{R}^d$ and the bias term b. They have to be learned from the training data in (2.1).

For the linear model, a typical regularization of $R[f]$ is $\frac{1}{2}\|\mathbf{w}\|^2$, the square of the L_2-norm on the weights. This can be traced back to the Tikhonov regularization, or *ridge regression*, which introduces the regularization on the function weight to solve ill-posed problems in the 1940's [159]. The standard ridge regression seeks the function weights by minimizing the square loss and the square regularization

$$\min_{\mathbf{w}} \quad C \sum_{i=1}^{N} (\mathbf{w}^\top \mathbf{x}_i + b - \mathbf{y}_i)^2 + \frac{1}{2}\|\mathbf{w}\|^2 \tag{2.6}$$

Many statistical learning models follow the above framework, but with variants. For example, SVMs [160] take the same regularization as ridge regression, but adopt a different loss function, which can introduce sparse solutions in the sample level. Lasso [67] adopts the same loss function as ridge regression, but applies the L_1-norm regularization on the weights, which can introduce sparse solutions in the feature level. We will introduce them with more details in Sec. 2.2.

2.1.2 Loss Functions

The second term in (2.4) defines the loss, which measures the empirical risk of the model. For different problems, usually,

different loss functions are adopted. In the following, we will explain several famous loss functions in the literature.

The most common loss for classification is the 0/1-loss. It is defined as

$$\ell^{0/1}(\mathbf{x}, y, f(\mathbf{x})) \equiv \mathbf{I}(yf(\mathbf{x}) < 0), \tag{2.7}$$

where $\mathbf{I}(\cdot)$ denotes the indicator function as $\mathbf{I}(\text{true}) = 1$ and $\mathbf{I}(\text{false}) = 0$. That is, when the loss in (2.3) is the 0/1-loss, it simply defines the proportion of training samples that f misclassifies.

However, minimizing the 0/1-risk is generally a very difficult problem with multiple maxima for any large class $\mathcal{H}$ since this loss function is discrete [53]. The standard solution is minimizing an upper bound on the 0/1-loss, $\bar{\ell}(\mathbf{x}, y, f(\mathbf{x})) \geq \ell(\mathbf{x}, y, f(\mathbf{x}))$. In addition to computational advantages of this approach, there are statistical benefits of minimizing a *convex* upper bound. Logistic regression and support vector machine are two of the primary classification methods which adopt this idea, but differ primarily in their choice of the upper bound on the training 0/1-loss. For example, in logistic regression, the loss function is defined as a logit loss [71]

$$\ell^{\text{logit}}(\mathbf{x}, y, f(\mathbf{x})) = \log_2 \left(1 + \exp(-yf(\mathbf{x}))\right), \tag{2.8}$$

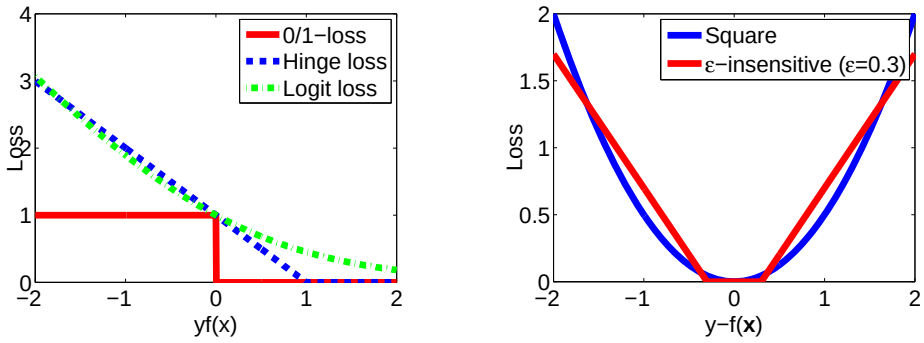
and in support vector machine, the hinge loss is adopted [160]

$$\ell^{\text{hinge}}(\mathbf{x}, y, f(\mathbf{x})) = \max\{1 - yf(\mathbf{x}), 0\}. \tag{2.9}$$

Here, it should be noted that y is selected from $\{-1, 1\}$. Figure 2.2(a) illustrates the difference among these three loss functions.

In regression, various loss functions can be adopted. Actually, different loss functions correspond to the noise with different distributions [160]. For example, the standard square loss corresponds to the Gaussian noise and it is defined as

$$\ell^{\text{square}}(\mathbf{x}, y, f(\mathbf{x})) = (y - f(\mathbf{x}))^2. \tag{2.10}$$



(a) Loss for classification (b) Loss for regression

Figure 2.2: Demonstration of loss functions. Fig. 2.2(a) illustrates the difference among the 0/1-loss, hinge loss, and the logit loss for classification. It is noted that the logit loss is a smoothed version of the hinge loss. Fig. 2.2(b) illustrate the difference between the square loss and the ε-insensitive loss for regression. Note that, the square loss is smoothed while the ε-insensitive loss is non-smoothed.

In support vector regression [160], the ε-insensitive loss function is adopted

$$\ell^{\varepsilon}(\mathbf{x}, y, f(\mathbf{x})) = \max\{|y - f(\mathbf{x})| - \varepsilon, 0\}. \tag{2.11}$$

The ε-insensitive loss function does not penalize those data points with deviation from the target value y being smaller than ε, a small constant. This loss function enjoys similar property of the hinge loss and introduces sparse solutions. Figure 2.2(b) illustrates the difference between the hinge loss and the square loss.

2.2 Sparse Models

As large-scale datasets become popular, developing parsimonious models contains several advantages. First, the succinct models are easier for interpretation and more helpful for revealing the characteristics of the data. Second, they may help for

saving the storage cost and reducing the cost in testing. Third, they may achieve good performance due to discarding the noisy information in the large-scale datasets.

As indicated in [58, 200], in the L_p penalty family, the estimates have a parsimonious property (with some components being exactly zero) when $p \leq 1$, while the optimization problem is convex only for $p \geq 1$. Hence, only the lasso penalty ($p = 1$) yields sparse solutions while maintaining the convex property. Adopting L_1 form is a way to yield sparse solutions and is promising for solving large-scale datasets.

Referring back to (2.4), since it contains two terms, by adopting different L_1 forms, the previously proposed sparse models can be categorized into two families. One is the lasso model, imposing L_1-norm regularization on the function weights to introduce the sparsity in the feature level. The other is support vector machine, adopting the hinge loss to introduce the sparsity in the sample level. We will introduce them in the following section.

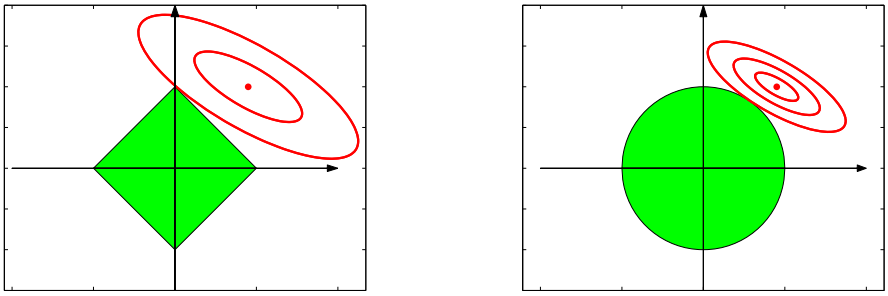
2.2.1 Lasso and its Extensions

Lasso is a shrinkage and selection method for linear regression. It minimizes the usual sum of squared errors, with a regularizer on the sum of the absolute values of the function weights [155]. The definition of the lasso model corresponds to

$$\min_{\mathbf{w},b} \quad \sum_{i=1}^{N} \left(y_i - b - \sum_{j=1}^{d} w_j x_{ij} \right)^2 + \lambda \sum_{j=1}^{d} |w_j|, \quad (2.12)$$

where λ plays the trade-off term similar to C in SVMs. Obviously, a larger λ will make all function weights become zero, while a small λ can keep all function weights. As shown in Fig. 2.3(a), the optimal solution of the lasso model is usually hit on the corner of the contour. This parsimonious property

helps lasso to select important features while solving the regression problem. Hence, we classify the lasso model as those sparse models achieving sparsity in the feature level. Figure 2.3 illustrates the difference of the lasso solution and the ridge regression solution in a two dimensional space. The lasso solution corresponds to the first place that the contours touch the square, and this sometimes occurs at the corner, yielding a zero weight. On the contrary, for ridge regression, there are no corners for the contours to hit and thus resulting in no zero solutions [155].



(a) Lasso (b) Ridge regression

Figure 2.3: Illustration of solutions on the lasso and ridge regression.

Due to its parsimonious property, lasso has been successfully applied in those applications with data in large dimensions and small training samples, e.g., species' prediction [200], microarray data analysis [60], coeliac data analysis [168], etc. There are also many extensions on lasso. We can categorize them into two directions. One direction is to extend the model to improve its learning and interpretable abilities. The other direction is to speed up the training procedure of the lasso model. Here, we first review those models extending the lasso idea to yield parsimonious solutions. Typical ones include the following models:

Elastic Net: This model extends the lasso model by introducing a new regularization, a linear combination of lasso regu-

larizer and ridge regularizer [200]. Hence, the elastic net not only yields sparse solutions as the lasso, but also encourages a grouping effect, where strongly correlated predictors tend to be in or out of the model altogether [200].

Group Lasso: This model extends the lasso model by introducing the L_1-norm on the L_2-norm of the function weights to select the explanatory factors in a group manner [184]. The original model is to solve the regression problem. Due to its favorite properties, group lasso has been intensively studied in statistics and machine learning in recent years [8, 131, 184]. Various discussions and work include the group lasso for logistic regression [112], the group lasso for generalized linear models [134], the group lasso with overlap between groups [80], and the sparse group lasso [59], etc.

Other models with similar ideas of group lasso are also proposed, e.g., the *Component Selection and Smoothing Operator* (COSSO) [103], the *Composite Absolute Penalites* (CAP) approach [191], etc. The COSSO selects groups of predictors corresponding to sets of basis functions for smoothing splines by imposing the regularization as the square-root of the integrated squared second derivative of a spline function (a linear combination of the basis functions) [103]. The CAP approach generalizes the group lasso by introducing a generalized group penalty instead of L_2-norm on the function weights [191].

The original lasso model can be solved by a Quadratic Programming (QP) problem, which can be solved by a standard optimization toolbox. It can also be solved by the coordinate descent method proposed in [155]. Now, we review part of the proposed methods in speeding up the lasso model. These methods include the following:

LARs: Least Angle Regression (LAR) is a promising technique

for variable selection applications, offering a nice alternative to stepwise regression. It provides an explanation for the similar behavior of lasso and forward stagewise regression. More importantly, it provides a fast implementation of both [54]. The algorithm exploits the fact that the lasso contains piecewise linear solution paths, which leads to an algorithm with the same computational cost as the full least-square fit on the data.

Glmnet: It is a very efficient algorithm for estimation of generalized linear models with convex penalties by cyclical coordinate descent to compute along a regularization path [60]. This method can handle large datasets and can deal with sparse features efficiently.

The idea of L_1-regularization in statistics is the well-known "least absolute shrinkage and selection operator", Lasso algorithm [155]. This idea also comes up in signal processing in basis pursuit [34, 49], signal recovery from incomplete measurements [24, 25, 48, 50], portfolio optimization [108], sparse principle component analysis [23, 45, 186], computer vision [110], and neural computation [101, 129].

2.2.2 Support Vector Machines

Support vector machines (SVMs) contain solid theoretical foundations and have demonstrated outstanding performance in many applications [160]. Here, SVMs include all models based on the concept of the maximum margin, while SVM only indicate the classification model.

By adopting the hinge loss (2.9) and based on the "margin"

of confidence of $f_{\boldsymbol{\vartheta}}$, one can define the objective of the SVM as

$$\begin{aligned} \min_{\mathbf{w},b,\boldsymbol{\xi}\geq 0} \quad & C\sum_{i=1}^{N}\xi_i + \frac{1}{2}\|\mathbf{w}\|^2 \\ \text{s.t.} \quad & y_i(\mathbf{w}^\top\phi(\mathbf{x}_i) - b) \geq 1-\xi, \\ & \xi_i \geq 0, i = 1,\ldots,N, \end{aligned} \tag{2.13}$$

where C is a constant trading off the error and the regularization term. Different from common linear models, the SVM adopts a generalized linear model, which is fulfilled by a mapping function, $\phi : \mathbf{R}^d \mapsto \mathbb{R}^f$. Here the labels y_i are either $+1$ or -1 for a binary classification problem. The above convex optimization is usually solved by the Lagrange multiplier method [22, 160]. This leads to solving its dual form as a Quadratic Programming (QP) problem as follows:

$$\begin{aligned} \max_{\boldsymbol{\alpha}} \quad & \mathbf{1}_N^\top\boldsymbol{\alpha} - \frac{1}{2}(\boldsymbol{\alpha}\circ\mathbf{y})^\top\mathbf{K}(\boldsymbol{\alpha}\circ\mathbf{y}) \\ \text{s.t.} \quad & \mathbf{y}^\top\boldsymbol{\alpha} = 0 \\ & \mathbf{0}_N \leq \boldsymbol{\alpha} \leq C\mathbf{1}_N, \end{aligned} \tag{2.14}$$

where $\mathbf{K}$ defines the kernel matrix as $K_{ij} = \phi(\mathbf{x}_i)^\top\phi(\mathbf{x}_j)$. Many kernel functions can be adopted for calculating $\mathbf{K}$ and usually they can be learned from the data [137].

The above QP problem can be solved efficiently by a standard QP package or by other methods, e.g., the Sequential Minimal Optimization (SMO) methods [124], a general active set method [135]. The decision function of SVMs is represented as

$$f(\mathbf{x}) = \sum_{i=1}^{N} y_i\alpha_i^\star K(\mathbf{x}_i,\mathbf{x}) + b^\star, \tag{2.15}$$

where $\boldsymbol{\alpha}^\star$ is the optimal solution obtained by solving (2.14), while $b^\star$ is obtained from the equality constraints.

There are many extensions based on the framework in (2.13). They include

Support Vector Regression (SVR): The SVM have been extended to solve the regression problem by adopting the ε-insensitive loss function [147]. An illustration of the ε-insensitive loss function is shown in Fig. 2.2(b). By adopting this L_1 loss, SVR can also yield sparse solutions. Due to its solid theoretical background and effectiveness, SVR has been applied in various applications, e.g., control in chaotic systems [111], time series prediction [116], and financial market prediction [174, 176], etc.

ν-SVM: The ν-SVM has been derived so that the soft margin has to lie in the range of zero and one [33]. The parameter ν is not controlling the trade-off between the training error and the generalization error as that of C. Instead, it plays two roles: It is an upper bound on the fraction of margin errors and is the lower bound on the fraction of support vectors [40, 138]. Hence, ν is selected from the range of 0 and 1. Choosing the parameter ν is easy and intuitive.

One-class SVM: There are two kinds of SVM derivatives to solve the one-class classification problem. One idea is the *Support Vector Domain Description* (SVDD): It maps the data into a feature space and seeks a sphere with minimum volume containing all or most of the samples in the target class [154]. When a future point falls in the ball, it is deemed to be a "target" object; otherwise, it is an outlier object. Another idea is the ν-SVM: This model maps the data into a feature space and aims to separate the given data from the origin with a maximum margin. The algorithm returns a decision function f taking the value $+1$ in a "small" region capturing most of the data points in the target class, and -1 elsewhere [139]. The latter approach introduces a favorable parameter $\nu \in (0, 1]$, which can control the fraction of outliers and the fraction of support vec-

tors [136]. This model is termed as one-class ν-SVM. The above two approaches can be transformed and represented in a kernel form and the SVDD also can be introduced by the ν parameter [136].

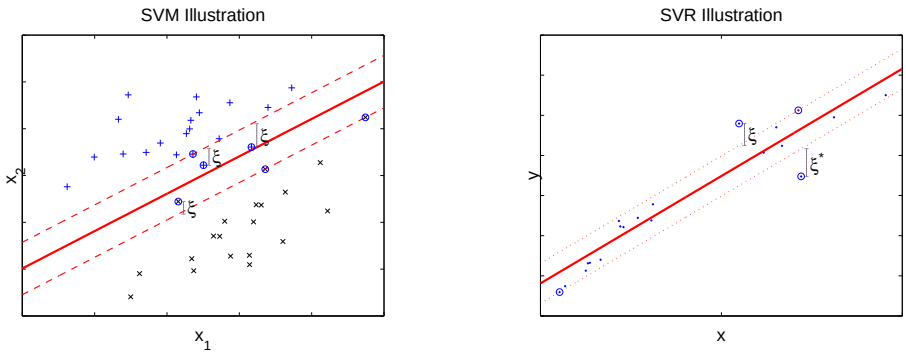
SVM for semi-supervised learning: There are two models to extend SVM for solving the semi-supervised learning task, transductive SVMs (TSVM) and semi-supervised support vector machine (S^3VM) [36, 84]. They seek the largest separation between labeled and unlabeled data through regularization. Differently, TSVM utilizes the labeled data and test data to improve the prediction on the test data; while S^3VM seeks an inductive rule from the labeled and unlabeled data with the aim of improving the performance on unseen coming data.

Other than the introduction in the above, there are many theoretical results with various applications in SVMs. Interested readers can refer to the books [28, 41, 137, 144, 160, 161] and the references therein.

It is noted that SVM and the corresponding extensions usually achieve sparsity in the sample level. That is the decision function in (2.15) is represented by several important samples, called *support vectors*, corresponding to $\alpha_i \neq 0$. This helps in reducing the cost in the test procedure. Here, we give an illustration example of SVMs for classification and regression in Fig. 2.4.

2.3 Online Learning

As large-scale and dynamic datasets become popular, online learning algorithms become a very promising direction in machine learning. The problem of online learning algorithms is to make decisions about the present based only on the knowledge of



(a) SVM (b) SVR

Figure 2.4: Illustration of SVM and SVR with sparsity in the sample level.

the past when samples sequentially become available. More specially, a sequence of i.i.d. samples $(\mathbf{x}_1, y_1), (\mathbf{x}_2, y_2), \ldots$ are drawn to calculate a sequence of the weights $\boldsymbol{\vartheta}_1, \boldsymbol{\vartheta}_2, \ldots$. At time t, given the most up-to-date weight vector $\boldsymbol{\vartheta}_t$, as $(\mathbf{x}_t, y_t)$ is available, we can evaluate the loss $\ell(\mathbf{x}_t, y_t, \boldsymbol{\vartheta}_t)$, and its subgradient $\mathbf{u}_t \in \partial\ell(\mathbf{x}_t, y_t, \boldsymbol{\vartheta}_t)$, where $\partial\ell(\mathbf{x}, y, \boldsymbol{\vartheta})$ denotes the subgradient of $\ell(\mathbf{x}, y, \boldsymbol{\vartheta})$ with respect to $\boldsymbol{\vartheta}$. The weight $\boldsymbol{\vartheta}_{t+1}$ is updated based on these information, and even the information of second-order derivatives if the loss functions are smooth.

The most widely used online algorithm is the *stochastic gradient descent* (SGD) method [18]. The *online gradient descent* algorithm updates the parameter by

$$\boldsymbol{\vartheta}_{t+1} = \boldsymbol{\vartheta}_t - \eta_t \mathbf{u}_t, \tag{2.16}$$

where η_t is an appropriate stepsize. As the regularization is included, the objective in (2.4) for an online learning algorithm becomes

$$\tilde{R}(\boldsymbol{\vartheta}) = R(\boldsymbol{\vartheta}) + \mathbf{E}_{\boldsymbol{\vartheta}}\ell(\mathbf{x}, y, \boldsymbol{\vartheta}). \tag{2.17}$$

When a general regularization is considered, e.g., $R(\boldsymbol{\vartheta}) = I_C(\boldsymbol{\vartheta}) + \psi(\boldsymbol{\vartheta})$ ($I_C(\boldsymbol{\vartheta})$ is a "hard" set constraint and $\psi(\boldsymbol{\vartheta})$ is a "soft" reg-

ularization), the updating rule of SGD becomes

$$\boldsymbol{\vartheta}_{t+1} = \Pi_C \left(\boldsymbol{\vartheta}_t - \eta_t (\mathbf{u}_t + \boldsymbol{\mu}_t) \right), \tag{2.18}$$

where $\boldsymbol{\mu}_t$ is a subgradient of ψ at $\boldsymbol{\vartheta}_t$, and $\Pi_C(\cdot)$ denotes Euclidean projection onto the set C. The SGD method belongs to the family of *Stochastic approximation*, which was first developed in [88, 133].

The objective of online learning algorithms is to generate a sequence $\{\boldsymbol{\vartheta}_t\}_{t=1}^{\infty}$ such that

$$\lim_{t\to\infty} \mathbf{E}\tilde{R}(\boldsymbol{\vartheta}_t) = \tilde{R}(\boldsymbol{\vartheta}^\star),$$

with reasonable convergence rate. In the above, an optimal solution $\boldsymbol{\vartheta}^\star$ to the problem (2.17) is assumed existing.

There are several directions in developing online learning algorithms. One direction is to develop promising algorithms with improving convergence rate, e.g., [68, 140, 199]. The other direction is to exploit the problem structure, especially for problems with explicit regularization, to gain advantages of the models [52, 72, 97, 169]. Due to their capability of scaling well for very large datasets and their good generalization performances observed in practice [20, 188], online learning algorithms have been actively developed, forming an attractive and promising area in machine learning.

2.4 Semi-supervised Learning

Semi-supervised learning is an active research topic in recent years. It considers the problem of learning from both labeled and unlabeled data. There are two types of semi-supervised learning paradigms; inductive and transductive semi-supervised learning. Inductive semi-supervised learning works like an in-class exam, where the questions are not known in advance, and a student needs to prepare all possible questions [197]. Its goal is to derive

the inductive rule for predicting future test data. On the other-hand, transductive semi-supervised learning works like a take-home exam, where the student knows the exam questions and needs not prepare beyond those [197]. Its goal is to improve the performance of the test data by training on the labeled data and the test (unlabeled) data. Recently, many methods have been proposed for solving semi-supervised learning problems, such as EM with generative mixture models [120], co-training [119], Transductive Support Vector Machines [36, 84], and graph-based methods [6, 44, 193, 196], etc. Interested readers are referred to the books and the survey in [28, 195, 197] and the references therein.

2.5 Multiple Kernel Learning

Multiple kernel learning (MKL) has been an attractive topic in machine learning recently [96, 122]. It has been regarded as a promising technique for identifying the characteristics of multiple data sources or feature subsets and has been applied in a number of applications, such as genome fusion [95], splice site detection [151], image annotation [64], etc.

The problem of multiple kernel learning is to seek a combination of base kernel functions/matrices that maximizes a generalized performance measure. Typical measures studied for multiple kernel learning include maximum margin classification errors [9, 96, 198], kernel target alignment [42], Fisher discriminative analysis [182], etc.

There are two active research directions in multiple kernel learning. One is to improve the efficiency of MKL algorithms. Following the Semi-Definite Programming (SDP) algorithm proposed in the seminal work of [96], in [9], a block-norm regularization method based on Second Order Cone Programming (SOCP) was proposed in order to solve medium-scale problems. Due to

the high computation cost of SDP and SOCP, these methods cannot process large-scale datasets with large number of kernels and large number of training data. Recent studies suggest that wrapping-based approaches [130, 151, 170] are more efficient due to the efficiency of SVM solvers and the possible speeding-up techniques in seeking the kernel weights. In the wrapping-based methods, they first solve a classical SVM given the current solution of kernel weights, then a specific procedure is used to update the kernel weights. Many recent research efforts have been focused on updating the kernel weights [151, 170, 171].

The second direction is to improve the effectiveness of MKL models by exploring possible combinations of base kernels. The L_1-norm on the kernel weights, also known as the simplex constraint, is the first and mostly used constraint in MKL methods to seek the kernel weights. The advantage of the simplex constraint is that it yields a sparse solution, i.e., only a few base kernels carry significant weights. However, as argued in [89], the simplex constraint may discard complementary information when base kernels encode orthogonal information, and lead to suboptimal performance. To improve the accuracy, an L_2-norm constraint on the kernel weights, known as a ball constraint, is introduced in [89]. The L_2-norm constraint can be easily extended to the L_p-norm constraint on the kernel weights. The formulation is approximated by the second order Taylor expansion and is transformed into a convex optimization problem [90]. Another possible extension is to explore the grouping property or the mixed-norm combination, which is helpful when there are principal components among the base kernels [81, 153]. Other researchers also study the possibility of non-linear combination of kernels [38, 162]. Developing MKL models with clear interpretation continues to be a promising research direction.

□ **End of chapter.**

Chapter 3

Online Learning for Group Lasso

3.1 Introduction

Group lasso [184], a technique of selecting key explanatory factors in a grouped manner, is an important extension of lasso [155]. It has been successfully employed in a number of applications, such as birthweight prediction and gene finding [112, 184]. In these applications, data may either be dominated by k-th order polynomial expansions of some inputs or contain categorical features which are usually represented as groups of dummy variables [80, 112, 134]. Due to its advantages, group lasso has been intensively studied in statistics and machine learning [8, 184]. Extensions include the group lasso for logistic regression [112], the group lasso for generalized linear models [134], the group lasso with overlap between groups [80], etc.

Despite its success in the above applications, the original group lasso model and most of its extensions have several limitations which need to be addressed: (i) The models are learned by a batch-mode training. In the training process, data are given in advance, and then they are fed into a convex optimization problem which minimizes the empirical loss with a regularization that introduces the group sparsity. However, in real-world

applications, the training data may appear sequentially. (ii) Existing group lasso algorithms can only handle data up to several thousands of instances or features [112, 134, 184]. While in real-world applications, data can be in large volume, over millions in both of the sample size and the feature space. Previous group lasso algorithms will fail in this situation due to their inefficiency or poor scalability. (iii) The original group lasso can only yield solutions with sparsity in the group level. It usually lacks the ability in further finding the key factors in an important group. This is a non-trivial drawback for some real-world applications, where data may be explained by the key features within the important groups. Only seeking sparsity in the group level may lose some useful information that is important to accurately interpret the data.

To address the above problems caused by the batch-mode training and poor data scalability, we develop a novel and very efficient online learning algorithm for the group lasso, which updates the learning weight vector at each iteration by a closed-form solution based on the average of the previous subgradients. To the best of our knowledge, our algorithm is the first online algorithm for the group lasso. Our algorithm enjoys several good properties in terms of efficiency and effectiveness. First, the efficiency of the algorithm can be guaranteed by its low complexity in both memory space and time cost: At each iteration, the proposed algorithm only needs $\mathcal{O}(d)$ memory to store the required information and the updating process has a worst-case time complexity of $\mathcal{O}(d)$ in computation, where d is the number of features. Hence, our proposed algorithm has the potential to solve large-scale problems. Second, as the accuracy guarantee, we provide the convergence rate for both the regret bound and the bound of the learning weight vector for the proposed algorithm.

In order to seek the group lasso with more sparsity in both

the group level and the individual feature level, we successfully extend the algorithm to solve the *sparse group lasso* problem [59] and propose the *enhanced sparse group lasso* model. We further derive closed-form solutions to update the weight vectors in both models. Our algorithmic framework can also be easily extended to solve the group lasso with overlap and the graph lasso problems [80]. Therefore, this suggests the good applicability of our proposed algorithm in that it can be employed to solve a large family of group lasso algorithms. Finally, experiments on both synthetic and real-world datasets demonstrate the advantages of the proposed online algorithm.

3.2 Related Work

In the following, we mainly review the related work on online learning algorithms.

Online learning has been extensively studied in machine learning area in recent years [2, 19, 20, 39, 51, 57, 72, 141, 190, 199]. These methods can be cast into different categories. One family of online learning algorithms is based on the criterion of maximum margin [51, 141], which repeatedly chooses the hyperplane that correctly classifies the training samples with the maximum margin or updates the decision boundary when a new sample is misclassified or when its classification score does not exceed some predefined margin. Another family of online learning algorithms is solved by the stochastic gradient method [20, 199], where the weight vector is updated based on the subgradient of the coming sample and projected back to the constraint space if needed. An attractive feature of stochastic gradient decent methods is that their runtime may not depend at all on the number of examples [19, 142]. Although various online learning algorithms have been proposed, there is no online learning algorithm developed for the group lasso yet.

More recently, online learning algorithms on minimizing the summation of data fitting and L_1-regularization have been proposed to yield sparse solutions [12, 52, 97, 169]. These algorithms are very promising in real-world applications, especially for training large-scale datasets. In [97], a truncated gradient method is proposed to truncate the elements of the learning weight vector to 0 when they cross 0 after the stochastic gradient step. Experiments on data with over 10^7 samples and 10^9 features using about 10^{11} bytes are evaluated for that method [97]. In [52], a forward-backward splitting method (FOBOS) is studied for solving the regularized convex optimization problem, especially the lasso problem. The algorithm of FOBOS consists of two steps: performing an unconstrained gradient descent step first and then minimizing a regularization term while keeping the solution close to the result of the first phase. In [169], the regularized dual averaging method is proposed to solve the lasso problem, where the learning weight is updated based on the average of all calculated subgradients of the loss functions. The efficiency of the above methods motivates us to propose an online learning algorithm for the group lasso.

3.3 Group Lasso

Given a training dataset consisting of N independent and identically distributed observations, $\{\mathbf{z}_i = (\mathbf{x}_i, y_i)\}_{i=1}^N$, where $\mathbf{x}_i \in \mathbb{R}^d$ is a d-dimensional vector and $y_i \in \{-1, 1\}$ for the binary classification problem or $y_i \in \mathbb{R}$ for the regression problem. Suppose that these d features are divided into G groups with d_g, the number in g-th group. Hence, we can rewrite $\mathbf{x}_i = ({\mathbf{x}_i^1}^\top, \ldots, {\mathbf{x}_i^G}^\top)^\top$ with the group of variables $\mathbf{x}_i^g \in \mathbb{R}^{d_g}$, $g = 1, \ldots, G$. When $d_g = 1$ for all groups, the data do not form a group in the feature space.

The *lasso* algorithm [155] is a linear regression model that selects the variables individually and it cannot find the key fac-

tors in the grouped mode. Correspondingly, the *group lasso* algorithm [184] is proposed to select a subset of important factors for producing accurate prediction. Concretely, it is to seek the weight $\mathbf{w}$ and the bias b in $f(\mathbf{x}) = \sum_{g=1}^{G} \mathbf{w}^{g\top}\mathbf{x}^g + b$, by solving the following optimization problem:

$$\min_{\mathbf{w}} \quad \sum_{i=1}^{N} l(\mathbf{w},\ \mathbf{z}_i) + \Omega_\lambda(\mathbf{w}), \tag{3.1}$$

where $\mathbf{w} = (\mathbf{w}^{1\top}, \ldots, \mathbf{w}^{G\top})^\top$. In (3.1), since the bias usually can be absorbed by the weight without penalty, we only consider the optimization on the weight vector $\mathbf{w}$.

Various loss functions can be adopted for $l(\cdot)$ and they are usually assumed convex. They are

Squared loss: $l(\mathbf{w},\ \mathbf{z}) = \frac{1}{2}(y - \mathbf{w}^\top\mathbf{x})^2$. This loss is a standard loss which has been used in the original group lasso algorithm [184].

Logit loss: $l(\mathbf{w},\ \mathbf{z}) = \log(1+\exp(-y(\mathbf{w}^\top\mathbf{x})))$. This loss is used for binary classification problems [112].

In (3.1), $\Omega_\lambda(\cdot)$ defines the regularization on the weight. In the group lasso, the "groupwise" L_2-norm is adopted as the regularizer, i.e.,

$$\Omega_\lambda(\mathbf{w}) = \lambda \sum_{g=1}^{G} \sqrt{d_g}\|\mathbf{w}^g\|_2, \tag{3.2}$$

where the trade-off constant $\lambda \geq 0$ is to balance between the loss and the regularization term. The value $\sqrt{d_g}$ accounts for the varying group sizes and $\|\cdot\|_2$ is the Euclidean norm.

Remark 1. The regularizer in (3.2) makes the model act as the lasso at the group level: a large λ may make a whole group of predictors drop out of the model. As $d_g = 1$ for all the groups, the group lasso is equivalent to the lasso.

Remark 2. To introduce group sparsity, it is also possible to impose other joint regularization on the weight, e.g., the $L_{1,\infty}$-norm [127].

Remark 3. The group lasso regularizer has also been extended to use in Multiple Kernel Learning (MKL) [8]. The consistency analysis on the connection between the group lasso and MKL can be referred to [8].

Here, we can further introduce *sparse group lasso* as that in [59]

$$\Omega_{\lambda,\mathbf{r}}(\mathbf{w}) = \lambda \sum_{g=1}^{G} \left(\sqrt{d_g} \|\mathbf{w}^g\|_2 + r_g \|\mathbf{w}^g\|_1 \right), \tag{3.3}$$

where $r_g > 0$, for $g = 1, \ldots, G$, is a constant balancing the L_2-norm against the L_1-norm in each group. By imposing L_1-norm in each group, the sparse group lasso can further yield sparse solutions in the selected group.

To solve the optimization with the group lasso, various methods, e.g., Group LARs [184], block co-ordinate descent [112], active set algorithm [134], have been proposed. Some batch-mode training methods for group lasso penalties also have been proposed, e.g., [94, 107]. Interested readers can read the above papers and references therein.

3.4 Online Learning for Group Lasso

Inspired by recently developed first-order methods for optimizing composite functions [118] and the efficiency of the dual averaging method for minimizing the L_1-regularization in [169], we propose an online learning algorithm by adopting the dual averaging method to solve the group lasso, namely **DA-GL**. The algorithm is outlined in Algorithm 1. In this case, data come

Algorithm 1 Online learning algorithm for group lasso

Input:

- $\mathbf{w}_0 \in \mathbb{R}^d$, and a strongly convex function $h(\mathbf{w})$ with modulus 1 such that
$$\mathbf{w}_0 = \arg\min_{\mathbf{w}} h(\mathbf{w}) \in \arg\min_{\mathbf{w}} \Omega(\mathbf{w}) \,. \tag{3.4}$$
- Given const $\lambda > 0$ for the regularizer.
- Given const $\gamma > 0$ for the function $h(\mathbf{w})$.

Initialization: $\mathbf{w}_1 = \mathbf{w}_0$, $\bar{\mathbf{u}}_0 = \mathbf{0}$.

for $t = 1, 2, 3, \ldots$ **do**

1. Given the function l_t, compute the subgradient on $\mathbf{w}_t$, $\mathbf{u}_t \in \partial\, l_t$.
2. Update the average subgradient $\bar{\mathbf{u}}_t$:
$$\bar{\mathbf{u}}_t \;=\; \frac{t-1}{t}\bar{\mathbf{u}}_{t-1} + \frac{1}{t}\mathbf{u}_t.$$
3. Calculate the next iteration $\mathbf{w}_{t+1}$:
$$\mathbf{w}_{t+1} = \arg\min_{\mathbf{w}} \Upsilon(\mathbf{w}), \tag{3.5}$$
where $\Upsilon(\mathbf{w}) = \left\{ \bar{\mathbf{u}}_t^\top \mathbf{w} + \Omega_\lambda(\mathbf{w}) + \frac{\gamma}{\sqrt{t}} h(\mathbf{w}) \right\}$.

end for

in sequence. At each time, we have to make the decision of $\mathbf{w}_T$ based on the coming data. By defining the objective up to the T-th step as

$$S_T(\mathbf{w}_T) := \frac{1}{T}\sum_{t=1}^{T}(\Omega_\lambda(\mathbf{w}_T) + l_t(\mathbf{w}_T)), \tag{3.6}$$

the objective of online learning for group lasso is to find $\mathbf{w}_T$ in the T-th step such that the objective up to the T-th step, $S_T(\mathbf{w}_T)$, is not much larger than $\min_{\mathbf{w},b} S_T(\mathbf{w})$, the smallest objective of any fixed decision $\mathbf{w}$ from hindsight. Note that in (3.6), we have used $l_t(\cdot)$ to simplify the expression of the loss induced by the t-th coming instance.

The difference between the objective value up to the T-th step and the smallest objective value from hindsight is the *regret* of the online algorithm for group lasso. We can define the *average regret* as

$$\bar{R}_T(\mathbf{w}) := \frac{1}{T}\sum_{t=1}^{T}(\Omega_\lambda(\mathbf{w}_t) + l_t(\mathbf{w}_t)) - S_T(\mathbf{w}). \tag{3.7}$$

Analysis of the regret bound and convergence rate is a key problem to guarantee the online learning algorithms. We will delay the analysis until Section 3.5.

Remark 4. The above proposed online learning for the group lasso is derived from the regularized dual averaging method in [169]. We can also use the FOBOS [52] to solve the online learning for the group lasso. In this case, at each iteration, the FOBOS method is to solve the following minimization problem:

$$\mathbf{w}_{t+1} = \arg\min_{\mathbf{w}}\left\{\frac{1}{2}\|\mathbf{w} - (\mathbf{w}_t - \eta_t\mathbf{u}_t)\|^2 + \eta_t\Omega(\mathbf{w})\right\}, \tag{3.8}$$

where $\Omega(\mathbf{w})$ is defined as (3.2) for the group lasso or as (3.3) for the sparse group lasso. η_t is a constant term which can be set

to $O(1/\sqrt{t})$. It is easy to see the difference between FOBOS for the group lasso and our DA-GL algorithm: The FOBOS method scales the regularization term by a diminishing stepsize η_t while our method keeps it the same.

Remark 5. In the standard group lasso, the features are assumed belonging to one and only one group, i.e., the groups are non-overlapped. If data contain overlapped groups, we can simply replicate the overlapped features as that in [80] to obtain an enlarged dataset, then feed the data into Algorithm 1 to get the solution of the group lasso with overlap. The procedure can be performed similarly as that for the graph lasso.

The key to make Algorithm 1 efficiently solve the group lasso is that the update of the weight in (3.5) should be simple. Here, we first consider the calculation of the bias. In the batch-mode learning for the group lasso, data are given in advance. One can center the data to make the bias vanish. However, for online learning algorithms, the data are not preprocessed. Hence we have to calculate the bias. Here, since the bias is not regularized, it can be calculated by

$$b_{t+1} = \arg\min_b \left\{ \bar{b}_t b + \frac{\gamma}{2\sqrt{t}} b^2 \right\} = -\frac{\sqrt{t}}{\gamma} \bar{b}_t \,. \tag{3.9}$$

Next, let $[v]_+$ denote $\max\{0, v\}$. We can calculate the optimal solution of $\mathbf{w}_{t+1}$ in (3.5) in a closed-form for the following three group lasso models:

Theorem 1. *Given $\bar{\mathbf{u}}_t$ at each iteration, the optimal solution of (3.5) is updated correspondingly as follows:*

*a) **Group lasso**: $\Omega_\lambda(\mathbf{w})$ is defined in (3.2) for some $\lambda > 0$, and $h(\mathbf{w}) = \frac{1}{2}\|\mathbf{w}\|^2$. Then, for $g = 1, \ldots, G$, we have*

$$\mathbf{w}_{t+1}^g = -\frac{\sqrt{t}}{\gamma} \left[1 - \frac{\lambda \sqrt{d_g}}{\|\bar{\mathbf{u}}_t^g\|_2} \right]_+ \cdot \bar{\mathbf{u}}_t^g. \tag{3.10}$$

b) ***Sparse group lasso****:* $\Omega_{\lambda,\mathbf{r}}(\mathbf{w})$ *is defined in (3.3) for some* $\lambda > 0$ *and* $\mathbf{r} \geq \mathbf{0}$*, and* $h(\mathbf{w}) = \frac{1}{2}\|\mathbf{w}\|^2$*. Then we have*

$$\mathbf{w}_{t+1}^g = -\frac{\sqrt{t}}{\gamma}\left[1 - \frac{\lambda\sqrt{d_g}}{\|\mathbf{c}_t^g\|_2}\right]_+ \cdot \mathbf{c}_t^g, \tag{3.11}$$

where the j*-th element of* $\mathbf{c}_t^g$ *is calculating by*

$$c_t^{g,j} = \left[|\bar{u}_t^{g,j}| - \lambda r_g\right]_+ \cdot \text{sign}\ (\bar{u}_t^{g,j}),\ j = 1, \ldots, d_g. \tag{3.12}$$

c) ***Enhanced sparse group lasso****:* $\Omega_{\lambda,\mathbf{r}}(\mathbf{w})$ *is defined in (3.3) for some* $\lambda > 0$ *and* $\mathbf{r} \geq \mathbf{0}$*, and* $h(\mathbf{w}) = \frac{1}{2}\|\mathbf{w}\|^2 + \rho\|\mathbf{w}\|_1$ *with* $\rho \geq 0$ *being a* sparsity-enhancing *parameter. Then*

$$\mathbf{w}_{t+1}^g = -\frac{\sqrt{t}}{\gamma}\left[1 - \frac{\lambda\sqrt{d_g}}{\|\tilde{\mathbf{c}}_t^g\|_2}\right]_+ \cdot \tilde{\mathbf{c}}_t^g, \tag{3.13}$$

where the j*-th element of* $\tilde{\mathbf{c}}_t^g$ *is calculating by*

$$\tilde{c}_t^{g,j} = \left[|\bar{u}_t^{g,j}| - \lambda r_g - \frac{\gamma\rho}{\sqrt{t}}\right]_+ \cdot \text{sign}\ (\bar{u}_t^{g,j}),\ j = 1, \ldots, d_g. \tag{3.14}$$

Remark 6. Equation (3.10) indicates that the solution for the group lasso achieves sparsity in the group level. Equation (3.11) implies that the solution for the sparse group lasso achieves sparsity in both the group level and the individual feature level. Since $\|\mathbf{c}_t^g\|_2 \leq \|\bar{\mathbf{u}}_t^g\|_2$, the solution for the sparse group lasso can achieve more sparsity in the group level than that in the group lasso. Similarly, the solution of the enhanced sparse group lasso achieves more sparsity in both the group level and the individual feature level due to the introduced sparsity-enhancing parameter.

Remark 7. Theorem 1 indicates the simplicity of updating the weight in (3.5). It is noted that the algorithm only needs $\mathcal{O}(d)$ space to store the average subgradient, the weight at each

iteration. In addition, it is possible to adopt the efficient implementation of lazy update in [52] to avoid updating the whole weight at each time for high dimensional data.

3.5 Convergence and Regret Analysis

We have the following theorem providing the bound of the average regret and the weight:

Theorem 2. *Suppose there exists an optimal solution* $\mathbf{w}_T^\star$ *for the problem of (3.1) which satisfies* $h(\mathbf{w}^\star) \leq D^2$ *for some* $D > 0$*, and there exists a constant* L *such that* $\|\bar{\mathbf{u}}_T\|_*^2 \leq L^2$ *for all* $T \geq 1$*. Then we have the following properties for Algorithm 1:*

a) For each $T \geq 1$*, the average regret is bounded by*

$$\bar{R}_T \leq \left(\gamma\sqrt{T}D^2 + \frac{L^2}{2\gamma}\sum_{t=1}^{T}\frac{1}{\sqrt{t}}\right)/T \tag{3.15}$$

b) The sequence of primal variables are bounded by

$$\frac{1}{2}\|\mathbf{w}_{T+1} - \mathbf{w}^\star\|^2 \leq D^2 + \frac{L^2}{\gamma^2} - \frac{\sqrt{T}}{\gamma}\bar{R}_T \tag{3.16}$$

The proof of Theorem 2 can follow the framework developed in [118]. A detailed proof can be found in [169]. The bound in (3.15) can further be simplified as

$$\text{Bound of (3.15)} \leq \frac{\gamma\sqrt{T}D^2 + \frac{L^2}{2\gamma}2\sqrt{T}}{T} = \frac{\gamma D^2 + \frac{L^2}{\gamma}}{\sqrt{T}}.$$

This also indicates that the best γ for the above bound is attained when $\gamma^\star = L/D$, which leads to the *average regret* bound as $\bar{R}_T \leq 2LD/\sqrt{T}$.

Hence, Algorithm 1 can achieve the optimal convergence rate $O(1/\sqrt{T})$. It would be interesting to investigate that by introducing additional assumption, whether the average regret bound can be improve to $O(\log(T)/T)$ as that in [68].

The second result of Theorem 2 gives a bound for the difference between the learned weight and the optimal weight. If $\bar{R}_T > 0$, then the bound can be tighter. However, the term of $\bar{R}_T$ in the bound cannot be simply discarded since the average regret $\bar{R}_T$ may be negative although this is unlikely in practical situation.

3.6 Experiments

In the following, we present experimental results to demonstrate the advantages of the online learning algorithms for the group lasso models on both synthetic and real-world datasets.

We compare the following five algorithms: the batch-mode learning algorithms for the lasso and the group lasso (GL); the online learning algorithm by the dual averaging method on the L_1 regularization (L_1-RDA) in [169]; the online learning algorithm by the dual averaging method for the group lasso (DA-GL) in (3.10) and for the sparse group lasso (DA-SGL) in (3.11). We use the implementation of the R-package, `grplasso` [112] for the batch-mode learning algorithms. The online learning algorithms are implemented in Matlab. All algorithms run on a PC with 2.13 GHz dual-core CPU.

3.6.1 Synthetic data

We test the algorithms on various synthetic data similar to those generated in [59, 112, 184], including data with sparsity in the group level and the individual feature level. Our proposed online learning algorithms for the group lasso models consistently reveal merits. Here, we only report the results on the data with sparsity both in group level and individual feature level. The goal of this experiment is to test the efficiency and effectiveness of the proposed algorithms.

Before generating the data, we first generate a true model. A weight vector is in 100 dimensions consisting of ten blocks of ten, i.e., $\mathbf{w} \in \mathbb{R}^{100}$, and $d_g = 10$, for $g = 1, \ldots, 10$. The numbers of non-zero weights in the first six blocks of 10 are 10, 8, 6, 4, 2, and 1, respectively, with $w_i = \pm 1$, the sign chosen at random. The weights for the rest forty features are all zero. The bias is set to 0.

We then generate N_{tr} data points by letting $\mathbf{x}_i = L\mathbf{v}_i$, $i = 1, \ldots, N_{tr}$, where $\mathbf{v}_i \sim \mathcal{N}(0, \mathbf{I}_d)$ and L is the Cholesky decomposition of the correlation matrix, Σ. The (i, j)-th entry in the g-th group of the correlation matrix is $\Sigma^g_{i,j} = 0.2^{|i-j|}$ and zero for entries within different groups. The target value is set by $y_i = \text{sign}\,(\mathbf{w}^\top \mathbf{x}_i + \epsilon)$, where ϵ is a Gaussian noise with standard deviation 4.0. We randomly generate data with the size in $\{25, 50, 100, 500, 1000, 5000, 10^4, 10^5\}$. The model is evaluated on an additional test set of size N_{tr}. We repeat the experiments 50 times and average the results.

For the group lasso models, the regularization parameter λ is tested from $\lambda_{\max} * \{0.5, 0.2, 0.1, 0.05\}$, where $\lambda_{\max}$ is the maximum λ that makes the weight in the group lasso vanish. For the online learning algorithms, since we know the true model, we can obtain the corresponding L and D as defined in Theorem 2 and set $\gamma = L/D$. For the DA-SGL model, r_g is set to 1 for all groups.

Table 3.1 reports the average results on the synthetic data in terms of accuracy and the average F1 score on the true weight. The average F1 score is to verify whether the learned weight has the same sign of the true model. We calculate the F1 scores of the weight on the tasks of $+1$ vs. $\{-1, 0\}$, -1 vs. $\{+1, 0\}$, and 0 vs. $\{-1, +1\}$ and average these three F1 scores. The larger the F1 score, the more accurate it is in predicting the sign of the weight.

Several observations can be drawn from the results. First,

Table 3.1: Evaluation on the synthetic dataset when varying the number of training data. The best results are highlighted (achieved by paired t-test with 95% confidence level).

N_{tr}	Accuracy (%)					Average F1 (%)				
	Lasso	GL	L_1-RDA	DA-GL	DA-SGL	Lasso	GL	L_1-RDA	DA-GL	DA-SGL
25	54.2 ± 14.1	54.2± 11.4	56.6± 9.9	57.0± 11.6	**57.6**± 11.0	23.6± 8.5	37.3± 13.6	35.6± 6.3	37.2± 3.0	**37.9**± 4.5
50	58.2 ± 7.7	60.0± 6.3	59.5± 6.9	60.9± 6.2	**60.9**± 6.0	35.0± 9.3	**49.8**± 6.0	39.7± 6.5	49.7± 3.0	**49.8**± 4.9
100	62.7 ± 5.5	64.0± 5.1	61.7± 4.8	64.5± 4.1	**64.6**± 4.5	47.0± 7.2	**57.4**± 2.4	46.5± 9.7	57.1± 2.7	**57.4**± 5.9
500	75.6 ± 2.4	75.7± 2.3	66.2± 3.0	74.8± 2.3	**75.9**± 2.2	65.0± 2.5	65.5± 2.1	63.6± 9.7	65.2± 6.8	**81.9**± 5.3
1000	77.7 ± 1.5	77.8± 1.5	65.9± 2.0	76.3± 1.4	**77.9**± 1.6	70.1± 2.4	67.2± 2.1	64.9± 8.7	67.2± 4.7	**87.3**± 4.3
5000	**79.4** ± 0.4	**79.4**± 0.3	67.8± 1.5	78.2± 0.6	**79.4**± 0.8	88.2± 2.4	68.2± 2.0	66.8± 8.0	68.3± 2.9	**93.7**± 2.5
10^4	**80.0** ± 0.2	**80.0**± 0.1	68.0± 1.3	79.8± 0.3	**80.0**± 0.1	94.1± 2.3	69.1± 1.8	67.4± 5.5	68.4± 2.5	**94.2**± 2.1
10^5	**80.1** ± 0.1	**80.1**± 0.1	69.7± 1.2	79.9± 0.1	**80.1**± 0.1	**97.3**± 2.2	69.5± 1.7	68.1± 5.1	68.7± 2.3	**97.3**± 2.1

the accuracy values of all algorithms increase with the number of training instrances. Among them, the DA-SGL achieves the best accuracy, especially when the number is small. The DA-GL achieves slightly worse results than the DA-SGL and slightly worse results than the GL when the number is large. The two batch-mode algorithms achieve nearly the same accuracy when the number of training instances is large. Second, results about the average F1 score clearly show that the DA-SGL outperforms all the other four algorithms. With respect to F1-scores, the DA-SGL behaves similarly as the GL when the number of training instances is small and as the lasso when the number is large. The DA-SGL combines both the advantages of the lasso and the GL and is more accurate in predicting the sign of the weight. The average F1 scores on the GL and on the DA-GL are similar. Both models cannot achieve sparsity in the individual feature level and therefore, the scores are lower than those of the DA-SGL.

To see the efficiency of the online learning algorithms, we show the running time in Figure 3.1. Since the online learning algorithms and the batch-mode algorithms run in different programming platforms, the time comparison is not fair and a little bias to the R-package. However, the time cost by the online learning algorithms is clearly less than that cost by the batch-mode algorithms. These three online algorithms cost nearly the same time and the L_1-RDA costs less since the DA-GL and the DA-SGL need some calculation within each group. The batch-train algorithms cost much time in loading the data into memory when the size of the data is large.

3.6.2 Splice Site Detection

In order to evaluate performance of the online learning algorithms on real-world applications, we apply our algorithms in

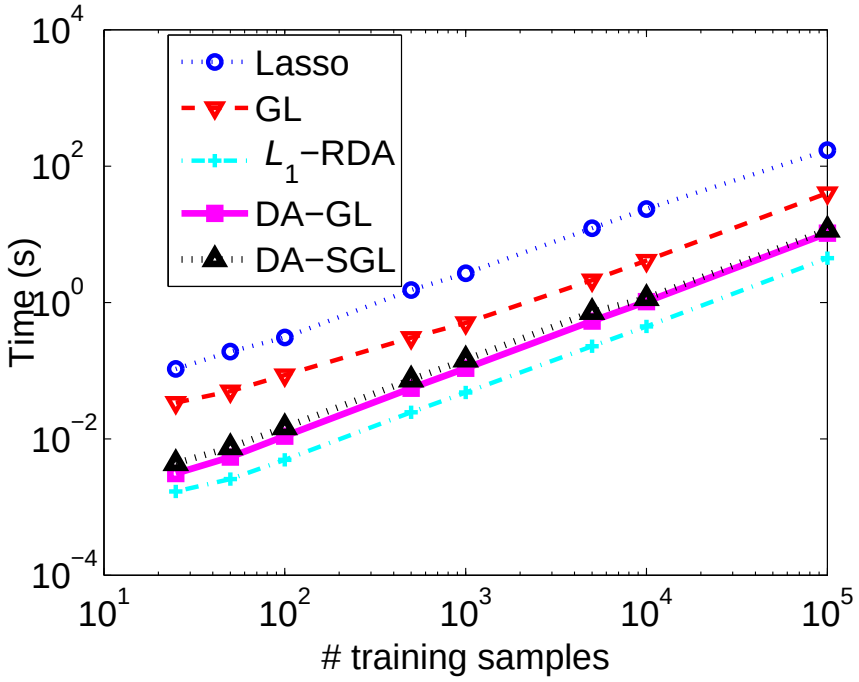


Figure 3.1: Log-log plot of computation time on training the synthetic dataset. The batch-model algorithms suffer from much time cost in loading large-scale datasets.

the task of splice site detection, which plays an important role in gene finding. Splice sites are the regions between coding (exons) and non-coding (introns) DNA segments. The 5' splice site (5'ss) end of an intron is called a donor splice site and the 3'ss ends an acceptor splice site.

We adopt the MEMset Donar dataset for the evaluation, which is available from `http://genes.mit.edu/burgelab/maxent/ssdata/`. This dataset is widely used to demonstrate the advantages of the group lasso models [112, 134]. It contains a training set of 8,415 true and 179,438 false human donor sites. An additional test set consists of 4,208 true and 89,717 false donor site. A sequence of a real splice site is modeled within a window over positions $[-3, 5]$ that consists of the last three bases of the exon and the first six bases of the intron. False splice sites

are sequences on the DNA which match the consensus sequence at positions 0 and 1. Removing the consensus "GT" results in a sequence length of 7 with 4 level $\{A, C, G, T\}$; see [183] for detailed description.

We follow the experimental setup in [112] and measure the performance by the maximum correlation coefficient [183]. The original training dataset is used to construct a balanced training dataset with 5,610 true and 5,610 false donar sites and an unbalanced validation set with 2,805 true and 59,804 false donor sites, which exhibits the same true/false ratio as the test set. All sites are chosen randomly without replacement such that the two sets are disjoint. The test set remains unchanged to evaluate the performance. The group lasso on the data with up to 2nd order interactions and up to 4^{th} order interactions has been analyzed in [112] and [134], respectively. As reported in [134], there is no much improvement using higher order interaction. Hence we construct a model consisting of all three-way and lower order interactions, which involves 64 terms or $d = 2604$-dimensional feature space.

In the algorithms, the parameter λ is varied from $[0.01, 10]$ to produce different levels of sparsity. The parameter γ for the online learning algorithms is tuned on the validation set. The element level sparsity parameter of the DA-SGL is set to $\sqrt{d_g}$ for simplicity. Table 3.2 shows the results of the online learning algorithms in terms of correlation coefficient vs. sparsity. We can see that the online learning algorithms attain satisfactory results and they are competitive with the results in [112, 183]. It is noted that the DA-SGL can achieve better performance in all given levels of structural sparsity. In terms of computation time, the online learning algorithms cost about 10^3 seconds for each epoch and the DA-GL costs the lest time, while the batch-train group lasso algorithm costs about quadruple of the online learning algorithms.

Table 3.2: Maximum correlation coefficients vs. sparsity on the MEMset Donar dataset.

% Non-zero	L1-RDA	DA-GL	DA-SGL
10	0.5632	**0.5656**	**0.5656**
40	0.6056	0.6071	**0.6082**
60	0.6481	0.6496	**0.6501**
80	0.6494	**0.6520**	**0.6520**

3.7 Summary

In this work, we propose a novel online learning algorithm framework for the group lasso. We apply this framework for different group lasso extensions, including the sparse group lasso and our proposed enhanced sparse group lasso. We provide closed-form solutions for all the group lasso models and give the convergence rate of the average regret. We also conduct empirical evaluation on the proposed algorithms in comparison to a recently proposed online learning algorithm for L_1-regularization minimization and the batch-mode learning algorithms for the lasso and the group lasso. The results clearly demonstrate the advantages of the proposed algorithms in both efficiency and effectiveness.

There are still some remaining work: 1) to further evaluate on the FOBOS method for the group lasso in (3.8); 2) to further study the lazy update scheme in the FOBOS method for handling high-dimensional data; 3) to derive a faster convergence rate for the online learning algorithm by including additional assumptions, e.g., the strongly convexity assumption; and 4) to extend the online learning algorithm to solve other problems in the group lasso style.

□ **End of chapter.**

Chapter 4

Online Learning for Multi-Task Feature Selection

4.1 Introduction

Learning multiple related tasks simultaneously by exploiting shared information across tasks has demonstrated advantages over those models learned within individual tasks [3, 4, 26, 55, 63, 126, 179, 189]. The multiple tasks learning framework has been successfully applied in various applications [10, 26, 31, 55, 56, 121, 173].

A key problem of multi-task learning is to find the explanatory features across these multiple related tasks [5, 121, 194]. Many methods have been proposed to solve this problem by utilizing a variant of the L_1-norm on the regularization, or more specifically, by imposing the mixed $L_{p,1}$-norm (p is usually set to 2 or ∞) on the regularization [5, 104, 121, 127]. The main idea of these methods is that they not only can gain benefit from the L_1-norm regularization which is theoretically proven to yield sparse solutions while maintaining good perfromance [155], but also can achieve grouped sparsity through the L_p-norm.

Although previous multi-task feature selection (MTFS) methods succeed in several aspects, they still contain some drawbacks. First, these methods are conducted in batch-mode train-

ing. The training procedure cannot be started until the data are prepared. A fatal drawback is that these methods cannot solve the applications when the training data is obtained sequentially. For example, in prompt new classification, if one needs a timely classifier, previous batch-mode trained algorithms cannot fulfill this objective. The second drawback is that these algorithms suffer from inefficiency when the size of training dataset is huge, especially when the data cannot be loaded into memory simultaneously. In this case, one may have to conduct additional procedure, e.g., subsampling, to choose the data for training. This may degrade the performance of the model since the available data are not sufficiently utilized. The third drawback is that most previous MTFS methods can only select features in individual tasks or across all tasks, but cannot find important tasks further from the important features. This reduces the ability of the MTFS methods in interpreting the physical meaning of the learned model. Although there is an MTFS method based on the Multiple Inclusion Criterion [47] to seek both important features and important tasks, its formulation needs to solve a non-convex problem by minimizing the $L_{0,0}$-norm regularization on the weights and it suffers from the local optimal solution.

To tackle the above problems, we first develop a novel MTFS model, named multi-task feature and task selection (MTFTS), which selects important features across all tasks and important tasks that dominate the selected features. Furthermore, we propose a novel online learning framework to convert the batch-mode trained MTFS models into their online ones. The key contributions of this article are highlighted as follows:

- The new proposed MTFTS model can select important features across all tasks and further reveal the important tasks that dominate the important features. This strengthens the interpretation ability of the MTFS models. As indicated in our experiments, the MTFTS can achieve more sparse so-

lutions even without sacrificing the performance.

- A novel online learning framework is proposed to solve the Multi-task feature selection problem. By converting the batch-mode trained models into their online learning versions, we can update the models sequentially with new coming data and achieve high efficiency.
- Three batch-mode trained MTFS models are converted into their online modes under our proposed online learning framework. More importantly, they can achieve high efficiency in both time complexity and memory cost. At each iteration, the algorithms only need $\mathcal{O}(d \times Q)$ memory space to store the required information and $\mathcal{O}(d \times Q)$ time to update the learning weight, where d is the number of features and Q is the number of tasks. Hence, the online algorithms can solve the MTFS problem in large-scale datasets.
- We have provided the convergence rate of the online learning algorithm. The result theoretically guarantees the convergence of the proposed online learning algorithms.
- We have conducted detailed experiments on three real-world datasets to demonstrate the merits of our proposed model and the online learning algorithm. Empirical results show the efficiency and effectiveness of the proposed online learning algorithms.

The rest of the chapter is organized as follows. In Section 4.2, we review the existing batch-mode training multi-task feature selection methods in the literature. In Section 4.3, we define the problem setup and introduce the multi-task feature selection formulation. In Section 4.4, we depict the proposed MTFTS method and in Section 4.5, we present the online learning framework for solving the MTFS models. The convergence rate of the average regret bound is provided in Section 4.6. In Section 4.7,

we report the experimental comparison and results. Finally, we conclude the chapter with future work in Section 4.8.

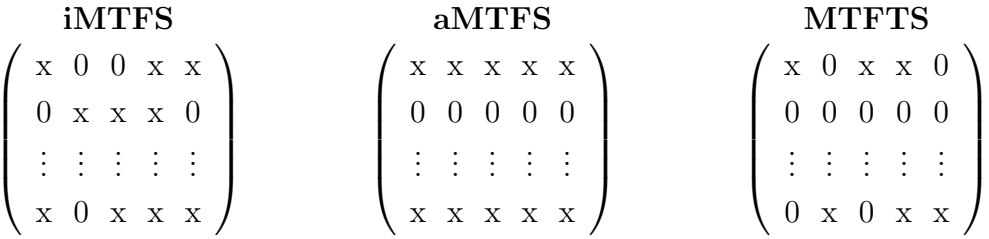


Figure 4.1: Illustration of the weight matrices learned by three multi-task feature selection methods. Here is an example with five tasks. The iMTFS method selects important features individually. The aMTFS method selects all relevant features and discards the second irrelevant feature for all the tasks. The MTFTS method not only discards the second irrelevant feature, but also finds out the important tasks for the important features. As in this example, the weights learned from the MTFTS method indicate that the first feature is further dominated by the first to fourth tasks, while the last feature is dominated by the second, fourth, and fifth tasks. This provides more specific information in interpreting the learned model.

4.2 Related Work

Multi-task feature selection is an important topic in machine learning [4, 31, 107, 82] as well as a very useful tool for data mining applications [5, 30, 47]. In the following, we review some existing methods to solve this problem.

Methods based on variants of the L_1-norm, particularly matrix norms such as the $L_{2,1}$-norm and $L_{\infty,1}$-norm have been widely used in multi-task feature selection [5, 121, 107, 127]. In [5], a generalization of the single-task L_1-norm regularization is formulated to learn a few common features across multiple tasks. Although the formulated model is a non-convex problem, it is solved by an iterative alternating algorithm, where at each

step a convex optimization problem is solved. This method can be easily extended to learn the sparse nonlinear representation using kernels, which leads to solving an optimization problem involving the trace norm. The time cost of this method is usually high since it needs to perform the Singular Value Decomposition (SVD) on the kernel matrices at each step. In [121], a blockwise path-following algorithm is proposed to solve the MTFS models based on the mixed norms of joint regularization of hybridizing L_1-norm with L_2-norm, or L_∞-norm. A theoretical result in [121] shows that the proposed algorithm with random projection can obtain non-linear solution which approximates the solution obtained from the trace norm minimization in [5]. The Nesterov's method, an optimal first-order black-box method for smooth convex optimization, has been adopted in [107] to solve the MTFS problem with the $L_{2,1}$-norm regularization on the weight matrices. An efficient Euclidean projection is proposed to solve the non-smooth problem in the $L_{2,1}$-norm regularizer. This method scales well, linearly on the number of training samples, the sample dimensionality and the number of tasks.

In summary, various batch-mode trained algorithms, but no online learning algorithm, have been proposed to solve the MTFS problem in the literature. For the batch-mode trained MTFS methods, they are unsuitable for those applications where the data appear in sequence and for the case that the data are so large that they cannot be loaded into memory simultaneously. The above problems motivate us to propose the online learning algorithm in this article.

4.3 Multi-Task Feature Selection

In this section, we first introduce the notation and the problem setup for multi-task learning. Following that, we present the formulation for multi-task feature selection in the literature.

4.3.1 Problem Setup

Suppose there are Q tasks with all data coming from the same space $\mathcal{X} \times \mathcal{Y}$, where $\mathcal{X} \subset \mathbb{R}^d$ and $\mathcal{Y} \subset \mathbb{R}$. For each task, there are N_q data points. Hence, it consists of a dataset of $\mathcal{D} = \bigcup_{q=1}^{Q} \mathcal{D}_q$, where $\mathcal{D}_q = \{\mathbf{z}_i^q = (\mathbf{x}_i^q, y_i^q)\}_{i=1}^{N_q}$ are sampled from a distribution $\mathcal{P}_q$ on $\mathcal{X} \times \mathcal{Y}$. Here, $\mathcal{P}_q$ is usually assumed different for each task but all $\mathcal{P}_q$'s are related, e.g., as discussed in [14]. The goal of multi-task learning (MTL) is to learn Q functions $f_q : \mathbb{R}^d \to \mathbb{R}$, $q = 1, \ldots, Q$ such that $f_q(\mathbf{x}_i^q)$ approximates y_i^q. When $T = 1$, it is the standard (single task) learning problem.

4.3.2 Formulation

Typically, in multi-task learning models, the decision function f_q for the q-th task is assumed as a hyperplane parameterized by the model weight vector $\mathbf{w}^q$ [5, 121], i.e.,

$$f_q(\mathbf{x}) = \mathbf{w}^{q\top}\mathbf{x}, \quad q = 1, \ldots, Q. \tag{4.1}$$

Hence, the total learned weights form a matrix in the size of $d \times Q$. To make the notation uncluttered, in the following, we express the learned weight matrix into column-wise and row-wise vectors as follows:

$$\mathbf{W} = \left(\mathbf{w}^1, \mathbf{w}^2, \ldots, \mathbf{w}^Q\right) = (\mathbf{W}_{\bullet 1}, \ldots, \mathbf{W}_{\bullet Q}) = \left(\mathbf{W}_{1\bullet}^\top, \ldots, \mathbf{W}_{d\bullet}^\top\right)^\top. \tag{4.2}$$

The objective of multi-task feature selection models is to learn the weight matrix $\mathbf{W}$ by minimizing an empirical risk and the regularization on the weights:

$$\min_{\mathbf{W}} \quad \sum_{q=1}^{Q} \frac{1}{N_q} \sum_{i=1}^{N_q} l^q(\mathbf{W}_{\bullet q}, \mathbf{z}_i^q) + \Omega_\lambda(\mathbf{W}), \tag{4.3}$$

where $\lambda \geq 0$ is a constant to balance the loss and the regularization term. $l^q(\mathbf{W}_{\bullet q}, \mathbf{z}_i^q)$ defines the loss on the sample $\mathbf{z}_i^q$ for

the q-th task. Various loss functions can be adopted and they are usually assumed convex. Some typical loss functions are

- **Squared loss:** $l(\mathbf{w},\ \mathbf{z}) = \frac{1}{2}(y - \mathbf{w}^\top \mathbf{x})^2$. This loss is a standard loss used in regression problems [107].
- **Logit loss:** $l(\mathbf{w},\ \mathbf{z}) = \log(1 + \exp(-y(\mathbf{w}^\top \mathbf{x})))$. This loss is usually used in binary classification problems [106].
- **Hinge loss:** $l(\mathbf{w},\ \mathbf{z}) = [1 - y\mathbf{w}^\top \mathbf{x}]_+$. This loss is usually used in solving binary classification problems by support vector machines [160].

Note that in the above loss function definitions, we use $\mathbf{w}$ to denote the weight of a task for simplicity.

In (4.3), $\Omega_\lambda(\mathbf{W})$ defines the regularization on the weights of tasks. Various mixed norms on the regularization have been proposed in the literature to impose sparse solutions so as to select the important features. They include

- **$L_{1,1}$-norm Regularization:** This model simply sums the L_1-regularizations on the weights of all tasks together to yield sparse solutions [121]. We name it as the individually learned Multi-Task Feature Selection (**iMTFS**) method.

$$\Omega_\lambda(\mathbf{W}) = \lambda \sum_{q=1}^{Q} \|\mathbf{W}_{\bullet q}\|_1 = \lambda \sum_{j=1}^{d} \left\|\mathbf{W}_{j\bullet}^\top\right\|_1 . \tag{4.4}$$

  The above formulation is equivalent to solving a lasso problem for each task when the squared loss is used [121]. The reason that the L_1-norm regularization can yield sparse solutions is that it usually attains the optimal solutions at the corner [155]. Hence, imposing the $L_{1,1}$-regularization in the formulation can yield sparse solutions for each individual task, but it does not find the information across the tasks.
- **$L_{2,1}$-norm Regularization:** It is to penalize the L_1-norm on the L_2-norms of the weight vectors across tasks [107, 121]. It is to conduct feature selection across multiple tasks.

We name it as the aMTFS method.

$$\Omega_\lambda(\mathbf{W}) = \lambda \sum_{j=1}^{d} \left\| \mathbf{W}_{j\bullet}^\top \right\|_2 . \tag{4.5}$$

It is easy to show that the $L_{2,1}$-norm regularization reduces to L_1-norm regularization if there is only one task. When there are multiple tasks, the weights corresponding to the j-th feature are grouped together via the L_2-norm of $\mathbf{W}_{j\bullet}^\top$. Hence, the $L_{2,1}$-norm regularization tends to select features based on the strength of the input variables of the Q tasks jointly, rather than on the strength of individual input variables as in the case of single task learning [5, 121].

- **$L_{p,1}$-norm Regularization:** It can be easily extended to include other joint regularization on the weights, e.g., the $L_{p,1}$-norm for $1 \leq p \leq \infty$ to yield sparse solutions [127]. The choice of p usually depends on how much feature sharing that one assume between tasks, from none ($p = 1$) to full sharing ($p = \infty$). Increasing p corresponds to allowing better "group discounts" for sharing the same feature, from $p = 1$ where the cost grows linearly with the number of tasks that use a feature, to $p = \infty$ where only the most demanding task matters [121].

Remark 8. Although the above introduced MTFS models find the decision functions in linear forms, it is noted that by projecting the data points on a random direction in the Reproducing Kernel Hilbert Space (RKHS), one can attain non-linear solutions on the original space [121].

4.4 Multi-Task on both Feature and Task Selection

The MTFS method imposed by the $L_{2,1}$-norm regularization can select features across all tasks, however, it yields non-sparse so-

lutions for that selected features [121]. This is because sparse solutions cannot be obtained when p is greater than one [200]. Hence, the aMTFS method cannot further find out the important tasks for the selected features. This degrades the model ability in interpreting the data. In order to find out important explanatory features across all tasks and to find out the important tasks on the selected features simultaneously, we propose the multi-task feature selection method on both feature and task selection (MTFTS) by introducing a new $L_{1/2,1}$-norm regularization as follows:

$$\Omega_{\lambda,\mathbf{r}} = \lambda \sum_{j=1}^{d} \left(r_j \left\| \mathbf{W}_{j\bullet}^{\top} \right\|_1 + \left\| \mathbf{W}_{j\bullet}^{\top} \right\|_2 \right), \tag{4.6}$$

where $r_j \geq 0$, for $j = 1, \ldots, d$, is a constant controlling the sparsity of the solutions on task level: a larger r_j value will introduce more sparsity on the task selection. Usually, we can select a very small r_j value to yield additional sparse solution while keeping the model performance [180]. Note that this model is similar to the sparse group lasso introduced in [59, 181].

4.5 Online Learning for Multi-Task Feature Selection

To tackle the insufficiency of batch-mode trained algorithms and motivated by the recent success of online learning algorithms for solving the L_1-regularization problem [12, 52, 97], we propose an online learning framework to solve the multi-task feature selection problem, namely **DA-MTFS**, in the following. This framework has been successfully developed for minimizing the L_1-regularization [169] and group lasso [181], which is based on recently developed first-order method for optimizing convex composite functions [118]. The algorithm is outlined in Algorithm 2.

Algorithm 2 Online learning framework for multi-task feature selection

Input:

- $\mathbf{W}_0 \in \mathbb{R}^{d\times Q}$, and a strongly convex function $h(\mathbf{W})$ with modulus 1 such that
$$\mathbf{W}_0 = \arg\min_{\mathbf{W}} h(\mathbf{W}) \in \arg\min_{\mathbf{W}} \Omega(\mathbf{W}) \,. \tag{4.7}$$
- Given a const $\lambda > 0$ for the regularizer, $\gamma > 0$ for the function $h(\mathbf{W})$

Initialization: $\mathbf{W}_1 = \mathbf{W}_0$, $\bar{\mathbf{G}}_0 = \mathbf{0}$.

for $t = 1, 2, 3, \ldots$ **do**

1) Given the function l_t, compute the subgradient on $\mathbf{W}_t$, $\mathbf{G}_t \in \partial\, l_t$ for the coming Q instances with each for one task, $(\mathbf{z}_t^1, \ldots, \mathbf{z}_t^Q)$
2) Update the average subgradient $\bar{\mathbf{G}}_t$:
$$\bar{\mathbf{G}}_t \;=\; \frac{t-1}{t}\bar{\mathbf{G}}_{t-1} + \frac{1}{t}\mathbf{G}_t$$
3) Calculate the next iteration $\mathbf{W}_{t+1}$ based on $\bar{\mathbf{G}}_t$:
$$\mathbf{W}_{t+1} = \arg\min_{\mathbf{W}} \Upsilon(\mathbf{W}) = \left\{ \bar{\mathbf{G}}_t^\top \mathbf{W} + \Omega(\mathbf{W}) + \frac{\gamma}{\sqrt{t}} h(\mathbf{W}) \right\}. \tag{4.8}$$

end for

For Algorithm 2, we have the following remarks:

Remark 9. In Algorithm 2, it should be noted that instances for every task come at each iteration. This follows the same online learning scheme for multi-task learning in [46]. In real-world applications, if we cannot get one instance for all tasks at one iteration, we an simply set the instance for that task to zero so as not update the weights for that task. However, intuitively, this will make the learned weight matrices unbalance bias to those tasks with training instances.

Remark 10. Algorithm 2 needs to calculate the subgradients of the loss functions on the weights at each iteration. For typical loss functions, we can calculate them by

$$l'(\mathbf{w}, \mathbf{z}) = \begin{cases} (\mathbf{w}^\top \mathbf{x} - y)\mathbf{x} & \text{square loss} \\ \frac{-y\mathbf{x}}{1+\exp(y(\mathbf{w}^\top \mathbf{x}))} & \text{logit loss} \\ [-y\mathbf{x}]_+ & \text{hinge loss} \end{cases}, \tag{4.9}$$

where $\mathbf{w}$ means the weight of a task and $\mathbf{x}$ corresponds to a feature vector in that task for simplicity.

Remark 11. The above proposed online learning framework for the MTFS models is motivated from the regularized dual averaging method for lasso [169] and group lasso [181]. We can also consider the FOBOS [52] to solve the online learning for the MTFS models. In this case, at each iteration, the FOBOS method is to solve the following minimization problem:

$$\mathbf{W}_{t+1} = \arg\min_{\mathbf{W}} \left\{ \frac{1}{2} \|\mathbf{W} - (\mathbf{W}_t - \eta_t \mathbf{G}_t)\|_F^2 + \eta_t \Omega(\mathbf{W}) \right\}, \tag{4.10}$$

where $\Omega(\mathbf{W})$ is defined as Eq. (4.4) for the iMTFS, Eq. (4.5) for the aMTFS, or Eq. (4.6) for the MTFTS. η_t is a constant term which can be set to $\mathcal{O}(1/\sqrt{t})$. The difference between the FOBOS for the MTFS and our DA-MTFS algorithm is clear: The

FOBOS method scales the regularization term by a diminishing stepsize η_t, while our method keeps it the same, a more balance setting for online algorithms.

In Algorithm 2, the key to solve the online MTFS algorithms efficiently depends on the simplicity of updating the weight in (4.8). Here, we have the following theorem to efficiently update the weight matrix $\mathbf{W}_{t+1}$ in (4.8) by closed-form solutions:

Theorem 3. *Given $h(\mathbf{W}) = \frac{1}{2}\|\mathbf{W}\|_F^2$, and the average subgradient $\bar{\mathbf{G}}_t$ at each iteration, the optimal solution of the corresponding MTFS models can be updated by*

(a) ***iMTFS****: For $i = 1, \ldots, d$ and $q = 1, \ldots, Q$,*

$$(W_{i,q})_{t+1} = -\frac{\sqrt{t}}{\gamma}\left[|(\bar{G}_{i,q})_t| - \lambda\right]_+ \cdot \text{sign}\ ((\bar{G}_{i,q})_t). \tag{4.11}$$

(b) ***aMTFS****: For $j = 1, \ldots, d$,*

$$(\mathbf{W}_{j\bullet})_{t+1} = -\frac{\sqrt{t}}{\gamma}\left[1 - \frac{\lambda}{\|(\bar{\mathbf{G}}_{j\bullet})_t\|_2}\right]_+ \cdot (\bar{\mathbf{G}}_{j\bullet})_t. \tag{4.12}$$

(c) ***MTFTS****: For $j = 1, \ldots, d$,*

$$(\mathbf{W}_{j\bullet})_{t+1} = -\frac{\sqrt{t}}{\gamma}\left[1 - \frac{\lambda}{\|(\bar{\mathbf{U}}_{j\bullet})_t\|_2}\right]_+ \cdot (\bar{\mathbf{U}}_{j\bullet})_t, \tag{4.13}$$

where the q-th element of $(\bar{\mathbf{U}}_{j\bullet})_t$ is calculated by

$$(\bar{U}_{j,q})_t = \left[|(\bar{G}_{j,q})_t| - \lambda r_j\right]_+ \cdot \text{sign}\ ((\bar{G}_{j,q})_t),\ \ q = 1, \ldots, Q. \tag{4.14}$$

The proof of Theorem 3 is provided in the Appendix. We first give some remarks about the results.

Remark 12. Equation (4.11) implies that the online learning algorithm for the iMTFS can yield sparse solutions in the element level, but it does not utilize any information across tasks.

Equation (4.12) indicates that the online learning algorithm for the aMTFS can select those important features in a grouped manner and it will discard irrelevant features for all tasks. Equation (4.13) implies that the online learning algorithm for the MTFTS can select important features and important tasks dominated the selected features. Since $\|(\bar{\mathbf{U}}_{j\bullet})_t\|_2 \leq \|(\bar{\mathbf{G}}_{j\bullet})_t\|_2$, the MTFTS tends to select fewer features than the aMTFS under the same regularization parameter.

Remark 13. Theorem 3 indicates simplicity in updating the weight in (4.8). It is noted that the algorithm only needs $\mathcal{O}(d \times Q)$ space to store the required information, the average subgradient and the weight at each iteration. In addition, the worst case time complexity for updating the weights is also $\mathcal{O}(d \times Q)$. However, it is possible to adopt the lazy update scheme in [52, 97] to update the weight only with the non-zero elements. This is especially useful when the data is sparse but with very large dimension.

4.6 Convergence and Regret Analysis

A main issue to guarantee the online learning algorithm is to analyze its regret bound and the convergence rate. The *regret* is defined as the difference of the objective value up to the T-th step and the smallest objective value from hindsight. Here, we use an *average regret* which is defined by

$$\bar{R}_T(\mathbf{W}) := \frac{1}{Q} \sum_{q=1}^{Q} \left(\frac{1}{T} \sum_{t=1}^{T} \left(\Omega(\mathbf{W}_{\bullet q_t}) + l_t(\mathbf{W}_t) \right) - S_T(\mathbf{W}_{\bullet q}) \right), \tag{4.15}$$

where $S_T(\mathbf{W}_T)$ defines the objective up to the T-th step, $S_T(\mathbf{W}_T)$ as follows:

$$\begin{aligned} S_T(\mathbf{W}_T) &:= \Omega_\lambda(\mathbf{W}_T) + \sum_{q=1}^{Q} \frac{1}{T} \sum_{t=1}^{T} l(\mathbf{W}_T,\ \mathbf{z}_t^q), \\ &= \frac{1}{T} \sum_{t=1}^{T} (\Omega_\lambda(\mathbf{W}_T) + l_t(\mathbf{W}_T)). \end{aligned} \tag{4.16}$$

In the above, we use $l_t(\cdot)$ to simplify the expression of the loss induced by the t-th coming instances for all tasks.

The following theorem provides the bound of the average regret:

Theorem 4. *Suppose there exists an optimal solution $\mathbf{W}^\star$ for the problem of (4.3) which satisfies $h(\mathbf{W}^\star) \leq D^2$ for some $D > 0$, and there exists a constant L such that $\left\| \left(\bar{\mathbf{G}}_{\bullet_q} \right)_T \right\|_* \leq L$ for all $T \geq 1$ and $q = 1, \ldots, Q$. Then we have the following properties for Algorithm 2: for each $T \geq 1$, the average regret is bounded by*

$$\bar{R}_T \leq \left(\gamma \sqrt{T} D^2 + \frac{L^2}{2\gamma} \sum_{t=1}^{T} \frac{1}{\sqrt{t}} \right) / T. \tag{4.17}$$

The proof of Theorem 4 can follow the scheme developed in [118, 169] for each task and (4.17) is summed up for all the tasks. The above theorem indicates that Algorithm 2 can achieve the optimal convergence rate $O(1/\sqrt{T})$. In addition, the bound in (4.17) can further be simplified as

$$\frac{\gamma \sqrt{T} D^2 + \frac{L^2}{2\gamma} \sum_{t=1}^{T} \frac{1}{\sqrt{t}}}{T} \leq \frac{\gamma \sqrt{T} D^2 + \frac{L^2}{2\gamma} 2\sqrt{T}}{T} \leq \left(\gamma D^2 + \frac{L^2}{\gamma} \right) / \sqrt{T}.$$

This indicates that the best γ for the above bound is attained when $\gamma^\star = L/D$ and this leads to the *average regret* bound as $\bar{R}_T \leq 2LD/\sqrt{T}$. For practical problems, the best γ is usually tuned by cross validation.

4.7 Empirical Analysis

In the following, we conduct detailed experiments to demonstrate the characteristics and merits of the online learning algorithms on the MTFS problem. Five algorithms are compared: the batch-mode learning algorithms for the iMTFS and the aMTFS; the online learning algorithms [1] by the dual averaging method for the iMTFS (DA-iMTFS) updated by Eq. (4.11), for the aMTFS (DA-aMTFS) updated by Eq. (4.12), and for the MTFTS (DA-MTFTS) updated by Eq. (4.13), respectively. All algorithms are run in Matlab on a PC with 2.13 GHz dual-core CPU.

The experiments try to answer the following questions: (1) What is the performance of the compared algorithms on the realworld datasets? (2) What is the trade-off between the performance and the sparsity in the online algorithms? (3) What is the effect of the algorithm parameter γ with respect to the regularization parameter λ? (4) What are the most important features learned and how are the learned weight matrices? (5) How is the efficiency of the algorithms?

4.7.1 School Data

We first test the algorithms on a benchmark dataset, the school dataset [2]. This dataset has been previously evaluated on the batch-mode trained multi-task learning [11, 56] and multi-task feature learning [5, 55, 107]. This dataset consists of the exam scores of 15,362 students from 139 secondary schools in London during the years 1985, 1986, and 1987. The goal is to predict the exam scores of the students based on the following features: year of the exam (YR), 4 school-specific and 3 student-specific

[1] Our source codes are available in `http://appsrv.cse.cuhk.edu.hk/~hqyang/doku.php?id=OLMTFS`

[2] `http://ttic.uchicago.edu/~argyriou/code/mtl_feat/school_splits.tar`

features. Features that are constant in each school in a certain year are: percentage of students eligible for free school meals, percentage of students in VR band one (highest band in a verbal reasoning test), school gender (S. GN.) and school denomination (S. DN.). Student-specific features are: gender (GEN), VR band (values are 1, 2, or 3) and ethnic group (EG). For the categorical features, we transform them into binary (dummy) variables and totally form 27 features as that in [5, 55]. Here, each school is taken as "one task". Hence, we obtain 139 tasks.

Following the same evaluation criterion in [5, 11, 56], we employ the explained variance, one minus the mean squared test error over the total variance of the data (computed within each task), and the percentage of variance explained by the prediction model. A large explained variance indicates better performance. This measure corresponds to a percentage version of the standard R^2 error measure for regression on the test data [11].

Since the task is a regression problem to predict the exam scores of the students, we use squared loss in the algorithms. In the training, we randomly generate 20 sets of training data and apply the rest data as the test data. The number of training data is set to the same, half of the minimum number of data among all individual tasks, which meets the requirement of Algorithm 2 that there is an instance in a task at each iteration.

Table 4.1 reports the best performance on the five compared algorithms and lists the corresponding parameters. For the batch-mode algorithms, the best results are obtained by tuning the parameters λ in a hierarchical scheme, from a large searching step in the whole parameter space to a small searching step in a small region. As a reference, the largest λ making all the learning weights of the aMTFS vanish is about 1,000 and is about 100 for the iMTFS, respectively. For the online algorithms, the parameters are tuned by the grid search scheme. λ is searched from $\{0.1, 1, 10, 20, 30, 40\}$. γ is searched from 0.1, 1, 10 to 100

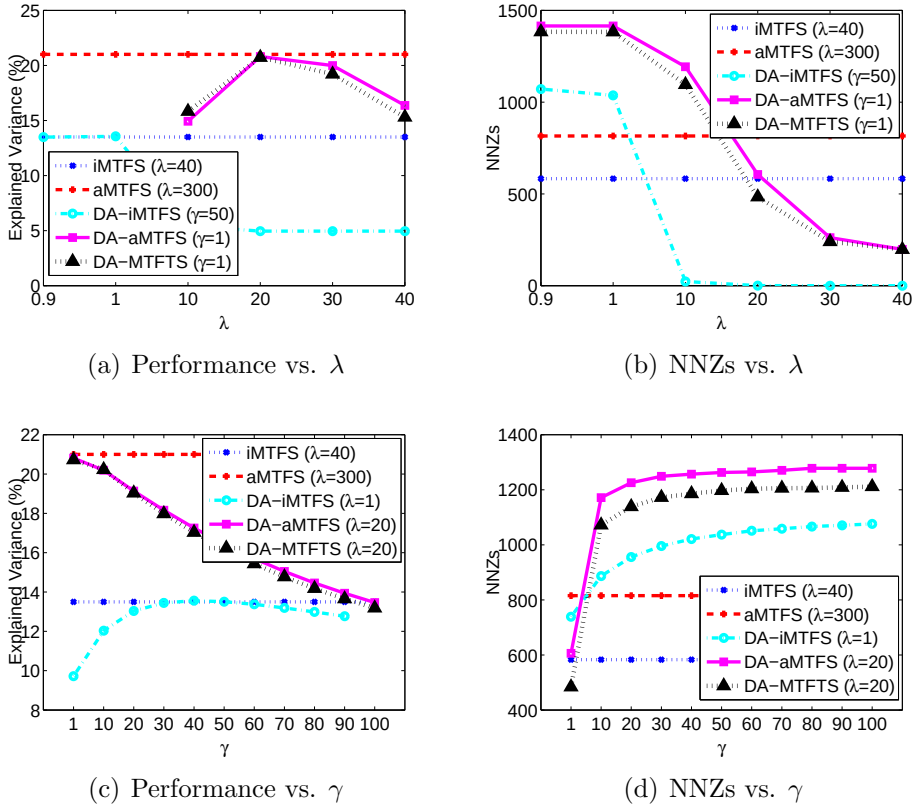
Table 4.1: Explained variance and the corresponding NNZs obtained by different methods on the school data.

Method	Explained variance	NNZs	Parameters
aMTFS	**21.0**±1.7	815.5±100.6	$\lambda = 300$
iMTFS	13.5±1.8	583.0±16.6	$\lambda = 40$
DA-aMTFS	**20.8**±1.8	605.8±180.3	$\lambda = 20, \gamma = 1$, ep=120
DA-MTFTS	**20.8**±1.9	483.7±130.7	$\lambda = 20, \gamma = 1$, ep=120
DA-iMTFS	13.5±1.8	1037.1±21.4	$\lambda = 1, \gamma = 50$, ep=120

with each increment being 10. The number of epoches is tested from 1 to 120. Here, epoch is a traditional concept in online algorithms. Multiple epoches mean that cycling through all the training examples multiple times with a different random permutation for each epoch. The sparse parameter in the DA-MTFTS is set to 0.01 at each task for simplicity in all the experiments. Here, we do not put much effort on tuning the sparse parameter for the DA-MTFTS since by this simple setting, we can achieve good performance.

There are several observations from Table 4.1. First, the results of the aMTFS vs. the iMTFS and the DA-aMTFS/DA-MTFTS vs. the DA-iMTFS clearly show that learning multiple tasks simultaneously can gain over 50% improvement than learning the task individually. Second, the DA-aMTFS and the DA-MTFTS attain the same explained variance, which is nearly the same as that obtained by the aMTFS. Both the number of non-zeros (NNZs) in weights obtained by the DA-aMTFS and the DA-MTFTS is less than that obtained by the aMTFS. More specially, the NNZs of the DA-aMTFS is about 25% less than that of the aMTFS. The DA-MTFTS gets fewer NNZs than the DA-aMTFS, about 20% decrease in the number. This indicates

that the learned DA-aMTFS and the DA-MTFTS are easier to be interpreted. Third, the DA-iMTFS obtains the same performance as that of the iMTFS and selects more NNZs than the iMTFS.



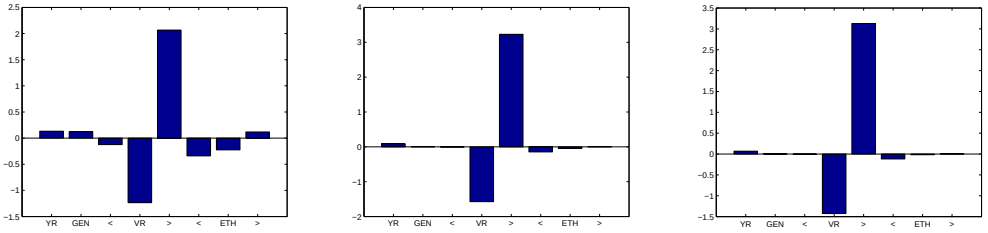
(a) Performance vs. λ (b) NNZs vs. λ

(c) Performance vs. γ (d) NNZs vs. γ

Figure 4.2: Trade-off results on the school data with varying regularizer parameter λ and the online algorithm parameter γ.

Figure 4.2 further shows the trade-off between the regularizer parameter λ and the algorithm parameter γ. The test first fixes one parameter to their best ones and varies the other. The best results of the batch-mode trained models are also shown for reference. From the results, we know that the number of non-zero elements (NNZs) decreases as λ increases for all three online algorithms. The best results are obtained when $\lambda = 1$ for

the DA-iMTFS and when $\lambda = 20$ for both the DA-aMTFS and the DA-MTFTS. By varying γ, it is shown that NNZs increases as γ increases. The best ones are obtained when $\gamma = 50$ for the DA-iMTFS and when $\gamma = 1$ for the DA-aMTFS and the DA-MTFTS. The results indicate that usually for a given dataset, the best λ and γ have to be tuned based on the given data.



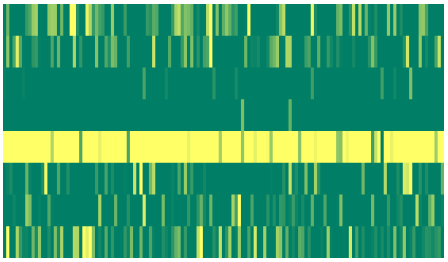
(a) aMTFS (b) DA-aMTFS (c) DA-MTFTS

Figure 4.3: The 8 most important features learned commonly across all 139 schools(tasks) are shown.

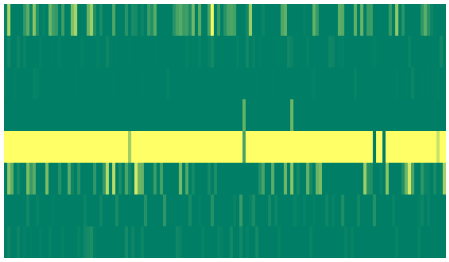
Figure 4.3 shows the learned features across all tasks by the aMTFS, the DA-aMTFS, and the DA-MTFTS, respectively. The results indicate that features learned from the online algorithms are consistent to those learned from the batch-training algorithm. That is, the predicted exam score depends very strongly on the students' VR band and it is influenced secondary by ethnic background and year admission of the students. A difference between the DA-aMTFS and the DA-MTFTS is that DA-MTFTS gets smaller values in the weight matrices and achieves more sparsity. This again verifies the results in Table 4.1 and the learned weight matrices shown in Fig. 4.4.

4.7.2 Conjoint Analysis

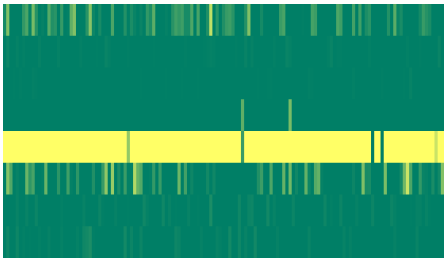
In the following, we conduct empirical study to demonstrate the characteristics and merits of the online learning algorithms for



(a) aMTFS



(b) DA-aMTFS



(c) DA-MTFTS

Figure 4.4: Weight matrices with the 8 most important features learned from the aMTFS, the DA-aMTFS, and the DA-MTFTS on the school dataset, resepctively are shown. Here, the values of the coefficients are represented by different colors. Brighter color means larger value while darker color means smaller value.

conjoint analysis on a real-world survey dataset. The objective of conjoint analysis is to estimate respondents' partworths vectors (weight matrices) while revealing the salient attributes dominating respondents' utilities [1]. Here, we test on a real-world dataset about MBA students' rating on personal computers [102], which consists of 180 students who rated the likelihood of purchasing one of 20 different personal computers. From this dataset, we have $Q = 180$. The output (response) is an integer rating on an 11-point scale (0 to 10). The input consists of 13 binary attributes and a bias term. The binary attributes include telephone hot line (TE), amount of memory (RAM), screen size (SC), CPU speed (CPU), hard disk (HD), CDROM/multimedia (CD), cache (CA), color (CO), availability (AV), warranty (WA), software (SW), guarantee (GU) and price (PR). As in [5], we use the first 8 examples per respondent as the training data and the last 4 examples per respondent as the test data. The root mean square errors (RMSEs) of the predicted from the actual ratings for the test data are averaged across all respondents to measure the error.

To simulate the online procedure on the collecting survey data, we let the training data come one example per respondent sequentially. To attain good performance, the regularizer parameters for the batch-mode trained algorithms are tuned by cross validation in a hierarchical search. The regularizer parameters and the algorithm parameters for the online-mode trained algorithms are tuned by cross validation in grid search [180]. Similarly, we set the sparse parameter of the DA-MTFTS to 0.01 at each task for simplicity.

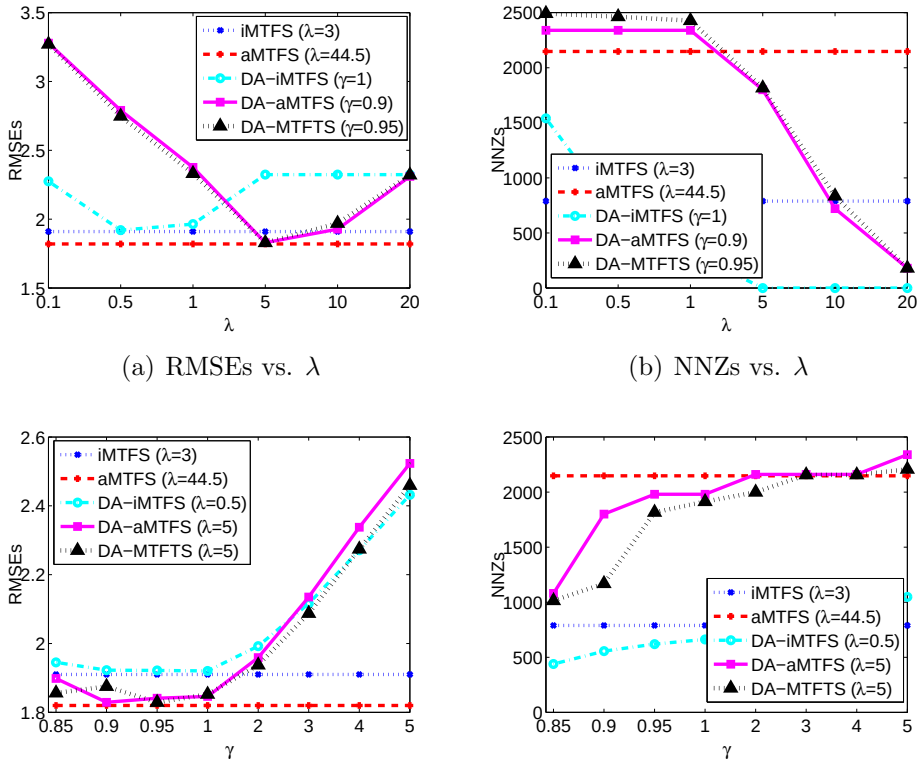
Table 4.2 reports the results obtained by the compared methods and the corresponding parameters. From these results, we have the following observations. First, learning partworths vectors across respondents can help to improve the performance. For batch-mode trained methods, the result improves from 1.91

Table 4.2: RMSEs and the corresponding NNZs obtained by different methods on the conjoint analysis dataset.

Method	RMSEs	NNZs	Parameters
aMTFS	**1.82**	2148	$\lambda = 44.5$
iMTFS	1.91	789	$\lambda = 3$
DA-aMTFS	**1.83**	1800	$\lambda = 5, \gamma = 0.9$, ep=20
DA-MTFTS	**1.83**	1816	$\lambda = 5, \gamma = 0.95$, ep=20
DA-iMTFS	1.92	662	$\lambda = 0.5, \gamma = 1.0$, ep=20

for the iMTFS to 1.82 for the aMTFS, about 5% improvement. For online-mode trained algorithms, the improvement is from 1.92 from the DA-iMTFS to 1.83 from the DA-aMTFS and the DA-MTFTS, which gains about 5% improvement. Second, the DA-aMTFS and the DA-MTFTS achieve the best results by a slight different γ value.

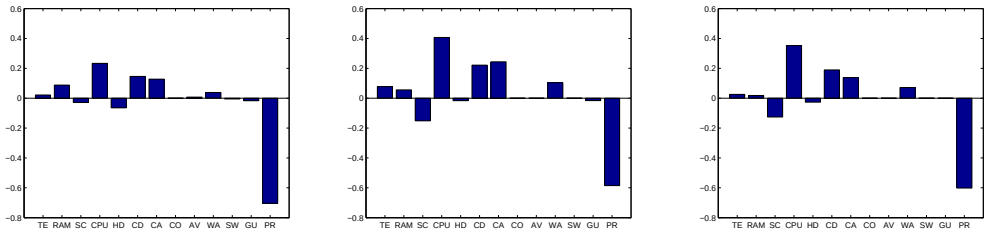
Again, we test the trade-off between the regularizer parameter λ and the algorithm parameter γ in the following. Here, we first fix γ to their best ones, and vary λ in $\{0.1, 0.5, 1, 5, 10, 20\}$ for both online learning algorithms. Fig. 4.5(a) and Fig. 4.5(b) show the results on this test (The best results of the batch-mode trained models are also shown for reference). It is shown that the number of non-zero elements (NNZs) in the partworths vectors decrease as λ increases. The best results are obtained when $\lambda = 0.5$ for the DA-iMTFS and when $\lambda = 5$ for the DA-aMTFS and the DA-MTFTS. In the second test, we fixe the best λ's for the corresponding models and changes γ in $\{0.85, 0.9, 0.95, 1, 2, 3, 4, 5\}$. It is shown that NNZs increases as γ increases. The best ones are obtained when $\gamma = 1$ for the DA-iMTFS, when $\gamma = 0.9$ for the DA-aMTFS, and when $\gamma = 0.95$ for the DA-MTFTS. The results again indicate that usually for a given dataset, the best λ and γ have to be tuned based on the given data.



(a) RMSEs vs. λ (b) NNZs vs. λ

(c) RMSEs vs. γ (d) NNZs vs. γ

Figure 4.5: Testing results on the conjoint analysis with the varying of the regularizer parameter λ and the online algorithm parameter γ.



(a) aMTFS (b) DA-aMTFS (c) DA-MTFTS

Figure 4.6: The most important features learned commonly across all 180 persons(tasks) are shown.

We further plot the learned features across all tasks by the aMTFS, the DA-aMTFS, and the DA-MTFTS in Figure 4.6. This shows that all methods obtain nearly the same important features. The results indicate that the students' ratings are strongly negative to the price and they are positive to the RAM, the CPU speed, CDROM, and cache. Figure 4.6 also shows the difference between the DA-aMTFS and the DA-MTFTS. That is, the DA-MTFTS yields more sparse than the DA-aMTFS and its coefficients are general in smaller mathan the DA-aMTFS. This can be seen from Fig. 4.7 that the weight matrix learned by the DA-aMTFT is brighter than that learned by the DA-MTFTS.

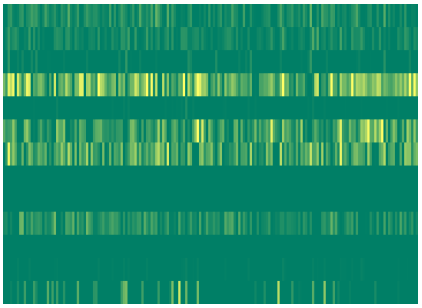
4.7.3 Time Cost Issue

When executing an algorithm, the time cost includes the time of loading data and the running time of the algorithm. Here, we emphasize again that an advantage of the online algorithms is that they are low-cost in loading the data, a very favorite property for training large-scale datasets. On the contrary, the batch-mode training algorithms may have the risk of out-of-memory when the size of training dataset is large.

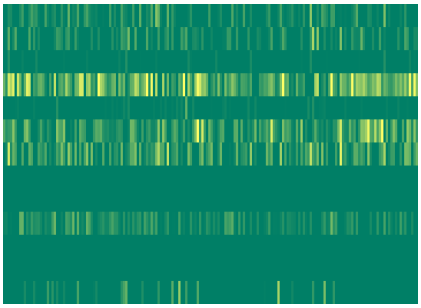
Table 4.3: Time cost in different datasets.

Dataset	aMTFS (s)	DA-MTFTS (s)	Reduction ratio
School	1.30	0.99	34.3%
Conjoint	0.162	0.115	40.9%

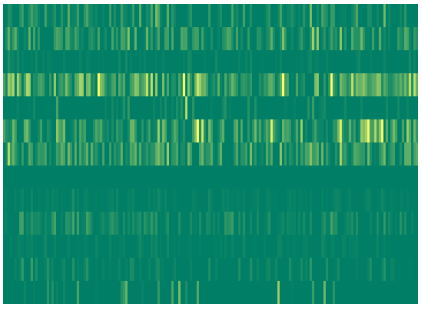
In terms of the running time of the algorithms, it is usually very difficult to carry out a fair comparison among different algorithms, due to the implementation issue, the choice of algorithm parameters, and different stopping criteria. A theoretical analysis of the convergence rate of the algorithms has been an-



(a) aMTFS



(b) DA-aMTFS



(c) DA-MTFTS

Figure 4.7: Weight matrices learned from the aMTFS, the DA-aMTFS, and the DA-MTFTS on the conjoint analysis dataset, resepctively.

alyzed in our paper and can be referred to the corresponding papers in [5, 107, 121]. As a reference, Table 4.3 reports the running time of the DA-MTFTS, the slowest MTFS online algorithm, and the aMTFS [5] with the setting of "epsilon_init=0", "iterations=50", and "method=3". It is shown that all the online learning algorithms cost less time than the batch ones. In addition, we will emphasize that another efficiency of the online learning algorithms lies that they can update the weights as a new data comes while maintaining the good performance. On the contrary, the batch-trained algorithms have to train the models starting from scratch when a new data arrives. The total time for the batch-trained algorithms will be accumulated.

4.8 Summary

In this work, we we study the multi-task feature selection problem, which has been previously applied in various applications. More specially, we propose the first online learning framework to solve the MTFS models, including a new developed MTFS model which can seek both important features and important tasks for the selected features. We derive closed-form solutions to update the weights of three MTFS methods, which makes the online learning algorithm work very efficiently in both time cost and memory cost. Moreover, we provide theoretical results for the online learning algorithms. Our detailed empirical evaluation demonstrates the characteristics and merits of the proposed online MTFS algorithms in various aspects.

There are several future work directions associated with this work. First, it is interesting to extend the current linear MTFS methods to non-linear forms by the projection method to improve the model performance. Second, as our proposed online algorithms require each instance from each task at a round, it is interesting to know how to balance the weight update when

the instances of some tasks do not appear. Third, the proposed online algorithm framework assumes the training samples are independent and identically-distributed. It is attractive to consider the case where the i.i.d. assumption does not hold in practice.

□ **End of chapter.**

Chapter 5

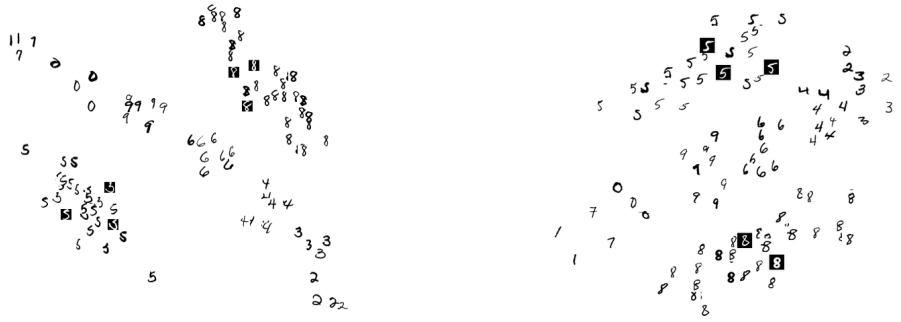
Maximum Margin Semi-supervised Learning With Irrelevant Data

5.1 Introduction

Traditional classification techniques need a large number of labeled data in the training to achieve good performance [53, 99, 137]. However, labeling data is usually difficult and time consuming. On the contrary, unlabeled data are relatively easy to acquire and they may provide useful information, e.g., data distribution, or data-dependent regularization, to improve classification performance. Hence, semi-supervised learning algorithms have been proposed to learn from both labeled and unlabeled data [28, 195, 197] and applied in various applications [7, 44, 83, 120].

In ordinary semi-supervised learning, unlabeled data are assumed relevant to the target task [3, 13, 161]. That is, they are assumed from the same distribution as the labeled data in the target task. This assumption means that the unlabeled data need well prepared [29, 195], excluding irrelevant data, or the data with distributions different from the labeled data. However, this is not in general true for real world applications.

Practically, it is easy to incorporate other irrelevant data into unlabeled data. For example, when one crawls web pages of some topics, it is easy to collect irrelevant web pages as unlabeled data; when classifying two handwritten digits, e.g., "5" and "8", with the help of unlabeled handwritten digits, it is also possible to include other digits, e.g., "3", or "4", as unlabeled data; see Figure 5.1 as an illustration. In these situations, when learning from both labeled and mixed unlabeled data, the irrelevant unlabeled data, which do not fit the assumption of previously proposed semi-supervised learning algorithms, indeed do harm to the consequently suggested semi-supervised learning models [115, 145].



(a) USPS (b) MNIST

Figure 5.1: Two benchmark handwritten digit datasets illustration: block digits are labeled data on the target task, while black digits are mixed unlabeled data.

To distinguish relevant and irrelevant data from the unlabeled data, we propose a novel maximal margin classifier, namely the *tri-class support vector machine* (3C-SVM), to learn a decision boundary from both labeled and mixed unlabeled data simultaneously. The idea is to introduce a new min loss function to measure the empirical risk on the unlabeled data. When an unlabeled data point is assumed the from relevant data, its em-

pirical risk is measured by the symmetrical hinge loss. When an unlabeled data point is assumed from the irrelevant data, its empirical risk is determined by the ε-insensitive loss function. This min loss counts the error of an unlabeled data point based on which error is smaller when the unlabeled data point is assigned to an associated class. The motivation of adopting the min loss function is based on two principles: First, from logistic regression perspective [71, 125], when a data point lies farther away from the decision boundary, it can be classified into ± 1-class with more confidence. That is, it is relevant to the target task. Second, following the maximum entropy principle, a classifier should rely more on the labeled data and the relevant data, while maximally ignoring the irrelevant data. Hence, we should make the irrelevant data close to the decision boundary and the min loss function can achieve these two principles. In some cases, the irrelevant data may be scattered around. To attain this condition, it corresponds to seeking a good kernel to represent the data similarity.

There are several advantages of our proposed 3C-SVM algorithm. First, the 3C-SVM provides a framework to generalize several popular maximum margin classifiers, including standard SVMs, Semi-supervised SVMs (S^3VMs), and SVMs learned from universum ($\mathcal{U}$-SVMs). Second, a theoretical analysis on the 3C-SVM indicates that the irrelevant data play the role of seeking a good subspace and guarantee why the 3C-SVM works. Third, the 3C-SVM is solved by a standard concave-convex procedure (CCCP) [185] and only requires to solve several quadratic programming (QP) problems, with the same worst case time complexity as that of S^3VMs [36]. It is a highly scalable algorithm currently existed. Fourth, empirical evaluation results on both synthetic and two benchmark hand-written digit datasets have shown the effectiveness and efficiency of the 3C-SVM algorithm.

The rest of this chapter is organized as follows: In Section 5.2,

we review the related work on learning from both labeled and universum data. In Section 5.3, the proposed 3C-SVM with its properties is presented. In Section 5.4, we detail how to solve the 3C-SVM algorithm through CCCP. We report the experimental comparison and results in Section 5.5 and conclude the paper in Section 5.6.

5.2 Related Work

In this section, we review some ideas in learning a binary classifier with auxiliary data.

$\mathcal{U}$-SVMs have been formulated in [161], implemented in [165] and further studied in [146]. The authors derive a new inductive principle by learning from the labeled data and the universum, a third kind of data whose distribution is different from neither of the ± 1-class. These models have to specify the universum data explicitly and choose them carefully. Those relevant unlabeled data, which are beneficial to S^3VMs, will contrarily hurt the $\mathcal{U}$-SVM eventually; see dash-dot line in Figure 5.2 for an illustration comparison.

Graph-based semi-supervised learning models have been proposed in the literature [44, 187, 195, 197]. In [187], the authors incorporate the manifold regularization [13, 86] with minimizing the empirical risk on universum data to learn a classifier on a manifold. Their method has to specify the universum data explicitly and also cannot tackle the problem of mixed unlabeled data.

There are also other algorithms concerning three kinds of data. In [192], an algorithm combining ν-support vector classification with ν-support vector regression [138], is proposed to solve the multiclass problem in one-vs-one strategy. In the learning, they additionally minimize the distance of the separating hyperplane to the examples that are in neither of the classes with

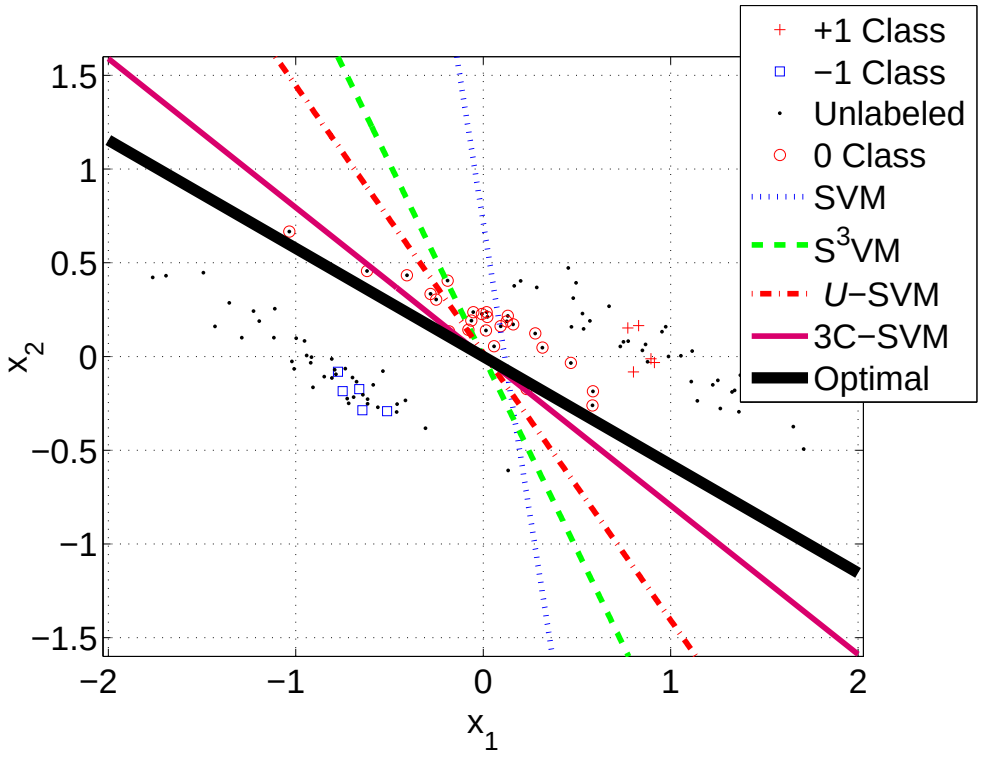


Figure 5.2: Illustration of different classifiers in $\mathbb{R}^2$ space. The 3C-SVM (the thin solid line) achieves the best result, which is closest to the Bayesian optimal classifier (the thick solid line), among all other maximum margin classifiers and automatically distinguishes the irrelevant unlabeled data (black dots with red circles) correctly.

the aim of sharpening the contrast between the different binary classifiers. The model in [192] has to know all the label information of training samples. Its setup is different from that of semi-supervised learning. In [98], a Gaussian process approach, which can be viewed as the parallel of Transductive SVM [84] is proposed to solve the semi-supervised learning task. Different from a standard Gaussian process, this model introduces a "null category noise model", which maps the hidden continuous variable to three instead of two labels, where the label "0" is assigned when the hidden variable is around zero. Further, the unlabeled data points are restricted not to take the label 0.

This model intends to omit those unlabeled data close to the decision boundary and does not utilize all available data, which may miss some useful information among them. In [74], a transductive method is proposed to learn the labels of test data by training on the labeled, the test data and the universum data simultaneously. The model is transformed into a Semi-Definite Program (SDP) problem [22], whose time complexity scales to $\mathcal{O}((L+U^2)^2(L+U)^{2.5})$, the same as that in the relaxed transductive SVM by SDP [28]. Here, L and U are the corresponding number of labeled and unlabeled training data. This is very time consuming and cannot solve large scale datasets and therefore, it cannot utilize those abundant unlabeled data sufficiently.

5.3 Learning with Irrelevant Data

5.3.1 Problem Statement and Formulation

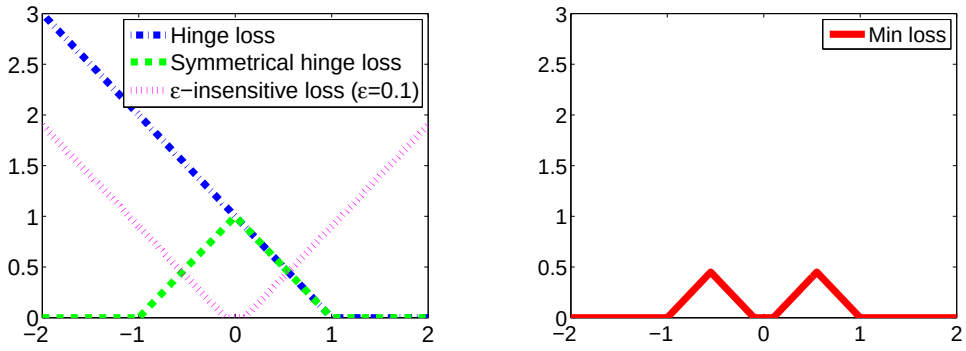
Suppose we are given L labeled samples, $\mathcal{L} = \{(\mathbf{x}_i, y_i)\}_{i=1}^{L}$, and U unlabeled samples, $\mathcal{U} = \{\mathbf{x}_i\}_{i=L+1}^{L+U}$, where $\mathbf{x}_i \in \mathcal{X} \subseteq \mathbb{R}^d$, and the labels are triple, i.e., $y_i \in \{-1, 0, 1\}$. Here, the labeled data consist of two sets of data, $\mathcal{L}_{\pm 1}$ and $\mathcal{L}_0$, where data in $\mathcal{L}_{\pm 1}$ follows the same distribution in the target task and they are labeled by -1 or $+1$; while data in $\mathcal{L}_0$ are irrelevant to the target task. That is, data with distributions different from the labeled target data are all cast into 0-class to construct the $\mathcal{L}_0$ dataset. Similarly, unlabeled data are a mixture of these data. We denote them as $\mathcal{U} = \mathcal{U}_{\mathcal{L}} \cup \mathcal{U}_0$, where data in $\mathcal{U}_{\mathcal{L}}$ follow the same distribution of ± 1 data, and data in $\mathcal{U}_0$ follow distributions different from the ± 1 data, but the same as the $\mathcal{L}_0$ data. Normally, the number of unlabeled data is much larger than the number of the labeled target data, i.e., $|\mathcal{L}_{\pm 1}| \ll U$, and given an unlabeled data point, one does not know whether it comes from $\mathcal{U}_{\mathcal{L}}$ or from $\mathcal{U}_0$.

Here, the goal is to seek a decision boundary, $f_{\vartheta}(\mathbf{x}) = \mathbf{w}^{\top}\phi(\mathbf{x}) +$

b, to classify the ± 1 data well with the help of given labeled and mixed unlabeled data, where $\boldsymbol{\vartheta} = (\mathbf{w}, b)$ and $\phi : \mathbb{R}^d \to \mathbb{R}^f$, is a feature mapping function often implicitly defined by a Mercer kernel [137, 160]. Hence, we formulate the objective as follows:

$$\min_{\boldsymbol{\vartheta}} \frac{\lambda}{2}\|\mathbf{w}\|^2 + \sum_{\mathbf{x}_i \in \mathcal{L}} r_i \ell_L(f_{\boldsymbol{\vartheta}}(\mathbf{x}_i), y_i) + \sum_{\mathbf{x}_i \in \mathcal{U}} r_i \ell_U(f_{\boldsymbol{\vartheta}}(\mathbf{x}_i)), \quad (5.1)$$

where λ is a trade-off constant for the regularization term. $\ell_L(\cdot, \cdot)$ is a loss function to measure the empirical risk of the labeled data and $\ell_U(\cdot)$ is a loss function to measure the empirical risk of the unlabeled data. $r_i, i = 1, \ldots, L+U$, is a ratio penalty to balance the loss on the point $\mathbf{x}_i$ and the regularization term.



(a) Loss functions (b) Loss for mixed unlabeled data

Figure 5.3: Illustration of loss functions. The min loss is the minimum between the symmetrical hinge loss and the ε-insensitive loss.

Typically, one may choose different loss functions to measure the empirical risk on the given data. These loss functions include

- **Hinge loss:** $H_1(u) = \max\{0, 1-u\}$, a loss function has been used to measure the empirical risk of labeled data in standard SVMs [160]; see dash-dot line in Figure 5.3(a).
- **Symmetric hinge loss:** $H_1(|\cdot|)$, a loss function has been applied to measure the empirical risk on unlabeled data for S^{3}VMs [36]; see dashed line in Figure 5.3(a).

- **ε-insensitive loss:** $I_\varepsilon(u) = \max\{0, |u| - \varepsilon\}$, a loss function has been adopted to measure the empirical risk in Support Vector Regression [160] and the Universum data in $\mathcal{U}$-SVMs [165]; see dotted line in Figure 5.3(a).

In practical scenarios, the unlabeled data may be a mixture of data relevant or irrelevant to the target task. How to distinguish them correctly is a very difficult task. Here, we are based on the following two principles. First, from logistic regression perspective [71, 125], when a data point lies farther away from the decision boundary, the data are classified as ± 1-class with more confidence; while data points lie near the decision boundary, their classification on ± 1-class is of less confidence. Hence, ideally, data from ± 1-class should lie on or outside of the margin gap; while other irrelevant data are close to the decision boundary. Second, the maximum entropy principle indicates that a classifier should rely more on the labeled and relevant data, while maximally ignoring the irrelevant data. These two principles indicate that irrelevant data should lie around the sought decision boundary.

In order to fulfill the above objectives, we adopt a min loss function to measure the risk of unlabeled data, so as to separate the unlabeled data into relevant and irrelevant data. The loss of unlabeled data is determined by the smaller value of which measured by the symmetric hinge loss and by the ε-insensitive loss; see Figure 5.3(b). Hence, for an unlabeled data point, when the error measured by the ε-insensitive loss is smaller than the error measured by the symmetric hinge loss, we can deem it as irrelevant data; otherwise, we set it as relevant data.

With this loss function, we can develop a novel maximum margin classifier, named the *tri-class support vector machine*

(3C-SVM), as follows:

$$\min_{\boldsymbol{\vartheta}} \frac{\lambda}{2}\|\mathbf{w}\|^2 + \sum_{\mathbf{x}_i \in \mathcal{L}_{\pm 1}} r_i H_1(y_i f_{\boldsymbol{\vartheta}}(\mathbf{x}_i)) + \sum_{\mathbf{x}_i \in \mathcal{L}_0} r_i I_\varepsilon(f_{\boldsymbol{\vartheta}}(\mathbf{x}_i)) + \sum_{\mathbf{x}_i \in \mathcal{U}} r_i \min\{H_1(|f_{\boldsymbol{\vartheta}}(\mathbf{x}_i)|), I_\varepsilon(|f_{\boldsymbol{\vartheta}}(\mathbf{x}_i)|)\}. \tag{5.2}$$

In the above, the first two terms correspond to the formulation of a standard SVM [160]. The third term measures the empirical risk of $\mathcal{L}_0$ data, the same in $\mathcal{U}$-SVMs [165]. The last term measures the loss of unlabeled data.

5.3.2 Properties of 3C-SVMs

We first list the properties of the 3C-SVM, then outline the intuition behind 3C-SVMs through a specific case with $r_i = \infty$ for unlabeled data and $\varepsilon = 0$.

Our 3C-SVM provides a framework for the following popular maximum margin classifiers:

1. A standard SVM formulation [160] is a special case of the 3C-SVM. This can be attained by setting r_i to zero for the third and fourth terms in (5.2). When only labeled data are given in the training set, we can adopt this formulation.

2. An S^3VM formulation [28] is a special case of the 3C-SVM. This can be achieved by setting r_i to zero in the third term and using only symmetrical hinge loss to measure the empirical risk of unlabeled data in the fourth term in (5.2). When only labeled data and relevant unlabeled data are given, we can use this formulation.

3. The 3C-SVM also includes a $\mathcal{U}$-SVM [165]. It can be easily obtained by setting r_i to zero in the fourth term of (5.2). This formulation works when only labeled data and universum data are given.

Hence, our 3C-SVM, a general maximum margin semi-supervised learning formulation, includes standard SVMs, S^{3}VMs, and $\mathcal{U}$-SVMs as its special cases. A summarization is shown in Table 5.1.

Table 5.1: Relation between different models and the training data.

3C-SVM			
$\mathcal{L}$	-1	0	1
$\mathcal{U}$	-1	0	1

SVM			
$\mathcal{L}$	-1		1
$\mathcal{U}$			

S^{3}VM			
$\mathcal{L}$	-1		1
$\mathcal{U}$	-1		1

$\mathcal{U}$-SVM			
$\mathcal{L}$	-1	0	1
$\mathcal{U}$			

It should be noted that the 3C-SVM can map the data to a feature space through the kernel trick. This can tackle the problem when data in $\mathcal{U}_0$ do not lie near the middle of the two kinds of labeled target data in the original data space.

We now study how the 3C-SVM can work and give an insight of the model in the following theorem:

Theorem 5. *A 3C-SVM with $r_i = \infty$ for unlabeled data and $\varepsilon = 0$ is equivalent to one of the following two cases: 1) training a general S^{3}VM to keep the unlabeled data falling on or out of the margin gap with only one or none of the unlabeled data in the decision boundary; or 2) separating the unlabeled data into two sets, $\mathcal{U}_{\mathcal{L}}$ and $\mathcal{U}_0$ with $|\mathcal{U}_0| \geq 2$, and training a general S^{3}VM on the training data projected onto the orthogonal complement of span $\{\phi(\mathbf{x}_j) - \phi(\mathbf{x}_0),\ \mathbf{x}_j \in \mathcal{U}_0\}$, where $\mathbf{x}_0$ is an arbitrary data point from $\mathcal{U}_0$, while keeping the unlabeled data in the set of $\mathcal{U}_{\mathcal{L}}$ falling on or out of the margin gap.*

Proof. $r_i = \infty$ for $\mathcal{U}$ data and $\varepsilon = 0$ imply that the min term in the fourth term of (5.2) vanish and the optimal solution of $\mathbf{w}$ and b in (5.2) is attained when one of the following conditions is fulfilled: (a) $|\mathbf{w}^\top \phi(\mathbf{x}_j) + b| \geq 1$, or (b) $\mathbf{w}^\top \phi(\mathbf{x}_j) + b = 0$. Hence, the above conditions set up the criterion of separating the unlabeled data into two sets, $\mathcal{U}_{\mathcal{L}}$ and $\mathcal{U}_0$, where data in $\mathcal{U}_{\mathcal{L}}$

satisfy the condition of (a) and data in $\mathcal{U}_0$ satisfy the condition of (b).

First, if $|\mathcal{U}_0| = 0$ or 1, it leads to the result of case 1) in the above theorem. Here, a general S^{3}VM means that it is a generalization of the S^{3}VM and the $\mathcal{U}$-SVM.

Next, if $\mathcal{U}_0$ contains at least two samples. For the data $\mathbf{x}_j$ from $\mathcal{U}_0$, we have $\mathbf{w}^\top\phi(\mathbf{x}_j) + b = 0$. Hence, picking arbitrary data $\mathbf{x}_0$ from $\mathcal{U}_0$, we obtain $\mathbf{w}^\top(\phi(\mathbf{x}_j) - \phi(\mathbf{x}_0)) = 0$. That is, $\mathbf{w}$ is orthogonal to span $\{\phi(\mathbf{x}_j) - \phi(\mathbf{x}_0),\ \mathbf{x}_j \in \mathcal{U}_0\}$. Now, let $P_{\mathcal{U}_0^\perp}$ denote an orthogonal project on the orthogonal complement of the mapped set $\mathcal{U}_0$, we have $\mathbf{w} = P_{\mathcal{U}_0^\perp}\mathbf{w}$, $\mathbf{w}^\top\mathbf{w} = \mathbf{w}^\top P_{\mathcal{U}_0^\perp}^\top P_{\mathcal{U}_0^\perp}\mathbf{w} = \mathbf{w}^\top\mathbf{w}$, and $\mathbf{w}^\top\mathbf{x}_i = \mathbf{w}^\top P_{\mathcal{U}_0^\perp}^\top\mathbf{x}_i = \mathbf{w}^\top P_{\mathcal{U}_0^\perp}\mathbf{x}_i$. This means that the optimal $\mathbf{w}$ is sought by training a general S^{3}VM on the projected labeled data and $\mathcal{U}_\mathcal{L}$ data with projection by $P_{\mathcal{U}_0^\perp}$ while keeping the condition (a) valid, or other unlabeled data falling on or out of the margin gap. □

Theroem 5 clearly shows that the optimization of our proposed model is to eventually find the most suitable subspace in which the margin is maximized while the overall empirical risk is minimized. The irrelevant data play the role of finding the subspace.

5.4 Solution and Computation

Due to the non-convexity of the min loss function, the formulation of the 3C-SVM in (5.2) is non-convex in general, which is the same as S^{3}VMs [15, 84]. Moreover, there are two difficulties to be solved in the formulation: the min term and the absolute operation on the unlabeled data. In the following, we show how to solve these two difficult problems.

5.4.1 Elimination of Min Terms and Absolute Values

First, we introduce decision variables, $d_k \in \{0, 1\}$, to remove the min term. This trick is similar to the L_1-norm S^3VM in [15]. We then transform the optimization as follows:

$$\begin{aligned}\min_{\boldsymbol{\vartheta},\mathbf{d}} \frac{\lambda}{2}\|\mathbf{w}\|^2 &+ \sum_{\mathbf{x}_i\in\mathcal{L}_{\pm1}} r_i H_1(y_i f_{\boldsymbol{\vartheta}}(\mathbf{x}_i)) + \sum_{\mathbf{x}_i\in\mathcal{L}_0} r_i I_\varepsilon(f_{\boldsymbol{\vartheta}}(\mathbf{x}_i)) \\ &+ \sum_{\mathbf{x}_{k+L}\in\mathcal{U}} r_{k+L} \underbrace{H_1(|f_{\boldsymbol{\vartheta}}(\mathbf{x}_i)| + D(1-d_k))}_{Q_1} \\ &+ \sum_{\mathbf{x}_{k+L}\in\mathcal{U}} r_{k+L} \underbrace{I_\varepsilon(|f_{\boldsymbol{\vartheta}}(\mathbf{x}_i)| - Dd_k)}_{Q_2}, \end{aligned} \tag{5.3}$$

where $D > 0$ is a suitable constant making $Q_1 = 0$ when $d_k = 0$ and $Q_2 = 0$ when $d_k = 1$. That means, when $d_k = 0$, the error is counted from Q_2 and the unlabeled data are classified as 0-class; when $d_k = 1$, the error is incurred by Q_1 and the unlabeled data are classified as one of the ±1-class.

The loss in Q_1 is $H_1(|u|+a)$, or $H_1(|u|+a) = \max\{0, 1-|u|-a\} = H_{1-a}(|u|)$. It can be approximated by a symmetrical loss, which is similar to the ramp loss used in [36, 164], as follows:

$$H_1(|u| + a) \approx H_{1-a}(u) - H_\kappa(u) + H_{1-a}(-u) - H_\kappa(-u)\,.$$

For the loss in Q_2, we can transform it into another symmetrical loss as follows:

$$I_\varepsilon(|u| - a) = H_{-\varepsilon-a}(-u) + H_{-\varepsilon-a}(u)\,.$$

Due to the symmetry of the losses, we introduce new pair-data for the unlabeled data to simplify the expression as [36]. The new pair-data are

$$\begin{aligned} \mathbf{x}_{L+k} &= \mathbf{x}_{L+k}, \quad y_{L+k} = 1, \\ \mathbf{x}_{LU+k} &= \mathbf{x}_{L+k}, \quad y_{LU+k} = -1, \quad k = 1, \ldots, U, \end{aligned}$$

where LU means $L + U$.

5.4.2 Concave-Convex Procedure (CCCP)

Hence, we can transform the problem in (5.3) into $Q^{\kappa}(\boldsymbol{\vartheta}, \mathbf{d})$, which is the summation of two terms, $Q_{vex}(\boldsymbol{\vartheta}, \mathbf{d})$ and $Q^{\kappa}_{cav}(\boldsymbol{\vartheta})$. They are defined as follows:

$$\begin{aligned}
Q_{vex}(\boldsymbol{\vartheta}, \mathbf{d}) = & \frac{\lambda}{2}\|\mathbf{w}\|^2 + \sum_{\mathbf{x}_i \in \mathcal{L}_{\pm 1}} r_i H_1(y_i f_{\boldsymbol{\vartheta}}(\mathbf{x}_i)) \\
& + \sum_{\mathbf{x}_i \in \mathcal{L}_0} r_i I_{\varepsilon}(f_{\boldsymbol{\vartheta}}(\mathbf{x}_i)) \\
& + \sum_{k=1}^{U} r_{k+L} H_{1-D(1-d_k)}(y_{k+L} f_{\boldsymbol{\vartheta}}(\mathbf{x}_{k+L})) \\
& + \sum_{k=1}^{U} r_{k+L} H_{1-D(1-d_k)}(y_{k+LU} f_{\boldsymbol{\vartheta}}(\mathbf{x}_{k+LU})) \\
& + \sum_{k=1}^{U} r_{k+L} H_{-\varepsilon-Dd_k}(y_{k+L} f_{\boldsymbol{\vartheta}}(\mathbf{x}_{k+L})) \\
& + \sum_{k=1}^{U} r_{k+L} H_{-\varepsilon-Dd_k}(y_{k+LU} f_{\boldsymbol{\vartheta}}(\mathbf{x}_{k+LU}))) \,, \\
Q^{\kappa}_{cav}(\boldsymbol{\vartheta}) = & - \sum_{j=L+1}^{L+2U} r_j H_{\kappa}(y_j f_{\boldsymbol{\vartheta}}(\mathbf{x}_j)) \,.
\end{aligned}$$

Note that the above concave term, Q^{κ}_{cav}, keeps the non-convexity of the model following from the ramp loss in approximating the Q_1. The optimization in $Q^{\kappa}(\boldsymbol{\vartheta}, \mathbf{d})$ is a summation of a convex term and a concave term, or a difference of convex programming. Hence, it can be solved by the concave-convex procedure (CCCP) [185], a technique which has been adopted in large scale transductive SVMs [36] and SVMs on data with missing values [148].

In the CCCP, we need to use the first order Taylor expansion to approximate the concave term of Q^{κ}_{cav}. Since the variable

$\mathbf{d}$ does not appear in the concave term, we only need to apply the first order Taylor expansion of Q_{cav}^{κ} at $\boldsymbol{\vartheta}^t$. Hence, we can seek the optimal variables by solving a sequence of the following optimization problem:

$$\min_{\boldsymbol{\vartheta},\mathbf{d}} \left(Q_{vex}(\boldsymbol{\vartheta}, \mathbf{d}) + \frac{\partial Q_{cav}^{\kappa}(\boldsymbol{\vartheta}^t)}{\partial \boldsymbol{\vartheta}} \cdot \boldsymbol{\vartheta} \right) . \tag{5.4}$$

The above optimization is a mixed integer optimization problem since $\mathbf{d}$ is an integer vector. Here, we adopt a standard routine to solve the integer programming problem [167]: 1) relaxing the integer variable to a real variable, then solve the whole optimization together; 2) rounding the corresponding variable to get its integer solution. For our problem in (5.4), we relax the decision variable d_k from $\{0, 1\}$ to $[0, 1]$ and solve the optimization problem in (5.4) first. We then determine the value of d_k by its definition, the error incurred is less when the data are assigned to the associated class, as follows

$$d_k = \begin{cases} 1 & \text{if} \quad \xi_k \leq \xi_k^* \\ 0 & \text{otherwise} \end{cases} , \tag{5.5}$$

where $\xi_k = H_1(|f_{\boldsymbol{\vartheta}}(\mathbf{x}_{k+L})|)$ and $\xi_k^* = I_{\varepsilon}(|f_{\boldsymbol{\vartheta}}(\mathbf{x}_{k+L})|)$, $k = 1, \ldots, U$.

To simplify the first order approximation of the concave term in (5.4), we define

$$\mu_{k+\mathfrak{s}} = y_{k+\mathfrak{s}} \frac{\partial Q_{cav}^{\kappa}(\boldsymbol{\vartheta})}{\partial f_{\boldsymbol{\vartheta}}(\mathbf{x}_{k+\mathfrak{s}})} = \begin{cases} r_{k+\mathfrak{s}} \text{ if } y_{k+\mathfrak{s}} f_{\boldsymbol{\vartheta}}(\mathbf{x}_{k+\mathfrak{s}}) < \kappa \\ 0 \quad \text{otherwise} \end{cases} , \tag{5.6}$$

for those unlabeled samples $\mathbf{x}_{k+\mathfrak{s}}$ with $d_k = 1$, where $k = 1, \ldots, U$, and $\mathfrak{s}$ is L or $L + U$. Hence, the first order Taylor expansion of the concave term is then expressed as

$$\frac{\partial Q_{cav}^{\kappa}(\boldsymbol{\vartheta}^t)}{\partial \boldsymbol{\vartheta}} \cdot \boldsymbol{\vartheta} = \sum_{j=L+1}^{L+2U} \mu_j y_j f_{\boldsymbol{\vartheta}}(\mathbf{x}_j).$$

Now we turn to solve the relaxed optimization in (5.4) and summarize the result in the following theorem:

Theorem 6. *The dual problem of the relaxed optimization in (5.4) is a Quadratic Programming (QP) problem as follows:*

$$\begin{aligned} \max_{\boldsymbol{\alpha},\boldsymbol{\alpha}^*} \quad & -\frac{\lambda}{2}\|\mathbf{w}(\boldsymbol{\alpha},\boldsymbol{\alpha}^*)\|^2 + \boldsymbol{\varrho}(\boldsymbol{\alpha},\boldsymbol{\alpha}^*), \\ \text{s.t.} \quad & \mathbf{0} \le \boldsymbol{\alpha},\boldsymbol{\alpha}^* \le \mathbf{r}, \\ & \mathbf{A}_e\left[\boldsymbol{\alpha};\boldsymbol{\alpha}^*\right] = \boldsymbol{\mu}^\top \mathbf{Y}_{\bullet 2U}, \quad \mathbf{A}[\boldsymbol{\alpha};\boldsymbol{\alpha}^*] \le \mathbf{0}\,, \end{aligned} \tag{5.7}$$

where $\|\mathbf{w}(\boldsymbol{\alpha},\boldsymbol{\alpha}^*)\|^2$ *is a quadratic term and* $\boldsymbol{\varrho}(\boldsymbol{\alpha},\boldsymbol{\alpha}^*)$ *is a linear term on dual variables* $\boldsymbol{\alpha}$ *and* $\boldsymbol{\alpha}^*$*. The variable* $[\boldsymbol{\alpha};\boldsymbol{\alpha}^*]$ *consists of an* $|\mathcal{L}_0| + L + 4U$*-dimensional vector.*

Detailed expression of notation and proof are in Appendix C.

Algorithm 3 CCCP for 3C-SVMs

Initialization:
$t = 0$
Calculate $\boldsymbol{\vartheta}^0 = (\mathbf{w}^0, b^0)$ from a $\mathcal{U}$-SVM solution on the labeled/unlabeled data

Compute
$\mu_i^0 = \begin{cases} r_i & \text{if } \ y_i f_{\boldsymbol{\vartheta}^0}(\mathbf{x}_i) < \kappa \text{ and } i \ge L+1 \\ 0 & \text{otherwise} \end{cases}$
repeat
$t \leftarrow t+1$
Solve the optimization in (5.7) to obtain $\boldsymbol{\vartheta}^t$
Update $\mathbf{d}^t$ from (5.5)
Update $\boldsymbol{\mu}^t$ from (5.6)
if $Q^\kappa(\boldsymbol{\vartheta}^t, \mathbf{d}^t) > Q^\kappa(\boldsymbol{\vartheta}^{t-1}, \mathbf{d}^{t-1})$ **then**
Let $\mathbf{d}^t = \mathbf{d}^{t-1}$
Solve the optimization in (5.7) to obtain $\boldsymbol{\vartheta}^t$ by restoring $\mathbf{d}$ to $\mathbf{d}^{t-1}$
Update $\boldsymbol{\mu}^t$ from (5.6)
end if
until $|\boldsymbol{\mu}^{t+1} - \boldsymbol{\mu}^t| \le \epsilon$

Hence, we obtain $\mathbf{w}$ as a linear combination of the dual vari-

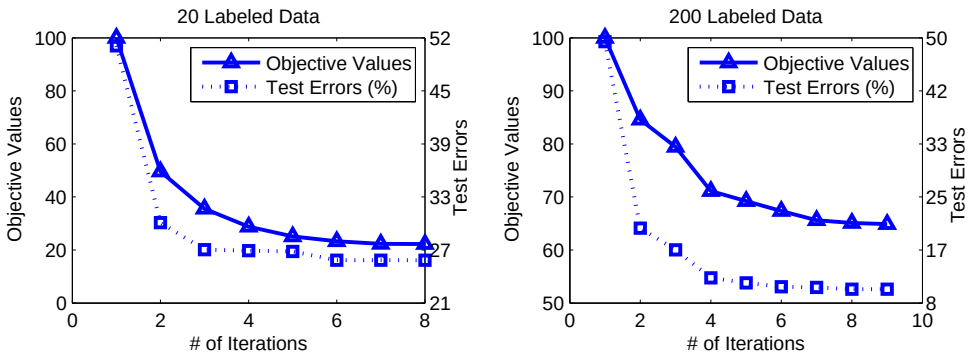


Figure 5.4: One trial for the values of the objective function and test errors in the procedure of training 3C-SVM on the toy data with 20 (left), 200 (right) labeled data and unlabeled data in the combination of $(100, 100)$ from the first kind of $\mathcal{U}_0$ data. The 3C-SVM algorithm usually converges in a few steps.

ables, $\boldsymbol{\alpha}$ and $\boldsymbol{\alpha}^*$,

$$\mathbf{w} = \frac{1}{\lambda}\left(\sum_{i=-|\mathcal{L}_0|, i\neq 0}^{L+2U} \alpha_i y_i \phi(\mathbf{x}_i) + \sum_{i=L+1}^{L+2U} (\alpha_i^* - \mu_i) y_i \phi(\mathbf{x}_i)\right), \quad (5.8)$$

and the variable b corresponds to the dual variable of the equality constraint. The form of the weight we have obtained is similar to that in [13]. We can also define the corresponding *support vectors*. They are those labeled data $\mathbf{x}_i$'s with non-zero α_i values and unlabeled data $\mathbf{x}_j$'s with non-zero $(\alpha_j + \alpha_j^* - \mu_j)$ values.

Hence, we obtain Algorithm 3 to solve the 3C-SVM algorithm. Recalling Theorem 5, we can know that, intuitively, the Algorithm 3 works in the following way: first finding out those unlabeled data which are certainly outside the margin gap, removing them from the training set; then training a $\mathcal{U}$-SVM model on the labeled data with the rest unlabeled data.

The following theorem summarizes the convergence of the Algorithm 3.

Theorem 7. *The Algorithm 3 converges in finite iterations.*

Proof. First, we prove that the objective Q^κ decreases in each iteration. From the CCCP, we have

$$\begin{aligned} Q_{vex}(\boldsymbol{\vartheta}^{t+1}, \mathbf{d}) &+ \partial Q^\kappa_{cav}(\boldsymbol{\vartheta}^t) \cdot \boldsymbol{\vartheta}^{t+1} \\ &\leq Q_{vex}(\boldsymbol{\vartheta}^t, \mathbf{d}) + \partial Q^\kappa_{cav}(\boldsymbol{\vartheta}^t) \cdot \boldsymbol{\vartheta}^t \qquad (5.9) \\ Q^\kappa_{cav}(\boldsymbol{\vartheta}^{t+1}) &\leq Q^\kappa_{cav}(\boldsymbol{\vartheta}^t) + \partial Q^\kappa_{cav}(\boldsymbol{\vartheta}^t) \cdot (\boldsymbol{\vartheta}^{t+1} - \boldsymbol{\vartheta}^t), \quad (5.10) \end{aligned}$$

where ∂Q^κ_{cav} defines the partial derivative of Q^κ_{cav} with respect to $\boldsymbol{\vartheta}$. Hence, summing (5.9) and (5.10) together, we get $Q^\kappa(\boldsymbol{\vartheta}^{t+1}, \mathbf{d}) \leq Q^\kappa(\boldsymbol{\vartheta}^t, \mathbf{d})$ for the same $\mathbf{d}$.

After rounding, the objective value Q^κ may increase, i.e., $Q^\kappa(\boldsymbol{\vartheta}^{t+1}, \mathbf{d}^{t+1})$ may be greater than $Q^\kappa(\boldsymbol{\vartheta}^t, \mathbf{d}^t)$. In order to avoid this case, we can restore $\mathbf{d}^{t+1}$ to $\mathbf{d}^t$ and seek $\boldsymbol{\vartheta}^{t+1}$ again by minimizing Q^κ with fixed $\mathbf{d}$. This additional step guarantees to decrease the objective of Q^κ at each step.

Second, the variable $\boldsymbol{\mu}$ can only take a finite number of distinct values. The algorithm converges in finite steps since Q^κ decreases in each iteration and the inequality (5.10) is strict unless $\boldsymbol{\mu}$ remains unchanged. □

Remark Note that the local optimal issue of the 3C-SVM has been alleviated by its initialization and the additional step to avoid increasing the rounded objective function is typically not needed. Our observation from the experimental results shows that our 3C-SVM works well using current initialization and the rounded objective function, $Q^\kappa(\boldsymbol{\vartheta}^t, \mathbf{d}^t)$, actually decreases in each step; see Figure 5.4.2.

Complexity Analysis Algorithm 3 has to solve a sequence of QPs in (5.7). In practice, we find that the number of iteration steps is a constant, usually less than 10; see figures (one trial result of the values of the objective function and test errors during the CCCP iterations of training 3C-SVM on toy datasets) shown in Figure 5.4.2. Thus, training a 3C-SVM is equivalent to solving a constant number of QP problems with $|\mathcal{L}_0| + L + 4U$

variables. Therefore, the 3C-SVM algorithm has a worst case complexity of $\mathcal{O}((|\mathcal{L}_0| + L + 4U)^3)$ [61, 137]. Possible tricks may be applied to speed up the 3C-SVM algorithm in a quadratic scale [36, 124, 137].

5.4.3 Balance Constraint

In the formulation of (5.2), we do not consider the balance constraint for the unlabeled data. Actually, balance constraint can be easily incorporated into our formulation.

There are two observations: 1) Data from $\mathcal{U}_{\mathcal{L}}$ need the balance constraint [161]; 2) Data from $\mathcal{U}_{\mathcal{L}_0}$ do not need the balance constraint. By Theorem 5, ideally, their decision values approach to 0. Hence, we can adopt the same balance constraint as that used in [37],

$$\frac{1}{U}\sum_{t=L+1}^{L+U} f_{\boldsymbol{\vartheta}}(\mathbf{x}_t) = \frac{1}{L}\sum_{i=1}^{L} y_i. \tag{5.11}$$

This constraint can be included in the optimization in (C.1) and rewritten into kernel form in (5.7) similar to the trick in [37]. The balance constraint in (5.11) is affected by the summation of y_i. A possibly better setting for the balance constraint is $\frac{1}{U}\sum_{t=L+1}^{L+U} f_{\boldsymbol{\vartheta}}(\mathbf{x}_t) = c$, where c is a user-specified constant related to the portion of the number of the unlabeled data assigning to the positive class [28].

5.5 Experiments

In this section, we test our proposed 3C-SVM algorithm on synthetic and real world benchmark handwritten digits recognition datasets, and compare it with an SVM implemented by LibSVM [27], an S^{3}VM [36], and a $\mathcal{U}$-SVM [165]. Our 3C-SVM

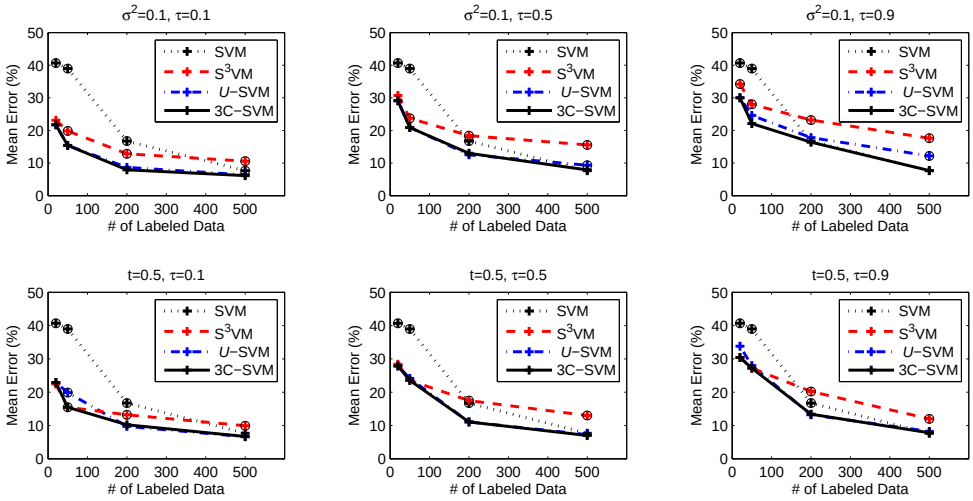


Figure 5.5: The performance of four algorithms on toy datasets with different combinations of mixed unlabeled data. The results of 3C-SVMs outperform the corresponding models with 95% significant level on paired t-test are marked by circles. 3C-SVMs consistently obtain the best results.

algorithm is implemented in Matlab 7.3 and the QP problem is solved by a general optimization toolbox, MOSEK[1].

Linear kernel is employed for the synthetic datasets and an RBF kernel, $K(\mathbf{x}, \mathbf{y}) = \exp(-\gamma\|\mathbf{x}-\mathbf{y}\|^2)$, with $\gamma = \frac{1}{0.3d}$ as [137], is applied for the real world datasets. We seek other optimal hyperparameters on the training datasets through cross-validation in an incremental way from the convex models: We first tune the soft-margin hyperparameter C for the SVM [160]; we then feed the obtained optimal C into $\mathcal{U}$-SVM and tune the parameters, ε and $C_{\mathcal{U}}$, in the $\mathcal{U}$-SVM [146, 165]. For the S^{3}VM, we further tune its parameters on all used training data. Finally, we set the parameters of our 3C-SVM based on the obtained optimal parameters from other models. More specially, λ is set to $\frac{1}{C}$, $r_i = 1$ for labeled data and $r_i = \frac{C_{\mathcal{U}}}{C}$ for unlabeled data, ε and κ are set the same as the optimal value from the corresponding

[1] `http://www.mosek.com`

model. D is set to 2.

A goal of the experiment is to test the performance of the models learning with mixed unlabeled data. We therefore test them on different combinations of mixed unlabeled data. That is, U unlabeled data are selected in the combination of $(\tau U, (1-\tau)U)$, where τU data are randomly chosen from ± 1-class and $(1-\tau)U$ data are randomly chosen from $\mathcal{U}_0$ data. τ is tested in $\{0.1, 0.5, 0.9\}$.

5.5.1 Synthetic Datasets

Data from the ± 1-class are generated following the scheme of [146], where the means are $c_i^{\pm} = \pm 0.3$ for $i = 1, \ldots, 50$ and variance values are $\sigma_{1,2}^2 = 0.08$ and $\sigma_{3,\ldots,50}^2 = 10$. In this setting, we can generate two Gaussians with the Bayes risk being approximately 5%. Two kinds of $\mathcal{U}_0$ data similar to those in [146] are generated. For the first kind, it is a zero mean with $\sigma_{1,2}^2 = 0.1$ and $\sigma_{3,\ldots,50}^2 = 10$. For the second kind, the variance values are the same as the ± 1-class data, but the mean is $\frac{t}{2} \cdot (\mathbf{c}^+ - \mathbf{c}^-)$ $(t = 0.5)$, shifted a little bit from the origin, where the optimal Bayesian classifier passes through.

In the experiment, ten sets of data with different size (20, 50, 200, 500) from ± 1-class are randomly selected as labeled training data; while ten sets of U (500) unlabeled data are selected as above mentioned combinations and additional 500 data from ± 1-class are used as test set.

Figure 5.5 shows the average performance (10 runs) of all four algorithms. 3C-SVMs consistently attain the best results. The performance of $\mathcal{U}$-SVMs decreases as the number of $\mathcal{U}_0$ data decreases and cannot beat that of SVM when the size of the labeled training data is 500; while our 3C-SVMs keep nearly the same accuracies and outperform $\mathcal{U}$-SVMs and S^3VMs when the number of labeled training data is large.

We also test our 3C-SVMs with balanced constraints but do

not get significant improvement. One reason may be in that $\mathcal{U}_0$-data has overcome the effect of imbalance in the labeled target data. Another reason may be that we need to choose a more suitable constant c, instead of just using $\sum_{i=1}^{L} y_i/L$ in the balance constraint.

We further show the objective function and test errors during the CCCP iterations of training on the toy datasets with different number ($L = 20, 200$, $U = 500$) of training data in Figure 5.4.2. The figures show that the 3C-SVM algorithm tends to converge in only a few iterations, usually less than 10.

5.5.2 Results on Real World Datasets

Table 5.2: The average (10 runs) accuracies (%) of SVMs, S^{3}VMs, $\mathcal{U}$-SVMs, and 3C-SVMs on the USPS and the MNIST ("5" vs "8") datasets for different combinations of mixed unlabeled data. The p-values of paired t-test on the 3C-SVMs scores against other algorithm's scores are given in brackets. Significant improvement with 90% confidence level and the best accuracy are in bold. The 3C-SVMs achieve significant improvement only except for $\tau = 0.9$ against SVMs, where the results of SVMs are unstable and with very large variances.

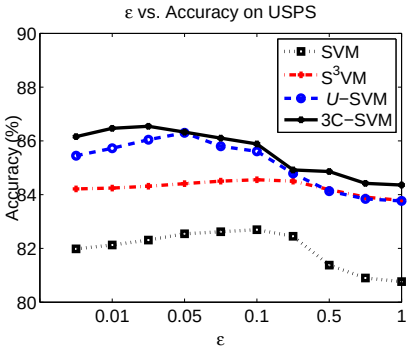
Dataset	**Algorithm**	$\tau = 0.1$	$\tau = 0.5$	$\tau = 0.9$
USPS	SVM	72.4± 15.9 (**0.7**)	72.4± 15.9 (**9.5**)	72.4± 15.9 (53.1)
	S^{3}VM	63.6 ± 8.9 (**0.0**)	68.2 ± 8.0 (**2.2**)	73.2 ± 7.0 (**9.5**)
	$\mathcal{U}$-SVM	83.1 ± 2.5 (**0.0**)	73.4 ± 4.4 (**0.0**)	64.2 ± 3.6 (**0.0**)
	3C-SVM	**87.2**±2.3	**80.6**±4.8	**75.4**±7.3
MNIST	SVM	70.9± 11.4 (**0.3**)	70.9± 11.4 (**0.8**)	70.9± 11.4 (13.6)
	S^{3}VM	70.9 ± 10.5 (**0.7**)	72.4 ± 10.1 (**1.0**)	75.7 ± 9.1 (**9.8**)
	$\mathcal{U}$-SVM	84.2 ± 2.2 (**0.2**)	80.0 ± 4.6 (**0.9**)	75.0 ± 3.9 (**1.0**)
	3C-SVM	**85.3**±1.6	**82.8**±2.9	**77.6**±3.9

The USPS dataset and the MNIST dataset are two popular benchmark handwritten digit datasets used in the literature to validate the proposed models [36, 67, 99, 137]. The USPS

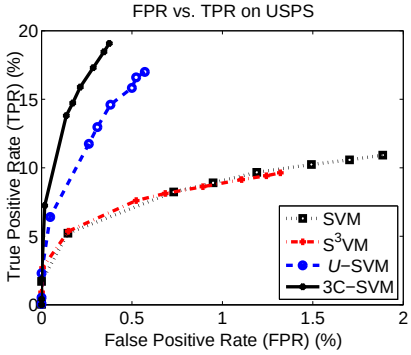
dataset was first scanned by the U.S. Postal Service from envelopes that passed through the Baffalo, NY Post Office [79]. Each image was further normalized and centered with the size of 16×16 (Fig. 5.1(a)). This dataset contains 9,298 grayscale handwritten digit images, 7,291 of which are used as the training set, while the remaining 2,007 are used as the test set. The MNIST dataset consists of a training set of 60,000 digits and a test set of 10,000 digits (Fig. 5.1(b)). The digits are grayscale handwritten images normalized and centered in 28×28 [100]. We have normalized each pixel value in an image to the range of -1 and 1.

We follow the experimental work of [146, 165]: Digits "5" and "8" are constructed as a binary classification problem for ± 1-class data. The labeled, $\mathcal{L}_{\pm 1}$-data with the size being 20 and the $\mathcal{U}_{\mathcal{L}}$-data with the size being τU ($U = 500$) are randomly selected from digits "5" and "8". $(1 - \tau)U$ other digits are randomly chosen to construct the $\mathcal{U}_0$-data.

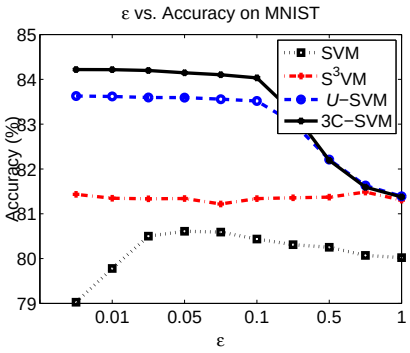
Table 5.2 reports the average (10 runs) accuracies of four algorithms on the two digit datasets. Our 3C-SVM obviously achieves the best results and outperforms $\mathcal{U}$-SVMs and S^3VMs. The 3C-SVM achieves significant improvement only except for $\tau = 0.9$ against SVMs, where the results of SVMs are unstable and with large variances. Here, our 3C-SVM uses the same regularized parameters as those in the $\mathcal{U}$-SVM. That means our 3C-SVM still has room to improve its performance. Under this setting, the results of 3C-SVMs follow the same trend of the results of $\mathcal{U}$-SVMs. That is, as the number of $\mathcal{U}_0$-data decreases, the performance of 3C-SVMs decreases. For S^3VM, as τ increases, i.e., the number of unlabeled data from the target task increases, the performance of S^3VM increases gradually. This observation shows that more unlabeled data coming from the target task will help S^3VM, but when these related data become less, the performance of S^3VM drops clearly.



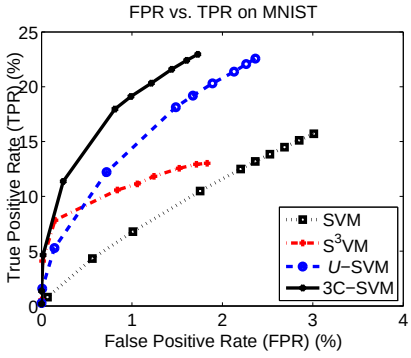
(a) ε vs. Accuracy on USPS



(b) FPR vs. TPR on USPS



(c) ε vs. Accuracy on MNIST



(d) FPR vs. TPR on MNIST

Figure 5.6: Results of accuracies, false positive rates and true positive rates on detecting 0-class from two benchmark handwritten digits datasets. The above results are obtained by varying ε and adopting the definition of $\mathbf{d}$ in (5.5) as the rule to distinguish 0-class data from the best classifiers obtained from previous experiment. The 3C-SVM achieves the best accuracy among all compared classifiers and gains about 20% improvement on the TPR for the USPS dataset and 10% improvement on the TPR for the MNIST dataset compared to the best of other three models at the same FPR, corresponding to the largest FPR of our 3C-SVM.

We further compare the performance of our 3C-SVM with other three models in distinguishing the 0-class data. Here, we prepare test data for a new binary classification task without training classifiers. More specifically, we use the best classifiers built in previous experiment and test them in the new task. The new binary classification task is constructed as follows: digits "5" and "8" from the test set of the benchmark datasets consisting of a new 1-class data while the other digits on the test set consisting of a new 0-class data. Since the other three models have no ability in distinguishing the 0-class data automatically, for fair comparison, we adopt a simple criterion, i.e., classifying the data into corresponding labels based on which error (the ε-insensitive loss or the symmetric hinge loss) incurred is smaller, the same definition as $\mathbf{d}$ in (5.5). We then use four corresponding best classifiers obtained in the above experiment to get the accuracies (although the accuracy here is not related to our target task), false positive rates (FPRs), and true positive rates (TPRs) on the new binary class data by varying ε from $\{0, 0.01, 0.025, 0.05, 0.075, 0.1, 0.25, 0.5, 0.75, 1.0\}$. Results are shown in Figure 5.6. We can see that our 3C-SVM attains the best accuracy versus different ε's and achieves the highest TPR when on the same FPR for all classifiers. From Fig. 5.6(b) and Fig. 5.6(d), we observe that the TPRs are a little low in all models. All models tend to classify the digits "5" and "8" into 0-class data. Relatively, under the same FPR, corresponding to the largest FPR of our 3C-SVM, our 3C-SVM can gains about 20% improvement on the TPR for the USPS dataset and 10% improvement on the TPR for the MNIST dataset compared to the best of other three models. These results again demonstrate the advantage of our 3C-SVM algorithm.

5.6 Summary

In this work, we have proposed a novel maximum margin classifier, named the *tri-class support vector machine*, to solve the binary classification task in a new scenario of semi-supervised learning, where the labeled and the unlabeled data are a mixture of data from the same or different distribution of the target labeled data. We introduce a new min loss function to distinguish the mixed unlabeled data into relevant and irrelevant data based on which error is smaller when assigning the data to the associated class. This min loss function can therefore achieve the maximum entropy principle and make the irrelevant data close to the decision boundary.

The 3C-SVM is a generalization of SVMs, S^3VMs, and $\mathcal{U}$-SVMs and provides a framework to derive these three models. We have further analyzed how the irrelevant data play the role of seeking feature subspace and why the model works from theoretical perspective. The implementation of the 3C-SVM needs to solve several QP problems, which is a highly scalable algorithm in existing algorithms.

Currently, the 3C-SVM is solved by a standard package to handle the QP problem, whose worse case time complexity is $O(N^3)$, where N is the number of the training data points mainly dominating by the number of unlabeled data. How to further reduce the time complexity of the model to quadratic or even sub-linear on the number of training data points is very promising research problem. Another issue is to provide theoretical analysis of generalization error bound on the model. The generalization error bound on related semi-supervised learning model has been provided in the literature. However, the generalization error bound for the $\mathcal{U}$-SVM is still an open problem. In our model, we also have to tackle some unlabeled data with the role similar to the universum data. This makes it specifi-

cally hard for analyzing the generalization error bound of the 3C-SVM.

□ **End of chapter.**

Chapter 6

Efficient Sparse Generalized Multiple Kernel Learning

6.1 Introduction

Kernel methods such as support vector machines (SVMs), kernel principle component analysis, etc. [77, 78, 137, 144] have become useful tools in various applications, e.g., pattern recognition [137, 144], bioinformatics [95, 166]. To achieve good performance, one has to define a good kernel representation. The kernel matrix is specified by the inner product of data points mapped in a high-dimensional (possibly infinite dimensional) feature space. The kernel matrix defines the similarity among data and usually has to be learned from the data.

The problem of learning the optimal kernel matrix received much attention in recent studies of machine learning [137] and research in this field has become quite active in recent years [35, 62, 73, 105, 158]. One of the important kernel learning techniques is multiple kernel learning (MKL), which was first introduced in [96]. In general, multiple kernel learning searches for the linear combination of base kernel functions/matrices that maximizes a generalized performance measure. Typical measures for multiple kernel learning include maximum margin classification errors [9, 96], kernel-target alignment [43], and Fisher

discriminative analysis [182]. MKL methods have been shown to be usually outperformed by SVM with uniformly-weighted kernels [38, 89, 90, 96].

Among various MKL methods, the L_1-MKL has shown its efficiency in learning the kernel weights. This method seeks the kernel weights in a simplex and thus yields a sparse solution. The sparsity of the selected kernels is helpful to identify appropriate combination of data sources or different feature subsets in real-world applications, such as genome fusion [95], splice site detection [150], image annotation [64], etc. However, when a problem contains kernels encoding orthogonal or correlation characterizations, the simplex solution space may discard useful information and thus result in suboptimal generalization performance [90]. Alternatively, an MKL with the L_2-norm constraint on the kernel weights is proposed [89] and an MKL with the L_p-norm ($p > 1$) constraint on the kernel weights is further presented [90] to improve the L_1-MKL method. Unfortunately, these extensions lead to a non-sparse solution and may be sensitive to noise. They suffer poor interpretation ability and subsequently can lead to high computational and storage cost.

To avoid problems of the above two types of approaches, it is strongly desirable to keep the locally orthogonal information in the base kernels [38, 90], while at the same time, to yield a sparse solution. Clearly, one approach toward this objective is first to cluster the kernel matrices/functions into groups and then to identify the leading groups. In this way, the complementary or locally orthogonal information can be kept and sparse solutions can also be obtained. Similar methods, e.g., group lasso [184], fussed lasso [156], etc., have been introduced in Statistics. Group lasso aims to find important explanatory factors in predicting the response variable, where each explanatory factor can be represented by a group of derived input variables. In [8], Bach has shown that group lasso reduces to multiple

kernel learning, when the Euclidean norms in group lasso are replaced by reproducing kernel Hilbert norms. The composite kernel learning [153] is an example of kernel learning approach based on the group lasso, where the kernels are hierarchically penalized. Despite their success, the group composition must be specified ahead as a prior knowledge. However, in some real-world problems, the prior knowledge on the composition of the group structure may not be available before learning. Moreover, the group penalization often involves high computation cost due to the projection to the hierarchical structure of the kernel weights.

To tackle the above problems, we propose a novel generalized multiple kernel learning (GMKL) model. Our model introduces the regularization with a linear combination of the L_1-norm and the squaredL_2-norm on the kernel weights, i.e., a combination of lasso and ridge penalties on the kernel weights. This model generalizes the L_1-MKL and the L_2-MKL methods. More importantly, our GMKL not only enjoys sparse solution as the L_1-MKL, but also encourages the grouping effect on the solution, where similar base kernels tend to be either in or out of the model altogether without specifying the group information in advance. Therefore, this demonstrates distinct advantages over the L_1-MKL [96, 130] or the L_2-MKL [89]. Furthermore, compared with group lasso based approaches, the proposed approach relaxes the needs for the prior knowledge of the group structure of base kernels.

In summary, our contributions of this work include

- A generalized multiple kernel learning model is introduced to advance the research progress in this area. The model generalizes several previously proposed MKL models, including the L_1-MKL and the L_2-MKL, and overcomes the insufficiency of these proposed methods.

- Theoretical analysis of the GMKL on why it contains a sparse solution with the grouping effect is provided. This guarantees the favorite properties of the GMKL.
- The GMKL is transformed into a convex-concave optimization problem. So the global optimal solution is guaranteed. A very efficient method, the level method, is proposed to solve the GMKL and its convergence rate is provided. This solution enables the GMKL for its potential on solving large scale datasets.
- A series of experiments have been conducted both on synthetic and real-world datasets to demonstrate the effectiveness and efficiency of the GMKL.

The rest of the chapter is organized as follows. In Section 6.2, we outline the multiple kernel learning framework and introduce the current research progress on extending this framework. In Section 6.3, we describe our proposed generalized multiple kernel learning model and provide theoretical analysis on the properties of the GMKL. In Section 6.4, we present the solution of the GMKL by the level method and provide its convergence analysis. In Section 6.5, we report the experimental results on both synthetic and real-world datasets. Finally, we conclude the chapter in Section 6.6.

6.2 Multiple Kernel Learning

In this section, we first introduce the basic concept of kernel methods. We then present the framework of multiple kernel learning. Finally, the L_1-MKL and its extensions are further discussed.

6.2.1 Preliminaries

In supervised learning, a set of labeled data $\mathcal{D} = \{(\mathbf{x}_i, y_i)\}_{i=1}^N$ is given, where $\mathbf{x}_i \in \mathcal{X} \subset \mathbb{R}^d$ for some input space $\mathcal{X}$, and $y_i \in \mathcal{Y}$. For binary classification, y_i can be -1 or 1. For regression problems, $y_i \in \mathbb{R}$. The objective of supervised learning is to find a hypothesis $f \in \mathcal{H}$, which can generalize well on new and unseen data. This is attained by minimizing the following regularized risk:

$$f^\star = \arg\min_f \; C\mathrm{R}_{\mathrm{emp}}(f) + \Omega(f), \tag{6.1}$$

where $\mathrm{R}_{\mathrm{emp}}(f) = \frac{1}{N}\sum_{i=1}^N R(f(\mathbf{x}_i), y_i)$ is the empirical risk of hypothesis f with respect to a loss function, $R : \mathbb{R} \times \mathcal{Y} \to \mathbb{R}$, and $\Omega(f)$ is a regularization term. The positive constant term, C, is a trade-off parameter balancing the regularization and the empirical risk.

For different problems, different loss functions $R(f(\mathbf{x}), y)$ are adopted. For example,

- Hinge loss: $R(f(\mathbf{x}), y) = \max\{0, 1 - yf(\mathbf{x})\}$. This loss function is usually applied in binary classification on SVMs [160].
- ε-insensitive loss function: $R(f(\mathbf{x}), y) = \max\{0, |f(\mathbf{x}) - y| - \varepsilon\}$, a loss function has been used in support vector regression [160].

In this chapter, similar to previous kernel methods [137], the regularizer, $\Omega(f)$, is $\frac{1}{2}\|\mathbf{w}\|_2^2$, corresponding to the squaredL_2-norm on the function weights and the function f takes a linear form with parameters $\mathbf{w}$ and b as

$$f_{\mathbf{w},b}(\mathbf{x}) = \mathbf{w}^\top \phi(\mathbf{x}) + b, \;\; \mathbf{w} \in \mathbb{R}^{d(\mathcal{H})}, \;\; b \in \mathbb{R}, \tag{6.2}$$

where $\phi : \mathcal{X} \to \mathcal{H}$ defines a (possible non-linear) feature mapping from the original input space to a Hilbert space $\mathcal{H}$. The

feature mapping is usually implicitly defined by a Mercer kernel computing the inner product in $\mathcal{H}$ as $\mathbf{K}(\mathbf{x}_i, \mathbf{x}_j) = \langle \phi(\mathbf{x}_i), \phi(\mathbf{x}_j) \rangle_{\mathcal{H}}$ [137].

The decision function can then be represented by

$$f(\mathbf{x}) = \sum_{i=1}^{N} \alpha_i^\star \mathbf{K}(\mathbf{x}, \mathbf{x}_i) + b^\star, \tag{6.3}$$

where the optimal parameter $\boldsymbol{\alpha}^\star$ and $b^\star$ are obtained by solving the dual of the optimization in (6.1).

6.2.2 Multiple Kernel Learning Framework

In the MKL framework, there are given Q base kernels. Each base kernel, $\mathbf{K}_q$, implicitly represents a feature mapping, $\phi_q : \mathcal{X} \to \mathcal{H}_q$, in a reproducing kernel Hilbert space (RKHS) $\mathcal{H}_q$, for $q = 1, \ldots, Q$. The hypothesis in (6.2) is then extended to

$$f_{\hat{\mathbf{w}},b,\boldsymbol{\theta}}(\mathbf{x}) = \hat{\mathbf{w}}^\top \boldsymbol{\phi}_{\boldsymbol{\theta}}(\mathbf{x}) + b = \sum_{q=1}^{Q} \sqrt{\theta_q} \mathbf{w}_q^\top \phi_q(\mathbf{x}) + b, \tag{6.4}$$

where the weight $\hat{\mathbf{w}}$ is defined as $\hat{\mathbf{w}} = (\mathbf{w}_1^\top, \ldots, \mathbf{w}_Q^\top)^\top$, consisting of a $\sum_{q=1}^{Q} d(\mathcal{H}_q)$-dimensional vector. The composite feature mapping is defined as $\boldsymbol{\phi}_{\boldsymbol{\theta}} = \sqrt{\theta_1}\phi_1 \times \ldots \times \sqrt{\theta_q}\phi_Q$, and θ_q is the corresponding coefficient, or the kernel weights of the kernel $\mathbf{K}_q$ and needs to be learned from the data.

The objective of MKL is to seek the optimal kernel combination, $\mathbf{K}_{\boldsymbol{\theta}} = \sum_{q=1}^{Q} \theta_q \mathbf{K}_q$, by minimizing the following optimization while imposing the Ivanov regularization on the kernel weights [90, 96]

$$\min_{\hat{\mathbf{w}},b,\boldsymbol{\theta} \geq \mathbf{0}} \quad C \sum_{i=1}^{N} R(f_{\hat{\mathbf{w}},b,\boldsymbol{\theta}}(\mathbf{x}_i), y_i) + \frac{1}{2} \hat{\mathbf{w}}^\top \hat{\mathbf{w}}, \tag{6.5}$$

$$\text{s.t.} \quad \mathcal{J}(\boldsymbol{\theta}) \leq 1, \tag{6.6}$$

where $\mathcal{J}(\boldsymbol{\theta})$ defines a regularizer on $\boldsymbol{\theta}$, which will be elaborated in the following subsections. It is noted that an MKL framework, which seeks optimal kernels in a compact set by minimizing a regularized functional, was also studied in [113] from a theoretical perspective. The paper [113] mainly studied the theoretical properties on the square loss with L_1-norm regularization on the functional. This is different from what we will propose in the next section.

In addition, we should note that the non-convexity of (6.5) can be resolved by applying the variable transformation, $\mathbf{v}_q := \sqrt{\theta_q}\mathbf{w}_q$, as that in [90, 198]. Hence, the objective in (6.5) becomes

$$\min_{\hat{\mathbf{v}},b,\boldsymbol{\theta}\geq\mathbf{0}} \quad C\sum_{i=1}^{N} R(f_{\hat{\mathbf{v}},b}(\mathbf{x}_i), y_i) + \frac{1}{2}\sum_{q=1}^{Q}\frac{\mathbf{v}_q^\top \mathbf{v}_q}{\theta_q}, \tag{6.7}$$

where $\hat{\mathbf{v}} = (\mathbf{v}_1^\top, \ldots, \mathbf{v}_Q^\top)^\top$ and $f_{\hat{\mathbf{v}},b}(\mathbf{x}) = \sum_{q=1}^{Q}\mathbf{v}_q^\top\phi_q(\mathbf{x}) + b$. In (6.7), we use the convention that $\frac{u}{0} = 0$ if $u = 0$ and ∞ otherwise. If R is a convex function and the constraint (6.6) is convex, then (6.7) is convex. This result can be referred to [22, Ch. 3.1.5].

6.2.3 L_1-MKL

Common approaches in multiple kernel learning [9, 96, 130, 151] impose the L_1-norm constraint on the kernel weights for the kernel selection. That is, $\mathcal{J}(\boldsymbol{\theta}) = \|\boldsymbol{\theta}\|_1$, or $\|\boldsymbol{\theta}\|_1 \leq 1$ in the condition of (6.6). We refer to this case as the L_1-MKL.

In [96], the L_1-MKL is first formulated into a semi-definite programming problem. Due to its effectiveness in learning an interpretable kernel representation, researchers have proposed various methods to speed up its computation. Methods such as second order cone programming [9], semi-infinite linear programming [151], gradient descent [130], and the extended level

method [170], have been proposed to reduce the time consumption in seeking the optimal kernel combination weights.

An advantage of the L_1-MKL constraint on the kernel combination weights is that it provides the favorite property of sparsity, where the obtained kernels can be easily interpreted. However, it may also discard some useful information when two kernels are orthogonal [38] or yield non-unique solutions when two kernels are strongly correlated.

6.2.4 MKL Extensions

In order to tackle the deficiency of the L_1-MKL, researchers have extended the MKL models. They include:

- The MKL model with the L_2-norm constraint on the kernel weights [89], i.e., $\mathcal{J}(\boldsymbol{\theta}) = \|\boldsymbol{\theta}\|_2^2$, or $\|\boldsymbol{\theta}\|_2^2 \leq 1$ for the condition (6.6). Similarly, a multiple kernel ridge regression is proposed in [38], where the kernel weights are constrained in a ball around a positive mean.

- The MKL with the L_p-norm ($p > 1$) constraint on the kernel weights [90, 171]. This corresponds to $\mathcal{J}(\boldsymbol{\theta}) = \|\boldsymbol{\theta}\|_p^p$, or $\|\boldsymbol{\theta}\|_p^p \leq 1$ in (6.6). The L_p-MKL is more general and includes the L_2-MKL as its special case. An interleaved optimization strategy with second order approximation is proposed to solved the L_p-MKL [90].

- The MKL model with mixed norm regularization on the kernel weights [94]. This model imposes a mixed norm regularization on the kernel weights, which yields structure sparsity on the solutions.

- Other MKL extensions: These models reformulate the MKL problem by imposing mixed norm regularization on the function weights [81], or by introducing the elastic net-type

regularization, i.e., a linear combination of the lasso penalty and the ridge penalty on the function weights [157]. These formulations correspond to modifying the regularizer to a block norm, i.e., a norm of the vector containing the individual kernel norms [9, 91]. Now, we discuss several MKL methods incorporating the elastic net-type regularization that may be similar to our work in this chapter. Longworth and Gales [109] included the squaredL_2-norm regularization on the kernel weights while keeping the L_1-norm simplex constraint on the kernel weights. Shawe-Taylor [143] proposed a linear combination of the squaredsum of L_1-norms and the squared L_2-norm on the function weights to solve the novelty detection problem. In [157], an MKL model was proposed to add the linear combination of the lasso penalty and the ridge penalty on the function weights, which includes the L_1-MKL and the uniformly-weighted MKL as its special cases. However, these models lack the analysis on the properties of the models, e.g., the grouping effect.

Among the above methods, the L_1-MKL yields a sparse solution, but cannot capture the complementary information on the kernels. For the L_p-MKL ($p > 1$) models, they will yield non-sparse solutions. As indicated in [200], in the L_p ($p \geq 1$) penalty family, only the lasso penalty ($p = 1$) can produce sparse solutions. The non-sparsity of the solution has the weaknesses in interpreting the model and may be sensitive to noise. In the following section, we will present our proposed generalized multiple kernel learning model to tackle the above insufficiency problem of previously proposed MKL models.

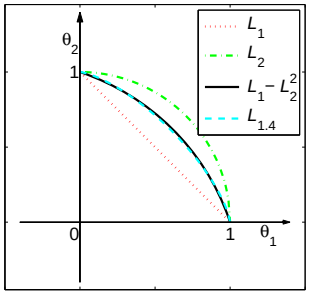


Figure 6.1: Demonstration of a linear combination ($v = 0.5$) of the L_1-norm and the squaredL_2-norm on $\boldsymbol{\theta}$ in R^2 space. Although, L_p-norm ($p = 1.4$) can obtain a similar curve to that of the linear combination of the L_1-norm and the squaredL_2-norm, it produces non-sparse solutions.

6.3 Elastic Net-type Regularization on Multiple Kernel Learning

In this section, we first introduce the formulation of the generalized multiple kernel learning (GMKL) model. We then provide theoretical analysis on the properties of the GMKL model, i.e., explaining why it can produce sparse solutions while encouraging the grouping effect.

6.3.1 Formulation and Duality

Motivated by the fact that the L_1-MKL produces sparse solutions and the L_p-MKL ($p > 1$) can capture correlations among kernels, we propose a generalized multiple kernel learning model incorporating a linear combination norm on the kernel weights as follows:

$$\min_{\boldsymbol{\theta}\in\Theta,\hat{\mathbf{v}},b} C\sum_{i=1}^{N} R(f_{\hat{\mathbf{v}},b}(\mathbf{x}_i), y_i) + \frac{1}{2}\sum_{q=1}^{Q}\frac{\mathbf{v}_q^\top \mathbf{v}_q}{\theta_q}. \tag{6.8}$$

More specially, we set $p = 2$, and the domain of $\boldsymbol{\theta}$ is

$$\Theta = \{\boldsymbol{\theta} \in \mathbb{R}_+^Q \,:\, v\|\boldsymbol{\theta}\|_1 + (1-v)\|\boldsymbol{\theta}\|_2^2 \leq 1\}\,, \tag{6.9}$$

where the parameter v, $0 \leq v \leq 1$, is a non-negative constant to balance the two terms in the constraint. For this MKL extension, we have several remarks:

- There are two main reasons why we adopt this elastic net-type regularization, i.e., a linear combination of the L_1-norm and the squared L_2-norm on the kernel weights. One is due to computational consideration. Through this setting, the optimization in (6.8) is a convex optimization problem given that the loss R is convex. More specifically, it is a quadratically-constrained quadratic programming (QCQP) problem when the hinge loss or the ε-insensitive loss is used. The second reason is that as discussed in Section 6.3.2, our GMKL enjoys the sparsity property as the L_1-MKL and encourages the grouping effect on the kernel weights similar to that of the elastic net on the model weights.

- Our formulation generalizes previously proposed L_1-MKL and the L_2-MKL models. When $v = 0$, the constraint reduces to a ridge penalty on $\boldsymbol{\theta}$ and the model is equivalent to the L_2-MKL [89]. When $v = 1$, the constraint is a lasso constraint and the model is the L_1-MKL [96]. This motivates us to name our model as GMKL. When $v \in (0, 1)$, the constraint contains the characteristics of both the lasso and ridge penalty and the model includes several favorite properties, which will be introduced in Section 6.3.2. Figure 6.1 illustrates the change of the combined L_1-norm and the squaredL_2-norm on $\boldsymbol{\theta}$ in a two-dimensional space.

- The constraint can be further extended by combining the L_1-norm and the L_p-norm ($p > 1$) on the kernel weights

and therefore generalizes previously proposed related MKL methods [90, 171]. When $v \in (0, 1)$, the extended constraint is strictly convex on $\boldsymbol{\theta}$ and contains similar properties of our GMKL formulation; see Section 6.3.2 for more details.

- Our proposed GMKL also generalizes the L_2-norm regularization proposed in [38]. A main difference is that the L_2-norm regularization in [38] introduces an L_2-ball with a predefined positive ball center. Actually, predefining a ball center is not necessary in practical applications and is not required in our model. Furthermore, the formulation in [38] lacks the properties of sparsity and the grouping effect.

Now, we derive the corresponding dual form of the optimization in (6.8) with respect to $\hat{\mathbf{w}}, b$ by fixing $\boldsymbol{\theta}$. Here, we consider the classification problem where the hinge loss is adopted. Hence, the primal problem of GMKL is equivalent to

$$\begin{aligned} \min_{\boldsymbol{\theta}\in\Theta} \ \min_{\hat{\mathbf{v}},b,\boldsymbol{\xi}} \ & C\sum_{i=1}^{N}\xi_i + \frac{1}{2}\sum_{q=1}^{Q}\frac{\mathbf{v}_q^\top \mathbf{v}_q}{\theta_q} \\ \text{s.t.} \quad & y_i\left(\sum_{q=1}^{Q}\mathbf{v}_q^\top \phi_q(\mathbf{x}_i) + b\right) \geq 1 - \xi_i \\ & \xi_i \geq 0, \quad i = 1, \ldots, N. \end{aligned}$$

Following the standard Lagrange multipliers method [152, 160], we construct the corresponding Lagrangian functional, $\mathcal{L}(\hat{\mathbf{v}}, b, \boldsymbol{\xi}, \boldsymbol{\alpha}, \boldsymbol{\gamma})$, of the minimization on the primal variables with fixed $\boldsymbol{\theta}$ as

$$\begin{aligned} \mathcal{L}(\cdot) \ = \ & C\sum_{i=1}^{N}\xi_i + \frac{1}{2}\sum_{q=1}^{Q}\frac{\mathbf{v}_q^\top \mathbf{v}_q}{\theta_q} - \sum_{i=1}^{N}\gamma_i\xi_i \\ & - \sum_{i=1}^{N}\alpha_i\left(y_i\left(\sum_{q=1}^{Q}\mathbf{v}_q^\top \phi_q(\mathbf{x}_i) + b\right) - 1 + \xi_i\right), \end{aligned}$$

where the multipliers satisfy $\boldsymbol{\alpha} \geq \mathbf{0}$ and $\boldsymbol{\gamma} \geq \mathbf{0}$.

Taking the partial derivative of the Lagrangian function with respect to the corresponding primal variables and setting them to zeros, we obtain

$$\frac{\partial \mathcal{L}}{\partial \mathbf{v}_q} = \mathbf{v}_q - \theta_q \sum_{i=1}^{N} \alpha_i y_i \phi_q(\mathbf{x}_i) = 0,\ q = 1, \ldots, Q, \tag{6.10}$$

$$\frac{\partial \mathcal{L}}{\partial b} = \sum_{i=1}^{N} \alpha_i y_i = 0, \tag{6.11}$$

$$\frac{\partial \mathcal{L}}{\partial \xi_i} = C - \alpha_i - \gamma_i = 0,\ i = 1, \ldots, N. \tag{6.12}$$

From (6.10), we can obtain the dual form of (6.8) as follows:

$$\min_{\boldsymbol{\theta} \in \Theta} \max_{\boldsymbol{\alpha} \in \mathcal{A}} \ \mathcal{D}(\boldsymbol{\theta}, \boldsymbol{\alpha}), \tag{6.13}$$

where the objective function is defined as

$$\mathcal{D}(\boldsymbol{\theta}, \boldsymbol{\alpha}) = \mathbf{1}_N^\top \boldsymbol{\alpha} - \frac{1}{2} (\boldsymbol{\alpha} \circ \mathbf{y})^\top \left(\sum_{q=1}^{Q} \theta_q \mathbf{K}_q \right) (\boldsymbol{\alpha} \circ \mathbf{y}). \tag{6.14}$$

Correspondingly, constraints (6.11) and (6.12) with the conditions of $\boldsymbol{\alpha} \geq \mathbf{0}$ and $\boldsymbol{\gamma} \geq \mathbf{0}$ yield the domain of $\boldsymbol{\alpha}$ defined in the set of $\mathcal{A}$ as

$$\mathcal{A} = \{ \boldsymbol{\alpha} \in \mathbb{R}_+^N,\ \boldsymbol{\alpha}^\top \mathbf{y} = 0,\ \boldsymbol{\alpha} \leq C \mathbf{1}_N \}. \tag{6.15}$$

The formulation in (6.13) is a convex-concave problem and its optimal solution is guaranteed to be the global optimal solution. Wrapping-based methods [130, 151] have been proposed to solve this kind of optimization problems. Especially, the maximization problem in (6.13) corresponds to a standard dual form of SVMs. Currently, solvers for SVMs are very efficient [21, 32] and can be directly adopted in our model.

6.3.2 Properties

Here, we present several properties for our GMKL model. First, we prove that the constraint on $\boldsymbol{\theta}$ is tight when the optimal solution is obtained. Second, we provide a theorem to show that the solution of the GMKL is sparse with the grouping effect. Third, we prove that our GMKL model introduces the grouping effect when two kernels are strongly correlated.

To simplify the analysis, we first define the optimal $(\boldsymbol{\theta}^\star, \boldsymbol{\alpha}^\star)$ as follows:

Definition 1. $(\boldsymbol{\theta}^\star, \boldsymbol{\alpha}^\star)$ *is the optimal solution of (6.13). That is,*

$$\boldsymbol{\theta}^\star = \arg\min_{\boldsymbol{\theta}\in\Theta} \ \mathcal{D}(\boldsymbol{\theta}, \boldsymbol{\alpha}^\star) \quad \boldsymbol{\alpha}^\star = \arg\max_{\boldsymbol{\alpha}\in\mathcal{A}} \ \mathcal{D}(\boldsymbol{\theta}^\star, \boldsymbol{\alpha}).$$

Now, the following theorem shows that the optimal solution in (6.13) is attained when the constraint of $\boldsymbol{\theta}$ is tight.

Theorem 8. *Suppose the kernel matrices,* $\mathbf{K}_1, \ldots, \mathbf{K}_Q$ *are positive semi-definite. Then the condition* $v\|\boldsymbol{\theta}^\star\|_1 + (1-v)\|\boldsymbol{\theta}^\star\|_2^2 = 1$ *always holds.*

Proof. First, we have the following two observations about the function $\mathcal{D}(\boldsymbol{\theta}, \boldsymbol{\alpha}^\star)$:

1. The function $\mathcal{D}(\theta_q, \boldsymbol{\alpha}^\star)$ is a monotonically, but not strictly, decreasing function on each element of $\boldsymbol{\theta}$, or θ_q, with fixed $\boldsymbol{\alpha}^\star$. It is because that $\mathcal{D}(\theta_q, \boldsymbol{\alpha}^\star)$ is a linear function on θ_q with each coefficient being non-positive. That is, $\mathcal{D}(\theta_q, \boldsymbol{\alpha}^\star) = u_q\theta_q$, where the q-th coefficient on θ_q is $u_q = -\frac{1}{2}(\boldsymbol{\alpha}^\star \circ \mathbf{y})^\top \mathbf{K}_q(\boldsymbol{\alpha}^\star \circ \mathbf{y})$. Obviously, u_q is non-positive since the kernel matrix $\mathbf{K}_q$ is positive semi-definite.

2. The constraint function, $v\|\boldsymbol{\theta}\|_1 + (1-v)\|\boldsymbol{\theta}\|_2^2$, is element-wise and it is an increasing function on each element of $\boldsymbol{\theta}$, or θ_q, with $\theta_q \geq 0$, for $q = 1, \ldots, Q$.

Hence, we can conclude that the optimal $\boldsymbol{\theta}^\star$ should be attained when the constraint of (6.9) is tight. Otherwise, we have the following two cases:

1. If there is an element, e.g., q with $u_q < 0$, we can select θ_q and increase its value to make (6.9) tight. This again will further reduce the function value of $\mathcal{D}(\boldsymbol{\theta}, \boldsymbol{\alpha}^\star)$, which is a better solution of (6.13), from the above first observation.

2. For all q, $u_q = 0$, we can select any θ_q and increase its value to make (6.9) tight, while keeping the same optimal objective function value.

□

We now turn to study the grouping effect of our GMKL. First, we note that the grouping effect is only derived from $\boldsymbol{\theta}$ and it is not related to the variable $\boldsymbol{\alpha}$. By the Lagrange multiplier method [22], we know that (6.13) is equivalent to the following minimization problem given the fixed $\boldsymbol{\alpha}^\star$ for some $\lambda \geq 0$:

$$\min_{\boldsymbol{\theta} \geq \mathbf{0}} \mathcal{D}(\boldsymbol{\theta}, \boldsymbol{\alpha}^\star) + \lambda \left(v\|\boldsymbol{\theta}\|_1 + (1-v)\|\boldsymbol{\theta}\|_2^2 \right). \tag{6.16}$$

We then have the following theorem stating one aspect of the grouping effect.

Theorem 9. *Suppose $\lambda > 0$, $\mathbf{K}_i = \mathbf{K}_j$, $i, j \in \{1, \ldots, Q\}$, and $\boldsymbol{\theta}^\star$ is a minimizer of (6.16), we have*

1. *If $v \neq 1$, then*

$$\theta_q^\star = \max \left\{ 0, \frac{1}{2(1-v)} \left(\frac{1}{2\lambda} (\boldsymbol{\alpha}^\star \circ \mathbf{y})^\top \mathbf{K}_q (\boldsymbol{\alpha}^\star \circ \mathbf{y}) - v \right) \right\}, \tag{6.17}$$

and therefore $\theta_i^\star = \theta_j^\star$.

2. *If $v = 1$, then $\tilde{\boldsymbol{\theta}}$ is another minimizer of (6.16) with*

$$\tilde{\theta}_q = \begin{cases} \theta_q^\star & \text{if } q \neq i \text{ and } q \neq j \\ (\theta_i^\star + \theta_j^\star) \cdot \sigma & \text{if } q = i \\ (\theta_i^\star + \theta_j^\star) \cdot (1 - \sigma) & \text{if } q = j \end{cases}$$

for any $\sigma \in [0, 1]$.

A detailed proof is in Appendix D.1. There are some remarks about the above theorem:

- Theorem 9 provides an explicit solution of $\boldsymbol{\theta}$ in (6.17). This is different from that of the elastic net in [200, Lemma 2].
- Theorem 9 indicates that our GMKL can achieve the grouping effect and the L_1-MKL does not have a unique solution when two kernels are the same. This analysis can be also extended to other regularizers with strictly convex property.
- Eqation (6.17) also indicates that our GMKL can yield sparse solutions when the second term in the bracket of (6.17) is less than 0. On the otherhand, by setting $v = 0$ into (6.17), we can obtain, $\theta_q^\star = \frac{1}{2(1-v)} \left(\frac{1}{2\lambda} (\boldsymbol{\alpha}^\star \circ \mathbf{y})^\top \mathbf{K}_q (\boldsymbol{\alpha}^\star \circ \mathbf{y}) \right)$ for the L_2-MKL. This also shows that the L_2-MKL yields the grouping effect, but usually yields non-sparse solutions on the kernel weights.

We further analyze the grouping effect when the given kernels are strongly correlated. Here, we define a ratio for two kernels to indicate the correlation of two kernels:

Definition 2. *Let r_{ij} define the ratio of two kernels on given $\boldsymbol{\alpha}^\star$ as*

$$r_{ij} = \frac{(\boldsymbol{\alpha}^\star \circ \mathbf{y})^\top \mathbf{K}_i (\boldsymbol{\alpha}^\star \circ \mathbf{y})}{(\boldsymbol{\alpha}^\star \circ \mathbf{y})^\top \mathbf{K}_j (\boldsymbol{\alpha}^\star \circ \mathbf{y})}.$$

If $r_{ij} \approx 1$, we say $\mathbf{K}_i$ and $\mathbf{K}_j$ are strongly correlated. Now, we can easily obtain the following theorem:

Theorem 10. *Given two kernels $\mathbf{K}_i$ and $\mathbf{K}_j$, if $v \neq 1$, as r_{ij} approaches 1, we have $\theta_i^\star$ approaches $\theta_j^\star$.*

Proof. Since $v \neq 1$, from (6.17) in Theorem 9, we note that $\theta_q^\star$ can be simplified as $\max\{0, t_q\}$, a continuous function of t_q, where $t_q = \frac{1}{2(1-v)}\left(\frac{1}{2\lambda}(\boldsymbol{\alpha}^\star \circ \mathbf{y})^\top \mathbf{K}_q(\boldsymbol{\alpha}^\star \circ \mathbf{y}) - v\right)$. Hence, we have

$$|\theta_i^\star - \theta_j^\star| \leq |t_i - t_j|, \text{ for } i, j = 1, \ldots, Q. \tag{6.18}$$

This inequality can be obtained by analyzing the following several cases.

1. When $\theta_i^\star$ and $\theta_j^\star$ are both positive, we have $t_i, t_j > 0$ and attain the equality in (6.18).
2. When $\theta_i^\star > 0$ and $\theta_j^\star = 0$, we have $|\theta_i^\star - \theta_j^\star| = |t_i| \leq |t_i - t_j|$. The inequality is due to the condition that $t_j \leq 0$. For the case of $\theta_i^\star = 0$ and $\theta_j^\star > 0$, we can derive the result similarly.
3. When $\theta_i^\star = \theta_j^\star = 0$, the inequality in (6.18) is satisfied for all t_i and t_j.

From (6.18), we have

$$|\theta_i^\star - \theta_j^\star| \leq \frac{1}{2(1-v)}\frac{1}{2\lambda}(\boldsymbol{\alpha}^\star \circ \mathbf{y})^\top \mathbf{K}_j(\boldsymbol{\alpha}^\star \circ \mathbf{y})|r_{ij} - 1|. \tag{6.19}$$

Hence, as $r_{ij} \approx 1$, we have $|\theta_i^\star - \theta_j^\star| \approx 0$. That is, $\theta_i^\star$ approaches $\theta_j^\star$. □

From (6.19), we note that the difference of two weights, $|\theta_i^\star - \theta_j^\star|$, is proportional to the ratio value, $|r_{ij} - 1|$, and inversely proportional to $2(1 - v)$. The ratio value indicates that if two kernels are strongly correlated, the weights are nearly the same. The inversely proportional value $2(1-v)$ indicates that a smaller

v will yield closer solutions for the kernel weights. Meanwhile, we should note that the coefficient, 2, is introduced due to the use of the L_2-norm on the kernel weights. If the L_p-norm is adopted, the ratio is inversely related to p. As p increases, e.g., approaches to infinity, it will lead to the same weights, i.e., uniformly-weighted MKL, which is the same as the result in the previous MKL models.

In summary, our GMKL contains the following properties:

- In view of (6.17), we can see that out GMKL imposes the sparsity on the coefficients of the model. This surpasses those non-sparse MKL models [89, 90], which may be prone to noise and have a larger computation/storage cost.
- Theorem 9 and Theorem 10 state that the GMKL can provide the grouping effect, which retains more useful information from the data than the L_1-MKL.
- Non-linearity is embedded in the formulation of (6.8) and is represented by the kernels. Our GMKL can therefore capture more information of the data than other statistic models, e.g., the lasso [155] and the elastic net [200].

Table 6.1 summarizes the above arguments.

Table 6.1: Comparison between GMKL and the other models.

	L_1-MKL	L_2-MKL	GMKL	Lasso	Elastic net	Group Lasso
Sparsity	✓	×	✓	✓	✓	✓
Non-linearity	✓	✓	✓	×	×	×
Grouping	×	✓	✓	×	✓	×

6.4 Optimizing the GMKL

Due to the efficiency in solving SVMs, the wrapping-based methods have been adopted to solve the MKL models, e.g., [90, 105, 130, 151, 162, 170]. In the wrapping-based methods, the first step is to seek the optimal $\hat{\mathbf{w}}, b$ or the dual variable $\boldsymbol{\alpha}$ given a fixed $\boldsymbol{\theta}$ by an SVM solver. The second step is to update the kernel weights $\boldsymbol{\theta}$ to further decrease the objective value of (6.5) with fixed primal variables $\hat{\mathbf{w}}$ and b or fixed dual variable $\boldsymbol{\alpha}$. Many previously proposed MKL methods try to speed up the model in the second step. For example, a gradient method is proposed in [130, 162]; an SILP method is applied in [90, 151]; and the level method is introduced in [170].

Among these optimization methods, the level method, a cutting plane method derived from the family of bundle methods [117], has shown better success in solving machine learning and kernel learning methods. For example, it has been introduced to efficiently solve regularized risk minimization problems [149], the L_1-MKL [170], and neighborhood kernel learning [105]. Hence, in this chapter, we adopt the level method to solve our GMKL of (6.13).

6.4.1 GMKL by the Level Method

The key part of the level method is to construct the corresponding lower bound and upper bound of the objective function. First, we know that $\mathcal{D}(\boldsymbol{\theta}, \boldsymbol{\alpha})$ in (6.13) is convex on $\boldsymbol{\theta}$ and concave on $\boldsymbol{\alpha}$. According to von Neumann Lemma [163], for any optimal solution $(\boldsymbol{\theta}^\star, \boldsymbol{\alpha}^\star)$, we have

$$\begin{aligned} \mathcal{D}(\boldsymbol{\theta}^\star, \boldsymbol{\alpha}) \leq \max_{\boldsymbol{\alpha} \in \mathcal{A}} \mathcal{D}(\boldsymbol{\theta}^\star, \boldsymbol{\alpha}) &= \mathcal{D}(\boldsymbol{\theta}^\star, \boldsymbol{\alpha}^\star) \\ &= \min_{\boldsymbol{\theta} \in \Theta} \mathcal{D}(\boldsymbol{\theta}, \boldsymbol{\alpha}^\star) \leq \mathcal{D}(\boldsymbol{\theta}, \boldsymbol{\alpha}^\star). \quad (6.20) \end{aligned}$$

The above property indicates that our model can easily obtain the corresponding lower bound and the upper bound.

Suppose $\{(\boldsymbol{\theta}^i, \boldsymbol{\alpha}^i)\}_{i=1}^t$ denote the solutions of (6.13) obtained in the last t iterations. We define the corresponding lower bound, $\underline{\mathcal{D}}^t$, and the corresponding upper bound, $\overline{\mathcal{D}}^t$, as follows:

$$\underline{\mathcal{D}}^t = \min_{\boldsymbol{\theta}\in\Theta} h^t(\boldsymbol{\theta}), \quad \overline{\mathcal{D}}^t = \min_{1\le i\le t} \mathcal{D}(\boldsymbol{\theta}^i, \boldsymbol{\alpha}^i)\,, \tag{6.21}$$

where $h^t(\boldsymbol{\theta})$ corresponds to a cutting plane as follows:

$$h^t(\boldsymbol{\theta}) = \max_{1\le i\le t} \mathcal{D}(\boldsymbol{\theta}, \boldsymbol{\alpha}^i). \tag{6.22}$$

It is noted that the lower bound is the minimum value at the cutting plane and the upper bound is the minimum objective value attained at previous steps.

We can then define the level set as follows

$$\mathcal{L}^t = \{\boldsymbol{\theta} \in \Theta : h^t(\boldsymbol{\theta}) \le \mathcal{V}^t = \tau\overline{\mathcal{D}}^t + (1-\tau)\underline{\mathcal{D}}^t = \underline{\mathcal{D}}^t + \tau\Delta^t\}\,, \tag{6.23}$$

where $\tau \in (0,1)$ is a given constant controlling the tradeoff of the two bounds. The level set specifies the set of solution where the objective is bounded by the lower bound and the upper bound. The gap, Δ^t, between the upper bound and the lower bound at each step is defined as

$$\Delta^t = \overline{\mathcal{D}}^t - \underline{\mathcal{D}}^t\,, \tag{6.24}$$

and measures the sub-optimality for the solution $(\boldsymbol{\theta}^t, \boldsymbol{\alpha}^t)$ at each step.

The final step in the level method is to project $\boldsymbol{\theta}^t$ onto the level set $\mathcal{L}^t$ to calculate a new solution, $\boldsymbol{\theta}^{t+1}$. That is, we obtain $\boldsymbol{\theta}^{t+1}$ by solving the following quadratic optimization problem:

$$\begin{aligned} \min_{\boldsymbol{\theta}\in\Theta} \quad & \|\boldsymbol{\theta} - \boldsymbol{\theta}^t\|_2^2 \\ \text{s.t.} \quad & \mathcal{D}(\boldsymbol{\theta}, \boldsymbol{\alpha}^i) \le \mathcal{V}^t, \quad i = 1, \ldots, t. \end{aligned} \tag{6.25}$$

The intuition of the projection is to make the solution satisfy the level set conditions in a faster way and to require $\boldsymbol{\theta}$'s in two consecutive steps close to each other, avoiding oscillation on the solution.

The following pseudo code presents how to solve the GMKL by the level method.

Algorithm 4 The Level Method for the Generalized Multiple Kernel Learning

Given: predefined tolerant error $\delta > 0$

Initialization: Let $t = 0$ and $\boldsymbol{\theta}^0 = c\mathbf{1}_Q$, where c is the positive root of the quadratic equation: $(1-v)c^2 + vc - \frac{1}{Q} = 0$;

repeat

1. Solve the dual problem of the SVM with $\sum_{q=1}^{Q} \theta_q^t \mathbf{K}_q$ to get the optimal solution, $\boldsymbol{\alpha}$;
2. Construct the cutting plane model, $h^t(\boldsymbol{\theta})$, in (6.22);
3. Calculate the lower bound $\underline{\mathcal{D}}^t$ and the upper bound $\overline{\mathcal{D}}^t$ in (6.21), and the gap Δ^t in (6.21);
4. Compute the projection of $\boldsymbol{\theta}^t$ onto the level set $\mathcal{L}^t$ by solving the optimization problem in (6.25);
5. Update $t = t + 1$;

until $\Delta^t \leq \delta$.

Remarks: Two points about Algorithm 4 need to be emphasized.

- Initialization of $\boldsymbol{\theta}$: We set $\boldsymbol{\theta}^0$ uniformly at each element, i.e., $\boldsymbol{\theta}^0 = c\mathbf{1}_Q$, where $c > 0$. From Theorem 8, we must have $v \cdot Q \cdot c + (1-v) \cdot Q \cdot c^2 = 1$. This requires to seek the positive root of the quadratic equation as that in Algorithm 4.
- In terms of computation, the main part of our GMKL is that we have introduced a quadratic constraint in (6.9). This may require a bit more computation when compared with the L_1-MKL approach. In Algorithm 4, there are two steps involving the quadratic constraint. They are: the step 2) in Algorithm 4, which constructs the cutting plane in (6.22) by solving a linear program with a quadratic con-

straint, and the step 4) in Algorithm 4, which projects $\boldsymbol{\theta}$ to the level set by solving a quadratic programming with a quadratic constraint in (6.25). We believe some warm start methods, e.g., solving the corresponding problems with previously obtained optimal value [92], can be adopted to speed up the seeking of the next optimal value.

6.4.2 Convergence Analysis

Algorithm 4 is terminated when the gap between the two bounds is small. To analyze the convergence of the level method on our GMKL model, we first have the following theorem to state that in each iteration, the gap is non-increasing and the difference between the optimal objective value and the attained objective value is bounded by the gap.

Theorem 11. *We have the following properties on the gap, Δ^i, $i = 1, \ldots, t$:*

1. $\Delta^i \geq 0$,
2. $\Delta^1 \geq \Delta^2 \geq \ldots \geq \Delta^t$,
3. $|\mathcal{D}(\boldsymbol{\theta}^i, \boldsymbol{\alpha}^i) - \mathcal{D}(\boldsymbol{\theta}^\star, \boldsymbol{\alpha}^\star)| \leq \Delta^t$.

We then have the following theorem, which provides the convergence rate of Algorithm 4.

Theorem 12. *For any $\delta > 0$, Algorithm 4 converges to the desired precision after T steps, and*

$$T \geq \frac{2c(\tau)V^2}{\delta^2}, \tag{6.26}$$

where $c(\tau) = \frac{1}{(1-\tau)^2\tau(2-\tau)}$, V is calculated by $\frac{1}{2}NC^2\sqrt{Q}\max_{1\leq q\leq Q}\Lambda_{\max}(\mathbf{K}_q)$, and $\Lambda_{\max}(\mathbf{K}_q)$ defines the maximum eigenvalue of matrix $\mathbf{K}_q$.

We put the proof of the Theorem 11 and the Theorem 12 in Appendix D.2 and Appendix D.3, respectively. It is noted that the convergence rate of the level method is $O(\delta^{-2})$. According to [117], empirically, a better convergence rate $O(N \log(\frac{1}{\delta}))$ can be observed.

Table 6.2: Summary of the synthetic and UCI datasets.

Type	Dataset	# Training (N)	# Test	# Dim (d)	# Kernel (Q)
Synthetic	Toy 1	150	150	20	273
	Toy 2	150	150	20	273
UCI	Breast	341	342	10	143
	Heart	135	135	13	182
	Ionosphere	175	176	33	442
	Liver	172	173	6	91
	Pima	384	384	8	117
	Sonar	104	104	60	793
	Wdbc	284	285	30	403
	Wpbc	99	99	33	442

Table 6.3: Summary of the proteins subcellular localization datasets.

Dataset	# Classes	# Training (N)	# Test	# Kernel (Q)
Plant	4	470	470	69
Psort+	4	270	271	69
Psort–	5	722	722	69

6.5 Experiments

We conduct a series of experiments on evaluating the proposed GMKL in contrast with the L_1-MKL, the L_2-MKL, and the

uniformly-weighted MKL (UW-MKL) with three objectives. The first objective is to show how our GMKL model can select important kernels in group manners. This is illustrated through two toy examples. The second objective is to show the efficiency of our GMKL model solved by the level method. This is verified by eight datasets from the UCI repository [17]. The third objective is to show the GMKL can improve the performance on predicting the proteins subcellular localization by different kinds of kernels [198]. A summary of the three types of data, including 13 datasets, is listed in Table 6.2 and Table 6.3, respectively. Detailed descriptions of the data are in Section 6.5.2 to Section 6.5.4, respectively.

6.5.1 Experimental Setup

In the experiment, for all compared four MKL models, the regularization parameter C is tuned by cross validation on one run of the training data. The tradeoff parameter v for the GMKL is set to 0.5 for simplicity.

The L_1-MKL is solved by the SimpleMKL toolbox [130]. The L_2-MKL is solved by our GMKL [1] with the parameter $v = 0$. The optimization on constructing the cutting plane of (6.22) and seeking the projection of (6.25) in the level method are solved by a standard toolbox, Mosek [2]. To conduct a fair comparison among the MKL algorithms, we set the stopping criterion similar to that in [130]: The duality gap is lower than 0.01 for the L_1-MKL; for other MKL algorithms (except UW-MKL), when the number of iterations exceeds 500, the difference of $\boldsymbol{\theta}$ in consecutive step is lower than 0.001. For our GMKL and the L_2-MKL, we empirically initialize the algorithm parameter τ in the level method to 0.9 and increase it to 0.99 when the ratio $\Delta^t/\mathcal{V}^t$ is

[1] Our GMKL toolbox can be downloaded in `http://appsrv.cse.cuhk.edu.hk/~hqyang/doku.php?id=gmkl`.

[2] `http://www.mosek.com`

less than 0.01 for all experiments, since a larger τ accelerates the projection when the solution is close to the optimal one.

6.5.2 Toy Examples

In designing the synthetic datasets, we have the following expectations: 1) data containing nonlinearity on the features; and 2) data being embedded with redundant and grouping features. We then generate two 20-dimensional toy examples by additive models motivated by an example in [65].

1. In example 1, the data are generated by

$$Y_i = \text{sign}\left(\sum_{j=1}^{3} f_1(x_{ij}) + \epsilon_i\right), \tag{6.27}$$

where sign $(\cdot)$ is determined by the sign of the value in the bracket, $\mathbf{x}$ is uniformly distributed in $[0,1]^{N\times 20}$, $f_1(a) = -2\sin(2a) + 1 - \cos(2)$ and the noise $\epsilon_i \sim \mathcal{N}(0,1)$ is a Gaussian noise. Hence, the data contain 17 irrelevant features.

2. In example 2, the data are generated by

$$Y_i = \text{sign}\left(\sum_{j=1}^{3} f_1(x_{ij}) + \sum_{j=4}^{6} f_2(x_{ij}) + \sum_{j=7}^{9} f_3(x_{ij}) + \sum_{j=10}^{12} f_4(x_{ij}) + \epsilon_i\right), \tag{6.28}$$

where there are four kinds of mapping f_1, f_2, f_3, and f_4. f_1 is the same as Example 1, $f_2(a) = a^2 - \frac{1}{3}$, $f_3(a) = a - \frac{1}{2}$, and $f_4(a) = e^{-a} + e^{-1} - 1$. For $\mathbf{x}$ and ϵ_i, they are the same as Example 1. The output Y_i is determined by the corresponding features, from 1 to 12, of $\mathbf{x}_i$ which are mapped by f_1, f_2, f_3, and f_4, respectively. The data therefore contain 8 irrelevant features.

It is noted that by the above generation scheme, the data have the following properties:

1. The outputs (labels) of the data are dominated by only some features. The corresponding feature is mapped by a linear function as f_3, or non-linear functions, f_1, f_2, and f_4.

2. Each mapping, f_i, $i = 1, 2, 3, 4$, acts on three features equally, which implicitly incorporates grouping effect on those features.

3. The mean of the output is zero since each mapping is with zero mean on the corresponding feature.

In the experiment, we randomly sample 300 instances, where 150 data are used for training and the other 150 data are used for test. Following the settings of [130], we construct the base kernel matrices as follows

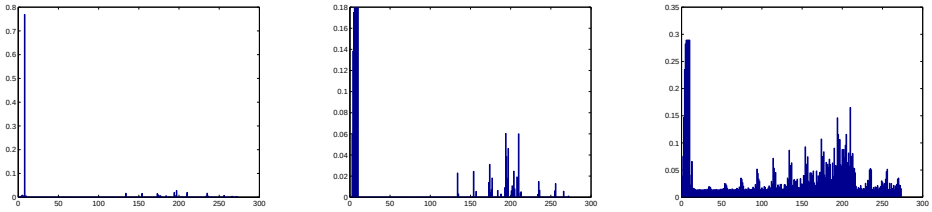
- Gaussian kernels with 10 different widths ($\{2^{-3}, 2^{-2}, \ldots, 2^{6}\}$) on all features and on each single feature;
- Polynomial kernels of degree 1 to 3 on all features and on each single feature.

Each base kernel matrix is further normalized to unit trace as [130]. Therefore, we build 273 kernels for the toy examples.

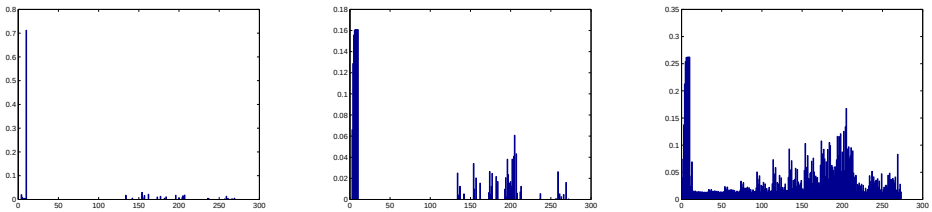
Table 6.4 reports the average accuracy, number of selected kernels, and executed time after repeating the algorithms 20 times. Our GMKL obtains significant improvement on the accuracy against the L_1-MKL and the L_2-MKL with 99% confidence level on the paired t-test. The results show that our GMKL can utilize the grouping structure information embedded in the data sufficiently. Table 6.4 also shows that both the L_2-MKL and the UW-MKL achieve worse accuracies than the sparse MKL models. This verifies that the non-sparse MKL models are prone to noise. In terms of the number of selected kernels, our GMKL selects more kernels, about 1.5 times of that selected by the

Table 6.4: Average performance measured by our GMKL, the L_1-MKL, the L_2-MKL, and the UW-MKL algorithms on Toy examples.

Dataset	Method	Accuracy	# Kernel	Times (s)
Toy 1	GMKL	**70.4**±3.3	36.8±5.0	2.9±0.2
	L_1-MKL	69.2±4.5	22.1±5.2	4.4±1.2
	L_2-MKL	68.2±3.0	273	2.9±0.4
	UW-MKL	66.3±5.3	273	–
Toy 2	GMKL	**72.9**±3.2	43.4±7.1	2.8±0.1
	L_1-MKL	72.3±3.1	30.2±8.1	4.9±1.3
	L_2-MKL	71.9±3.6	273	2.9±0.1
	UW-MKL	71.6±4.0	273	–



(a) L_1-MKL on Toy 1 (b) GMKL on Toy 1 (c) L_2-MKL on Toy 1

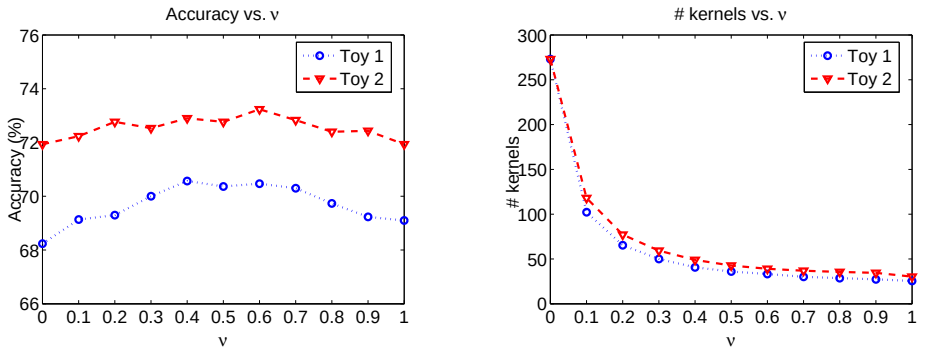


(d) L_1-MKL on Toy 2 (e) GMKL on Toy 2 (f) L_2-MKL on Toy 2

Figure 6.2: Figures in the first line correspond to the kernel weights on the Toy 1 example learned by the L_1-MKL, the GMKL, and the L_2-MKL, respectively. Figures in the second line are the kernel weights on Toy 2 example learned by the L_1-MKL, the GMKL, and the L_2-MKL, respectively. The L_1-MKL selects few kernels and discards some useful information. The L_2-MKL selects all kernels and is easily affected by the noise. Meanwhile, the GMKL selects suitable kernels with the grouping effect.

L_1-MKL; while the L_2-MKL selects all kernels (see Figure 6.2 for more details). The computation cost of our GMKL and the L_2-MKL is nearly the same, and they cost less time than that of the L_1-MKL. This is due to that the level method consumes less outer iterations than the SimpleMKL [130] used.

Figure 6.2 further shows the average coefficients obtained by the L_1-MKL, the GMKL, and the L_2-MKL. The figure again shows the grouping effect and the sparsity of our GMKL. The results in the figures refer to the results in the fourth column of the Table 6.4.



(a) Accuracy vs. v (b) No. of selected kernels vs. v

Figure 6.3: Accuracy and the number of selected kernels by the GMKL with varying v on the toy datasets. It is noted that the best accuracy for the Toy 1 dataset is achieved when $v = 0.4$ while the best accuracy for the Toy 2 dataset is achieved when $v = 0.6$. The number of selected kernels decreases as v increases (see text for more descriptions).

We further test the effect of v on the accuracy and the number of selected kernels for the toy datasets. We vary v from 0 to 1 with an incremental step being 0.1 and show the results in Figure 6.3. Actually, Figure 6.3 includes the results reported in Table 6.4, the L_1-MKL ($v = 1$), the L_2-MKL ($v = 0$), and our GMKL with $v = 0.5$. It is shown that the optimal v is around 0.5 for both toy datasets. Figure 6.3(b) indicates that as v increases, the number of selected kernels decreases. This shows

that the optimal v is data-dependent, i.e., a better v corresponds to the suitable number of kernels selected for that training data. Hence, usually we can tune the parameter v by cross-validation on the training data.

6.5.3 UCI Datasets

In order to verify the performance of our GMKL on datasets which do not show manifest group structure on the base kernels, we employ eight UCI datasets in our test, i.e., Breast, Heart, Ionosphere, Liver, Pima, Sonar, Wdbc, and Wpbc, from the UCI repository [17] are used in our test. These datasets have been frequently used in evaluating the MKL models [96, 130, 170].

We repeat all the algorithms 20 times on each dataset. In each run, 50% of the examples are randomly selected as the training data and the remaining data are used for testing. The training data are normalized to have zero mean and unit variance, and the test data are then normalized using the mean and variance of the training data. The construction and the post-processing of the base kernel matrices are conducted the same as the synthetic data in Section 6.5.2.

Table 6.5 reports the average results, including accuracy, the number of selected kernels, and the running time, on the UCI datasets. Our GMKL achieves the highest accuracy for five datasets: "Breast", "Heart", "Pima", "Wdbc", and "Wpbc". Especially, for the datasets of "Pima" and "Wdbc", our GMKL obtains significantly better results. The L_2-MKL gets the highest accuracy for the rest three datasets: "Ionosphere", "Liver", and "Sonar", and attains significantly better results for "Liver" and "Sonar". The UW-MKL gets the same highest accuracy as the GMKL for "Breast" and "Heart". It is important to note that better results can be obtained by tuning v through cross-validation on the training data. For example, the cross-

validation procedure on the "ionosphere" data set suggests that a smaller v with the value near zero can recover the result of the L_2-MKL.

In terms of the number of selected kernels, on average, our GMKL selects a little more kernels than the L_1-MKL, owing to the grouping effect on some features. This is deserved since among all the datasets, our GMKL achieves no worse results than the L_1-MKL. Especially, our GMKL improves the accuracy from 64.3% to 67.6% for the Liver dataset, and from 95.3% to 96.0% for the Wdbc dataset.

For the running time, our GMKL is efficient. The time needed by our GMKL and the L_2-MKL is much less than that used in the L_1-MKL for the datasets of Breast, Ionosphere, Pima, and Wdbc. Especially, for the datasets of Breast, Pima, and Wdbc, the number of data points is relatively larger than other datasets, the SimpleMKL costs more time. This is because that the simple MKL has to solve more QP problems when updating the descent direction. When the number of training samples is large, more time is required in the SVM solver.

6.5.4 Protein Subcellular Localization Datasets

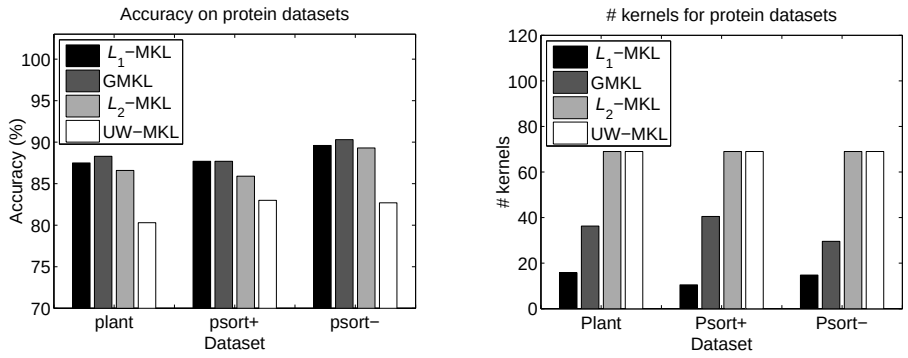
Three datasets are used to predict the proteins subcellular localization [3], where the plant dataset of TargetP is a four class problem, and the other two datasets of bacterial protein locations are the psort+ dataset consisting of four classes and the Psort– dataset consisting of five classes. The summary of the datasets is in Table 6.3. MKL methods have succeeded in these datasets with those well-defined graph kernels [90, 198]. We hypothesize the graph kernels may still provide the grouping effect and help to improve the prediction performance.

For the proteins subcellular localization datasets, we follow

[3] http://www.fml.tuebingen.mpg.de/raetsch/suppl/protsubloc/

Table 6.5: Average performance measured by the GMKL, the L_1-MKL, the L_2-MKL, and the UW-MKL algorithms on UCI datasets. Better results are in bold. Significantly better results with 95% confidence level over other methods are indicated by †.

Dataset	Method	Accuracy	# Kernel	Times (s)
Breast	GMKL	**97.2**±0.5	61.1±6.5	2.8±0.5
	L_1-MKL	97.0±0.7	18.6±3.8	23.0±3.9
	L_2-MKL	96.9±0.4	143	5.1±0.3
	UW-MKL	**97.2**±0.5	143	–
Heart	GMKL	**83.9**±1.9	38.5±5.4	1.4±0.1
	L_1-MKL	83.4±2.6	29.7±4.6	3.5±0.7
	L_2-MKL	82.8±2.5	182	1.7±0.1
	UW-MKL	**83.9**±1.9	182	–
Ionosphere	GMKL	91.8±1.7	66.5±7.2	5.1±0.3
	L_1-MKL	91.5±2.1	38.4±5.0	19.2±3.3
	L_2-MKL	**92.0**±1.8	442	4.0±0.4
	UW-MKL	89.9±1.8	442	–
Liver	GMKL	67.6±1.8	19.5±1.7	1.0±0.0
	L_1-MKL	64.3±2.8	9.2±3.0	1.7±0.4
	L_2-MKL	†**69.7**±2.2	91	1.4±0.0
	UW-MKL	67.2±4.6	91	–
Pima	GMKL	†**76.9**±1.6	27.1±2.4	3.8±0.2
	L_1-MKL	76.5±1.9	18.7±2.7	24.8±3.4
	L_2-MKL	76.0±1.8	117	6.2±1.0
	UW-MKL	76.2±1.7	117	–
Sonar	GMKL	80.4±4.1	81.1±6.5	12.4±0.6
	L_1-MKL	80.4±4.2	60.3±7.4	16.7±2.0
	L_2-MKL	†**83.8**±3.7	793	3.9±0.3
	UW-MKL	81.5±4.3	793	–
Wdbc	GMKL	**96.0**±1.1	79.7±7.6	6.6±0.8
	L_1-MKL	95.3±1.4	34.9±8.9	37.8±5.8
	L_2-MKL	95.9±0.7	403	7.8±1.6
	UW-MKL	93.9±1.0	403	–
Wpbc	GMKL	**76.7**±3.3	275.4±96.9	1.3±1.0
	L_1-MKL	76.6±2.8	40.4±10.2	4.8±1.0
	L_2-MKL	76.3±3.7	442	1.6±0.2
	UW-MKL	76.6±2.9	442	–



(a) Accuracy (b) No. of selected kernels

Figure 6.4: Accuracy and the number of kernels selected by the L_1-MKL, the GMKL, and the L_2-MKL on the protein subcellular localization datasets, where the L_2-MKL and the UW-MKL select all the 69 kernels. Our GMKL achieves the best results on all datasets and selects about three times to four times of kernels compared to that selected by the L_1-MKL. It should be noted that here the accuracy of the plant dataset is measured by the Matthew's Correlation Coefficient (MCC) [198], while for the psort+ and the psort- datasets, it is measured by the F1 score.

the setup of [198] and construct 69 kernels: 2 kernels on phylogenetic trees, 3 kernels from BLAST E-values, and 64 sequence motif kernels. Each kernel for the proteins subcellular localization datasets is normalized such that the implied variance equal one as [198] by $\mathbf{K}(\mathbf{x}, \mathbf{z}) = \mathbf{K}(\mathbf{x}, \mathbf{z}) / \left(\frac{1}{N} \sum_{i=1}^{N} \mathbf{K}(\mathbf{x}_i, \mathbf{x}_i) - \frac{1}{N^2} \sum_{i,j=1}^{N} \mathbf{K}(\mathbf{x}_i, \mathbf{x}_j) \right)$.

Different from [198], we randomly split the protein subcellular localization datasets into two parts equally, where half the data are used for training and the rest of the data are used for test. We use the 1-vs-rest scheme on the multi-class classification problems. As in [198], the Matthew's Correlation Coefficient (MCC) is used to evaluate the plant dataset and the F1 score is used to evaluate the psort+ and the Psort– datasets.

Figure 6.4(a) reports the average results on 10 runs. Our GMKL achieves the best results on all three datasets. The MCC obtained by our GMKL for the plant dataset is 88.3%

compared to 87.5% obtained by the L_1-MKL, 86.6% obtained by the L_2-MKL, and 80.3% obtained by the UW-MKL. For the two bacterial protein locations datasets, our GMKL and the L_1-MKL gets the same 87.7% F1 score compared to the L_2-MKL of 85.9% and the UW-MKL of 83.0% for the psort+ dataset and obtains 90.3% F1 score compared to the L_1-MKL of 89.6%, the L_2-MKL of 89.3%, and the UW-MKL of 82.7%. Hence, the results verify our hypothesis.

Table 6.6: p-values of the paired t-test of our GMKL vs. the L_1-MKL, the L_2-MKL and the UW-MKL on the proteins subcellular localization datasets.

Dataset	GMKL vs. L_1-MKL	GMKL vs. L_2-MKL	GMKL vs. UW-MKL
Plant	0.319	0.054	**0.000**
Psort+	0.545	**0.047**	**0.002**
Psort–	**0.049**	**0.040**	**0.000**

To further verify whether our GMKL model performs statistically better than the other three MKL methods, we report the p-values of the paired t-test of our GMKL on the L_1-MKL, the L_2-MKL, the UW-MKL in Table 6.6. The results show that our GMKL improves the classification accuracy significantly compared to the UW-MKL for all three protein datasets. Our GMKL performs significantly better results compared to the L_2-MKL for the psort+ dataset and the Psort– dataset. Compared to the L_1-MKL, our GMKL performs significantly better results for the Psort– dataset.

Figure 6.4(b) shows the number of selected kernels by the L_1-MKL, the GMKL, and the L_2-MKL. Our GMKL again selects more kernels than the L_1-MKL. Figure 6.5 further shows the obtained kernel weights by the L_1-MKL, our GMKL, and the L_2-MKL for the protein subcellular localization datasets. Our GMKL again can obtain sparse solutions with the grouping ef-

fect. It is noted that most of the groups are embedded in the sequence motif kernels and captured by our GMKL.

In summary, the experimental results in the above section indicate the good performance in terms of accuracy, sparsity, and efficiency. The advantage of our GMKL is more explicit on data with latent group structure.

6.6 Summary

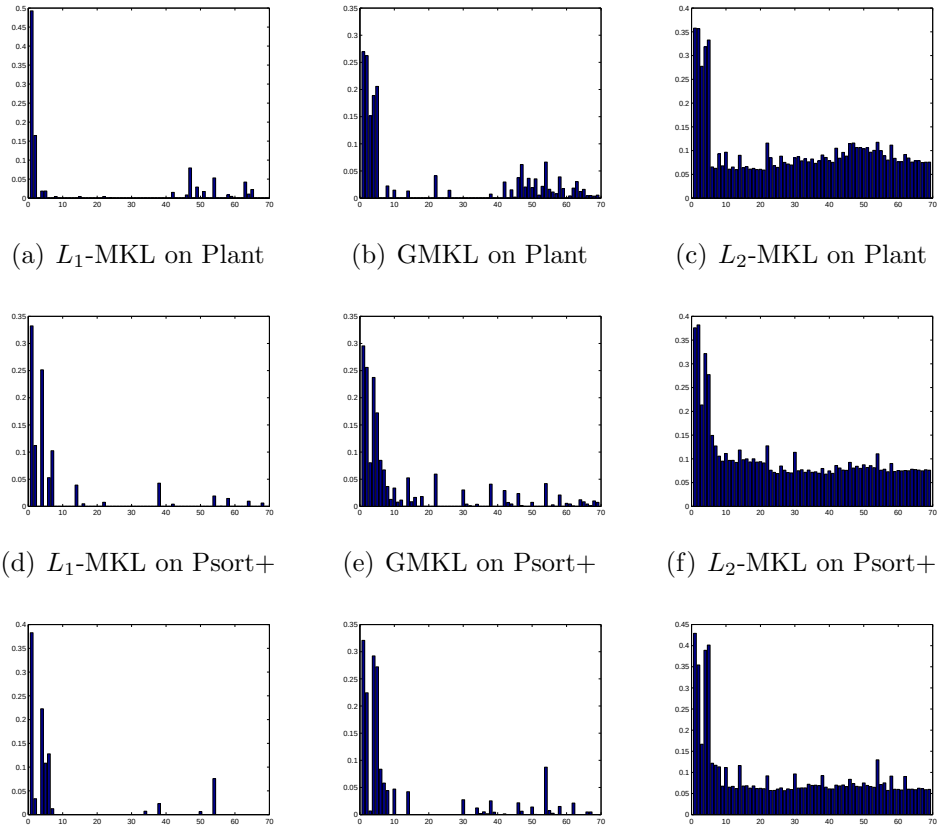
In this work, we presented a generalized multiple kernel learning (GMKL) model by introducing a linear combination of the L_1-norm and the squared L_2-norm regularization on the kernel weights to seek the optimal kernel combination. Our GMKL generalizes previously proposed L_1-MKL and the L_2-MKL methods. The theoretical analysis on the GMKL guarantees to having sparse solutions and also encourages the grouping effect. Moreover, the optimization of GMKL is a convex optimization problem, where the global optimality can be assured. We further derive a level method to efficiently solve the optimization problem, followed by the convergence analysis and optimal condition on the algorithm.

Experimental results on both synthetic and real-world datasets indicate that the proposed GMKL can take advantage of the group structure of data, and thus produce sparse solutions accordingly. In addition, it keeps the balance between accuracy and sparsity of MKL: it improves the accuracy of the L_1-MKL, and at the same time produces more sparse solutions than the L_2-MKL while achieving competitive accuracy. Moreover, the reported running time on the datasets indicates the efficiency of the level method on solving our GMKL.

There are several future work associated with our GMKL. First, it would be interesting to apply our GMKL model in other applications, e.g., regression, multiclass classification problems,

etc. Second, it is promising to employ advanced optimization techniques to speed up our GMKL, e.g., employing warm start on the previously obtained solution on solving the optimization problem with a quadratical-constraint, or solving the optimization problem by second-order methods or coordinate-wise optimizers. Third, it is attractive to extend our GMKL to include the uniformly-weighted MKL as a special case.

□ **End of chapter.**



(a) L_1-MKL on Plant (b) GMKL on Plant (c) L_2-MKL on Plant

(d) L_1-MKL on Psort+ (e) GMKL on Psort+ (f) L_2-MKL on Psort+

(g) L_1-MKL on Psort– (h) GMKL on Psort– (i) L_2-MKL on Psort–

Figure 6.5: Figures in the first line correspond to the average kernel weights on the plant dataset learned by the L_1-MKL, the GMKL, and the L_2-MKL, respectively. Figures in the second line are the average kernel weights on the psort+ dataset learn by the L_1-MKL, the GMKL, and the L_2-MKL, respectively. Figures in the third line are the average kernel weights on the Psort– dataset learn by the L_1-MKL, the GMKL, and the L_2-MKL, respectively. The horizontal axis indexes the 69 kernels. The two phylogenetic profile kernels and the three BLAST E-value kernels are on the left. The L_1-MKL selects few kernels and the L_2-MKL selects all kernels. Meanwhile, the GMKL selects suitable number of kernels. Most of the grouping kernels are from the sequence motif kernels.

Chapter 7

Conclusion and Future Work

7.1 Conclusion

In this chapter, we provide a summary of the thesis. The thesis provides promising solutions for the large-scale applications in three main learning areas: online learning, semi-supervised learning, and multiple kernel learning.

In the first part of this thesis, we propose a novel online learning framework to solve group lasso and multi-task feature selection. This work is extended from the lasso model to select the important explanatory features in different tasks. Through online learning framework, it can solve the applications where training data appear sequentially and scale well in any number of training data with any number of features. By deriving the closed-form solutions for the corresponding models, we can update the corresponding models in an extremely efficient way. The experimental results show that the proposed online learning algorithms work efficiently while achieving good performance.

In the second part of this thesis, we propose a novel maximum margin semi-supervised learning classifier, 3C-SVM, to learn from both labeled and unlabeled data. Different from the common assumption of previous standard semi-supervised learning models, we assume the data are a mixture of data relevant and irrelevant to the target labeled data. By introducing a novel

min loss function and based on the maximum entropy principle, we can distinguish the relevant data as well as the irrelevant data. The 3C-SVM is solved by CCCP and is potential for solving large-scale datasets. The experimental results show that the 3C-SVM outperforms the standard SVM, S^3VM, and $\mathcal{U}$-SVM.

In the third part of this thesis, we propose a generalized multiple kernel learning model, which contains sparse solutions with the grouping effect. Solving the model by an efficient method, the level method, we can solve the MKL model in large-scale applications. We demonstrate the advantages and properties of the GMKL not only by a series of detailed experiments, but also by theoretical analysis.

In summary, our proposed algorithms and models provide promising solutions for large-scale applications in various aspects.

7.2 Future Work

There are still several important research problems remaining in this thesis. We will elaborate them in the following:

First, in the sparse group lasso model and the multi-task feature task selection model, there is an additional sparse parameter. In this thesis, we set it as a very small value without tuning for simplicity since they have achieved good performance. How to determine a good sparse parameter is still a question.

Second, in the online learning algorithms, there are two main parameters, the model regularization parameter and the algorithm regularization parameter. Currently, we use cross validation method in grid search to tune these two parameters. How to devise solution path algorithms to efficiently explore the entire solution paths is very promising.

Third, the online multi-task feature selection algorithms requires that at one iteration, one instance arrives for each task si-

multaneously. Although this thesis has pointed out one method to remove this requirement, i.e., simply setting the instance for those tasks without coming instances to zero, how to avoid unbalance and to make the learned matrices not bias to those tasks with training instances needs to be explored.

Fourth, the tri-class SVM has relaxed the original integral programming problem into a quadratic programming problem with a box constraint on real variables. What is the approximation bound for the relaxation is an interesting question.

□ **End of chapter.**

Appendix A

Proof of Theorem 1, in Chapter 3

Proof. Since the objective of (3.5) is component-wise, we can focus on the solution in one group, say g. In the following, we first sketch the proof of a) in Theorem 1.

The optimal $\mathbf{w}_{t+1}^g$ in (3.5) should be $\mathbf{w}_{t+1}^g = \kappa_g \bar{\mathbf{u}}_t^g$ with $\kappa_g \leq 0$. Otherwise, we can assume for the sake of contradiction that $\mathbf{w}_{t+1}^g = \kappa_g \bar{\mathbf{u}}_t^g + \mathbf{v}^g$, where $\kappa_g \in \mathbb{R}$ and $\mathbf{v}^g$ is in the null space of $\bar{\mathbf{u}}_t^g$. It is easy to verify that $\mathbf{v}^g$ should be a zero vector.

Next, $\kappa_g > 0$ is not the optimal solution. If $\kappa_g > 0$, it can be easily verified that by setting $\kappa_g = -\kappa_g$ we can obtain a lower objective function value. Hence, the objective of (3.5) becomes

$$\min_{\kappa_g \leq 0} \quad \kappa_g \|\bar{\mathbf{u}}_t^g\|_2^2 - \lambda\sqrt{d_g}\kappa_g \|\bar{\mathbf{u}}_t^g\|_2 + \frac{\gamma}{2\sqrt{t}}\kappa_g^2 \|\bar{\mathbf{u}}_t^g\|_2^2 \tag{A.1}$$

By constructing the Lagrangian, $\mathcal{L}(\kappa_g, \nu)$, of the above optimization problem, we have $\nu \geq 0$ and

$$\mathcal{L} = \kappa_g \|\bar{\mathbf{u}}_t^g\|_2^2 - \lambda\sqrt{d_g}\kappa_g \|\bar{\mathbf{u}}_t^g\|_2 + \frac{\gamma}{2\sqrt{t}}\kappa_g^2 \|\bar{\mathbf{u}}_t^g\|_2^2 + \nu\kappa_g.$$

The Karush-Kuhn-Tucker (KKT) condition indicates the optimal solution must satisfy

$$\frac{\partial \mathcal{L}}{\partial \kappa_g} = \|\bar{\mathbf{u}}_t^g\|_2^2 - \lambda\sqrt{d_g}\|\bar{\mathbf{u}}_t^g\|_2 + \frac{\gamma}{\sqrt{t}}\kappa_g \|\bar{\mathbf{u}}_t^g\|_2^2 + \nu = 0,$$

$$\nu\kappa_g = 0.$$

Hence, the value of $\kappa_g < 0$ iff $\lambda\sqrt{d_g} < \|\bar{\mathbf{u}}_t^g\|_2$. If $\lambda\sqrt{d_g} > \|\bar{\mathbf{u}}_t^g\|_2$, then ν must be positive and κ_g should be zero. The above analysis concludes the closed form of $\mathbf{w}_{t+1}^g$ in (3.10).

The sparse group lasso and the enhanced sparse group lasso have an additional L_1-norm on the weight only with different coefficients. Hence, the proof of b) and c) is similar.

Here, we just sketch the proof of b). Since the objective of (3.5) for the sparse group lasso is also element-wise, we can consider one entry, say j, in the g-th group. The objective of (3.5) on $w_{t+1}^{g,j}$ is

$$\Upsilon(w_{t+1}^{g,j}) = \bar{u}_t^{g,j} w_{t+1}^{g,j} + \lambda r_g |w_{t+1}^{g,j}| + \xi((w_{t+1}^{g,j})^2), \tag{A.2}$$

where $\xi((w_{t+1}^{g,j})^2)$ is a non-negative function on $(w_{t+1}^{g,j})^2$ and $\xi(w_{t+1}^{g,j})^2) = 0$ iff $w_{t+1}^{g,j} = 0$ for all $j \in [1, d_g]$.

If $\bar{u}_t^{g,j} = 0$, obviously, the optimal solution for (A.2) is $w_{t+1}^{g,j} = 0$. When $\bar{u}_t^{g,j} \neq 0$, to simplify the analysis, we first assume $\bar{u}_t^{g,j} > 0$, then $w_{t+1}^{g,j}$ should be non-positive. Otherwise, if $w_{t+1}^{g,j} > 0$, we have $\Upsilon(-w_{t+1}^{g,j}) < \Upsilon(w_{t+1}^{g,j})$. It means that we can set $w_{t+1}^{g,j}$ to its negative and obtain a lower objective function value.

Next, if $\bar{u}_{t+1}^{g,j} \leq \lambda r_g$, then $w_{t+1}^{g,j} = 0$ is the optimal solution. Otherwise, we have $w_{t+1}^{g,j} < 0$ and $\Upsilon(w_{t+1}^{g,j}) = (\bar{u}_t^{g,j} - \lambda r_g) w_{t+1}^{g,j} + \xi((w_{t+1}^{g,j})^2) > \Upsilon(0)$. This implies that by setting $w_{t+1}^{g,j} = 0$ we can obtain a lower objective function value.

Third, $\bar{u}^{g,j} > \lambda r_g$ for all $j \in [1, d_g]$. The objective of (3.5) for the g-th group, $\Upsilon(\mathbf{w}_{t+1}^g)$, becomes

$$(\bar{\mathbf{u}}_t^g - \lambda r_g \mathbf{1}_{d_g})^\top \mathbf{w}_{t+1}^g + \lambda\sqrt{d_g}\|\mathbf{w}_{t+1}^g\|_2 + \frac{\gamma}{2\sqrt{t}}\|\mathbf{w}_{t+1}^g\|_2^2 \tag{A.3}$$

This objective function has the same structure to that of the group lasso in (3.5). The only difference is a slight change in the vector $\bar{\mathbf{u}}_t$. Hence, following the result of a), we can define $c^{g,j}$ as that in (3.12) and obtain a closed form solution in (3.11) for (A.3).

The analysis for $\bar{u}_t^{g,j} < 0$ is similar. Hence, we conclude the proof of b). □

□ **End of chapter.**

Appendix B

Proof of Theorem 3, in Chapter 4

Proof of Theorem 3. In the following, we first prove the result of (a) in Theorem 3. Since the objective of (4.8) is element-wise for the case of $L_{1,1}$-norm regularization defined in (4.4), we can only consider any one element, say, the (i, j)-th element. By denoting $(\bar{G}_{i,j})_t$ and $(W_{i,j})_{t+1}$ with $\bar{g}_t$ and w_{t+1}, respectively, we obtain the objective of (4.8) on the (i, j)-th element as

$$\Upsilon(w_{t+1}) = \bar{g}_t \cdot w_{t+1} + \lambda |w_{t+1}| + \frac{\gamma}{2\sqrt{t}} w_{t+1}^2. \tag{B.1}$$

If $\bar{g}_t = 0$, obviously, the optimal solution for (B.1) is $w_{t+1} = 0$. Now, we consider the case of $\bar{g}_t \neq 0$. To simplify the analysis, we first assume $\bar{g}_t > 0$, then w_{t+1} should be non-positive. Otherwise, if $w_{t+1} > 0$, we have $\Upsilon(-w_{t+1}) < \Upsilon(w_{t+1})$. It means that by setting w_{t+1} to its negative, we can obtain a lower objective function value.

Next, if $\bar{g}_t \leq \lambda$, then the optimal w_{t+1} should be zero. Otherwise, we have $w_{t+1} < 0$ and $\Upsilon(w_{t+1}) = (\bar{g}_t - \lambda) w_{t+1} + \frac{\gamma}{2\sqrt{t}} w_{t+1}^2 > 0 = \Upsilon(0)$. This implies that by setting $w_{t+1} = 0$ we can obtain a lower objective function value.

Third, if $\bar{g}_t > \lambda$, by setting the derivative of (B.1) to zero, we obtain the solution of that in (4.11).

If $\bar{g}_t < 0$, we can follow the above analysis. Hence, we conclude the proof of (a).

Now, we turn to prove (b) in Theorem 3. It is noted that the objective of (4.8) is component-wise on one row of $\mathbf{W}$ for the case of $L_{1,2}$-norm regularization defined in (4.5). Hence, we focus on one row of $\mathbf{W}$, say $\mathbf{W}_{j\bullet}^{\top}$, and use $\mathbf{w}$ to denote it for simplicity. Correspondingly, we use $\bar{\mathbf{g}}_t$ to denote $(\bar{\mathbf{G}}_{j\bullet})_t$. Then, the objective of (4.8) on $(\mathbf{W}_{j\bullet}^{\top})_t$ becomes

$$\Upsilon(\mathbf{w}_{t+1}) = \bar{\mathbf{g}}_t^{\top}\mathbf{w}_{t+1} + \lambda\|\mathbf{w}_{t+1}\|_2 + \frac{\gamma}{2\sqrt{t}}\|\mathbf{w}_{t+1}\|_2^2. \tag{B.2}$$

It is noted that the optimal $\mathbf{w}_{t+1}$ in (B.2) should be $\mathbf{w}_{t+1} = \kappa\bar{\mathbf{g}}_t$ with $\kappa \leq 0$. Otherwise, for the sake of contradiction, we can assume that $\mathbf{w}_{t+1} = \kappa\bar{\mathbf{g}}_t + \mathbf{v}$, where $\kappa \in \mathbb{R}$ and $\mathbf{v}$ is in the null space of $\bar{\mathbf{g}}_t$. It is easy to verify that $\mathbf{v}$ should be a zero vector.

Next, $\kappa > 0$ is not the optimal solution. If $\kappa > 0$, it can be easily verified that by setting $\kappa = -\kappa$ we can obtain a lower objective function value. Hence, the objective of (B.2) becomes

$$\min_{\kappa \leq 0} \quad \kappa\|\bar{\mathbf{g}}_t\|_2^2 - \lambda\kappa\|\bar{\mathbf{g}}_t\|_2 + \frac{\gamma}{2\sqrt{t}}\kappa^2\|\bar{\mathbf{g}}_t\|_2^2 \tag{B.3}$$

By constructing the Lagrangian of the above optimization problem, we have $\nu \geq 0$ and

$$\mathcal{L}(\kappa, \nu) = \kappa\|\bar{\mathbf{g}}_t\|_2^2 - \lambda\kappa\|\bar{\mathbf{g}}_t\|_2 + \frac{\gamma}{2\sqrt{t}}\kappa^2\|\bar{\mathbf{g}}_t\|_2^2 + \nu\kappa.$$

The Karush-Kuhn-Tucker (KKT) condition [22] indicates that the optimal solution must satisfy

$$\frac{\partial\mathcal{L}}{\partial\kappa} = \|\bar{\mathbf{g}}_t\|_2^2 - \lambda\|\bar{\mathbf{g}}_t\|_2 + \frac{\gamma}{\sqrt{t}}\kappa\|\bar{\mathbf{g}}_t\|_2^2 + \nu = 0,$$
$$\nu\kappa = 0.$$

This leads to

$$\kappa = -\frac{\sqrt{t}}{\gamma}\left(1 - \frac{\lambda}{\|\bar{\mathbf{g}}_t\|_2} + \frac{\nu}{\|\bar{\mathbf{g}}_t\|_2^2}\right) \tag{B.4}$$

The KKT conditions indicate that the value of $\kappa < 0$ iff $\lambda < \|\bar{\mathbf{g}}_t\|_2$. If $\lambda > \|\bar{\mathbf{g}}_t\|_2$, then ν must be positive and κ should be zero. By substituting $(\mathbf{W}_{j\bullet}^\top)_{t+1}$ with $\mathbf{w}_{t+1}$ and $(\bar{\mathbf{G}}_{j\bullet}^\top)_t$ with $\mathbf{g}_t$ back to (B.4), we obtain the closed form solution of $\mathbf{W}_{t+1}$ as that in (4.12).

We now sketch the proof of (c) in Theorem 3. Similar to the proof of (b), we use $\bar{\mathbf{g}}_t$ to denote $(\bar{\mathbf{G}}_{j\bullet}^\top)_t$ and $\mathbf{w}_{t+1}$ to denote $\left(\mathbf{W}_{j\bullet}^\top\right)_{t+1}$, and c to denote c_j for simplicity. Then, the objective of (4.8) on $(\mathbf{W}_{j\bullet}^\top)_{t+1}$ becomes

$$\Upsilon(\mathbf{w}_{t+1}) = \bar{\mathbf{g}}_t^\top \mathbf{w}_{t+1} + \lambda(c\|\mathbf{w}_{t+1}\|_1 + \|\mathbf{w}_{t+1}\|_2) + \frac{\gamma}{2\sqrt{t}}\|\mathbf{w}_{t+1}\|_2^2. \tag{B.5}$$

It is noted that the objective in (B.5) is element-wise. Hence we consider one element, say k. The objective of (B.5) on $(w_q)_{t+1}$ then becomes

$$\Upsilon\left((w_q)_{t+1}\right) = (\bar{g}_q)_t \cdot (w_q)_{t+1} + \lambda c|(w_q)_{t+1}| + \xi((w_q)_{t+1}^2), \tag{B.6}$$

where $\xi((w_q)_{t+1}^2)$ is a non-negative function on $(w_q)_{t+1}^2$ and $\xi((w_q)_{t+1}^2) = 0$ iff $(w_q)_{t+1} = 0$ for all $k \in [1, Q]$.

It is noted that the objective of (B.6) follows the same structure of (B.1) with the only difference on the coefficient of the first term. Hence, similar to the proof of (a), we can first assume $(\bar{g}_q)_t \geq 0$ and can easily conclude that the optimal $(w_q)_{t+1} = 0$ when $(\bar{g}_q)_t \leq \lambda c$. Hence, when $(\bar{g}_q)_t \geq \lambda c$, for all $q = 1, \ldots, Q$, the objective of (B.5) becomes

$$\Upsilon(\mathbf{w}_{t+1}) = (\bar{\mathbf{g}}_t - \lambda c)^\top \mathbf{w}_{t+1} + \lambda\|\mathbf{w}_{t+1}\|_2 + \frac{\gamma}{2\sqrt{t}}\|\mathbf{w}_{t+1}\|_2^2 \tag{B.7}$$

The above objective function has the same structure as that in (B.2) with the only difference in the coefficient of the first term. Hence, following the result of (b), we can define $(\bar{\mathbf{U}}_{j\bullet}^\top)_t$ as that in (4.14) and obtain a closed form solution as that in (4.13) for (B.7). The analysis for $(\bar{g}_q)_t < 0$ is similar and we conclude the proof of (c). □

□ **End of chapter.**

Appendix C

Proof in Chapter 5

Proof. Due to the symmetry of ε-insensitive loss function, I_ε, we first introduce new pair-training data for $\mathcal{L}_0$-data in the convex term, Q_{vex}, as $\mathcal{U}$-SVM in [165]. That is for $\mathcal{L}_0$-data, we introduce new paired data as $\mathbf{x}_{-i} = \mathbf{x}_i$, $y_{-i} = -1$ and $\mathbf{x}_i = \mathbf{x}_i$, $y_i = 1$, for $i = 1, \ldots, |\mathcal{L}_0|$. In this setting, when no $\mathcal{L}_0$-data, i.e., $|\mathcal{L}_0| = 0$, no new paired data are introduced. Here, for simplicity, but in a slight abuse of notation, we use $-i$ to indicate the corresponding index shifting i advance from the 0-index. Hence, the relaxed

optimization in (5.4) can be expanded as

$$\min_{\mathbf{w},b,\boldsymbol{\xi},\boldsymbol{\xi}^*,\mathbf{d}} \frac{\lambda}{2}\|\mathbf{w}\|^2 + \sum_{i=-|\mathcal{L}_0|,i\neq 0}^{L} r_i\xi_i + \sum_{i=L+1}^{L+2U} r_i(\xi_i+\xi_i^*) + \sum_{i=L+1}^{L+2U} \mu_i y_i f_{\boldsymbol{\vartheta}}(\mathbf{x}_i) \tag{C.1}$$

$$\text{s.t.} \quad \begin{cases} y_i f_{\boldsymbol{\vartheta}}(\mathbf{x}_i) + \varepsilon + \xi_i \geq 0, i = -|\mathcal{L}_0|, \ldots, |\mathcal{L}_0|, i \neq 0 \\ y_i f_{\boldsymbol{\vartheta}}(\mathbf{x}_i) - 1 + \xi_i \geq 0, i = |\mathcal{L}_0|+1, \ldots, L, \\ y_{k+L} f_{\boldsymbol{\vartheta}}(\mathbf{x}_{k+L}) + D(1-d_k) - 1 + \xi_{k+L} \geq 0, \\ y_{k+LU} f_{\boldsymbol{\vartheta}}(\mathbf{x}_{k+LU}) + D(1-d_k) - 1 + \xi_{k+LU} \geq 0, \\ y_{k+L} f_{\boldsymbol{\vartheta}}(\mathbf{x}_{k+L}) + Dd_k + \varepsilon + \xi_{k+L}^* \geq 0, \\ y_{k+LU} f_{\boldsymbol{\vartheta}}(\mathbf{x}_{k+LU}) + Dd_k + \varepsilon + \xi_{k+LU}^* \geq 0, \\ \xi_i \geq 0, \qquad i = -|\mathcal{L}_0|, \ldots, L+2U, i \neq 0, \\ \xi_i^* \geq 0, \qquad i = L+1, \ldots, L+2U, \\ 0 \leq d_k \leq 1, \qquad k = 1, \ldots, U. \end{cases}$$

This is a standard QP problem with inequality constraints. We can adopt the standard Lagrange multiplier method [16, 22] to solve it.

Hence, we construct the corresponding Lagrange function,

$\mathcal{L}(\mathbf{w}, b, \boldsymbol{\xi}, \boldsymbol{\xi}_i, \mathbf{d}, \boldsymbol{\alpha}, \boldsymbol{\alpha}^*, \boldsymbol{\gamma}, \boldsymbol{\gamma}^*, \mathbf{p}, \mathbf{q})$, as follows:

$$\begin{aligned}
\mathcal{L}=&\frac{\lambda}{2}\|\mathbf{w}\|^2+\sum_{i=-|\mathcal{L}_0|,i\neq 0}^{L} r_i\xi_i+\sum_{i=L+1}^{L+2U} r_i(\xi_i+\xi_i^*)\\
&+\sum_{i=L+1}^{L+2U}\mu_i y_i f_{\boldsymbol{\vartheta}}(\mathbf{x}_i)-\sum_{i=-|\mathcal{L}_0|,i\neq 0}^{|\mathcal{L}_0|}\alpha_i(y_i f_{\boldsymbol{\vartheta}}(\mathbf{x}_i)+\varepsilon+\xi_i)\\
&-\sum_{i=|\mathcal{L}_0|+1}^{L}\alpha_i(y_i f_{\boldsymbol{\vartheta}}(\mathbf{x}_i)-1+\xi_i)\\
&-\sum_{k=1}^{U}\alpha_{k+L}(y_{k+L} f_{\boldsymbol{\vartheta}}(\mathbf{x}_{k+L})+D(1-d_k)-1+\xi_{k+L})\\
&-\sum_{k=1}^{U}\alpha_{k+LU}(y_{k+LU} f_{\boldsymbol{\vartheta}}(\mathbf{x}_{k+LU})+D(1-d_k)-1+\xi_{k+LU})\\
&-\sum_{k=1}^{U}\alpha_{k+L}^*(y_{k+L} f_{\boldsymbol{\vartheta}}(\mathbf{x}_{k+L})+Dd_k+\varepsilon+\xi_{k+L}^*)\\
&-\sum_{k=1}^{U}\alpha_{k+LU}^*(y_{k+LU} f_{\boldsymbol{\vartheta}}(\mathbf{x}_{k+LU})+Dd_k+\varepsilon+\xi_{k+LU}^*)\\
&-\sum_{i=-|\mathcal{L}_0|,i\neq 0}^{L+2U}\gamma_i\xi_i-\sum_{i=L+1}^{L+2U}\gamma_i^*\xi_i^*-\sum_{k=1}^{U}p_k(1-d_k)-\sum_{k=1}^{U}q_k d_k
\end{aligned}$$

Hence, taking the derivative of $\mathcal{L}$ with respect to the primal variables, setting them to zeros, and utilizing the conditions of $\boldsymbol{\gamma}\geq\mathbf{0}$ and $\boldsymbol{\gamma}^*\geq\mathbf{0}$, we obtain

$$\mathbf{w}=\frac{1}{\lambda}\left(\sum_{i=-|\mathcal{L}_0|,i\neq 0}^{L+2U}\alpha_i y_i\phi(\mathbf{x}_i)+\sum_{i=L+1}^{L+2U}(\alpha_i^*-\mu_i)y_i\phi(\mathbf{x}_i)\right), \qquad \text{(C.2)}$$

and

$$\sum_{i=-|\mathcal{L}_0|,i\neq 0}^{L+2U} \alpha_i y_i + \sum_{i=L+1}^{L+2U} \alpha_i^* y_i = \sum_{i=L+1}^{L+2U} \mu_i y_i, \tag{C.3}$$

$$0 \leq \alpha_i \leq r_i, \qquad i = -|\mathcal{L}_0|, \ldots, L+2U, i \neq 0,$$

$$0 \leq \alpha_i^* \leq r_i, \qquad i = L+1, \ldots, L+2U,$$

$$D(\alpha_{k+L} + \alpha_{k+LU} - \alpha_{k+L}^* - \alpha_{k+LU}^*) = q_k - p_k, \tag{C.4}$$

where $p_k, q_k \geq 0, \quad k = 1, \ldots, U$.

Similar to the solution in SVMs [160], minimizing the objective in (C.1) corresponds to maximizing the following objective

$$\max_{\boldsymbol{\alpha},\boldsymbol{\alpha}^*,\mathbf{p},\mathbf{q}} \quad -\frac{1}{2\lambda}[\boldsymbol{\alpha};\boldsymbol{\alpha}^*]^\top \boldsymbol{\Omega}[\boldsymbol{\alpha};\boldsymbol{\alpha}^*] + \boldsymbol{\varrho}^\top[\boldsymbol{\alpha};\boldsymbol{\alpha}^*] - \mathbf{p}^\top \mathbf{1} \tag{C.5}$$

$$\text{s.t.} \quad (C.3) - (C.4), \text{ and } \mathbf{p} \geq \mathbf{0}, \ \mathbf{q} \geq \mathbf{0}, \tag{C.6}$$

where $\boldsymbol{\Omega} = \begin{bmatrix} Q_{|\mathcal{L}_0|+L+2U,|\mathcal{L}_0|+L+2U} & Q_{|\mathcal{L}_0|+L+2U,2U} \\ Q_{2U,|\mathcal{L}_0|+L+2U} & Q_{2U,2U} \end{bmatrix}$, and

$$\boldsymbol{\varrho} = \frac{1}{\lambda} \begin{bmatrix} Q_{2U,|\mathcal{L}_0|+L+2U} \\ Q_{2U,2U} \end{bmatrix} \boldsymbol{\mu} + \begin{bmatrix} -\varepsilon \mathbf{1}_{2|\mathcal{L}_0|} \\ \mathbf{1}_{L-|\mathcal{L}_0|} \\ (1-D)\mathbf{1}_{2U} \\ -\varepsilon \mathbf{1}_{2U} \end{bmatrix}.$$

In (C.5), the variable $[\boldsymbol{\alpha};\boldsymbol{\alpha}^*]$ constructs an $|\mathcal{L}_0|+L+4U$-dimensional vector. The kernel expression, Q, is abstracted as $Q_{\mathcal{R},\mathcal{C}}$, where $\mathcal{R}$ and $\mathcal{C}$ indicate the corresponding row and column ranges of data indices. For $\mathcal{R} = 2U$, it means the row index ranges from $L+1$ to $L+2U$. For $\mathcal{R} = |\mathcal{L}_0| + L + 2U$, in a little abuse the notation of index, it denotes the row index ranges from $-|\mathcal{L}_0|$ to $L+2U$ without the 0 index. $\mathcal{C}$ is defined in the same way. The (i,j)-element of $Q_{i,j}$ is calculated by a kernel, $k(\mathbf{x}_i, \mathbf{x}_j)$. $\mathbf{1}_n$ is an n-dimensional vector with all elements being 1.

In the following, we analyze the optimization in (C.5) on how to discard variables $\mathbf{p}$ and $\mathbf{q}$. The following are two reasons:

1. Since p_k and d_k are non-negative, in order to maximize the objective in (C.5), we will get $p_k d_k = 0$, for all k. In addition, from the KKT conditions, we have $p_k(1-d_k) = 0$. Summarizing these two equalities, we obtain $p_k = 0$.

2. After p_k vanishes, adding the condition of $q_k \geq 0$, we can transform the inequality constraint of (C.4) to $\alpha_{k+L} + \alpha_{k+LU} - \alpha^*_{k+L} - \alpha^*_{k+LU} \geq 0$, for $k = 1, \ldots, U$.

Now we define $\mathbf{A}_e^\top = [\mathbf{Y}; \mathbf{Y}_{\bullet 2U}]$ and $\mathbf{A} = [\mathbf{0}_{U,L}, \ -\mathbf{I}_U, \ -\mathbf{I}_U, \ \mathbf{I}_U, \ \mathbf{I}_U]$, where $\mathbf{Y}$ is a vector containing the label value of all training data with the index ranging from $-|\mathcal{L}_0|$ to $L+2U$ exclusive 0, $\mathbf{Y}_{\bullet 2U}$ is a vector consisting of the label value for the unlabeled data with the index ranging from $L+1$ to $L+2U$. With these notations, we obtain the QP problem as that in (5.7).

□

□ **End of chapter.**

Appendix D

Proof in Chapter 6

D.1 Proof of Theorem 9

Proof. 1. When $v \neq 1$, we denote the objective in (6.16) as $\mathcal{L}(\boldsymbol{\theta}) = \mathcal{D}(\boldsymbol{\theta}, \boldsymbol{\alpha}^\star) + \lambda \left(v\|\boldsymbol{\theta}\| + (1-v)\|\boldsymbol{\theta}\|_2^2\right)$. Since the objective function is continuous on $\boldsymbol{\theta}$, its minimizer (6.16) should satisfy

$$\frac{\partial \mathcal{L}}{\partial \theta_q} = -\frac{1}{2}(\boldsymbol{\alpha} \circ \mathbf{y})^\top \mathbf{K}_q (\boldsymbol{\alpha} \circ \mathbf{y}) + \lambda(v + 2(1-v)\theta_q) = 0.$$

As $\lambda > 0$, combining with $\boldsymbol{\theta} \geq \mathbf{0}$, we get $\theta_q^\star$ as (6.17). When $\mathbf{K}_i = \mathbf{K}_j$, we then have $\theta_i^\star = \theta_j^\star$.

2. When $v = 1$, the regularizer, $v\|\boldsymbol{\theta}\| + (1-v)\|\boldsymbol{\theta}\|_2^2$, reduces to lasso regularizer. It can be easily verified that both minimizers, $\boldsymbol{\theta}^\star$ and $\tilde{\boldsymbol{\theta}}$, achieve the same objective value.

□

D.2 Proof of Theorem 11

Proof. To prove Theorem 11, we first need the following proposition.

Proposition 1. *For any $\boldsymbol{\theta} \in \Theta$, we have*

1. $h^{t+1}(\boldsymbol{\theta}) \geq h^t(\boldsymbol{\theta})$, *and*
2. $h^t(\boldsymbol{\theta}) \leq \max_{\boldsymbol{\alpha}\in\mathcal{A}} \mathcal{D}(\boldsymbol{\theta}, \boldsymbol{\alpha})$.

The above two propositions can be easily checked by their definitions. They support the definition of the lower bound and the upper bound in (6.21).

Next, we have the following lemma indicating the relation between bounds:

Lemma 1. *Suppose we have a sequence of bounds,* $\{\underline{\mathcal{D}}^t\}_{t=1}^T$ *and* $\{\overline{\mathcal{D}}^t\}_{t=1}^T$*, defined in (6.21), we can obtain the following properties for their relation:*

1. $\underline{\mathcal{D}}^t \leq \mathcal{D}(\boldsymbol{\theta}^\star, \boldsymbol{\alpha}^\star) \leq \overline{\mathcal{D}}^t$,
2. $\overline{\mathcal{D}}^1 \geq \overline{\mathcal{D}}^2 \geq \ldots \geq \overline{\mathcal{D}}^t$, *and*
3. $\underline{\mathcal{D}}^1 \leq \underline{\mathcal{D}}^2 \leq \ldots \leq \underline{\mathcal{D}}^t$.

We now give a short proof of 1) in Lemma. For 2) and 3) of Lemma 1 can be easily verified based on the definitions.

First, Proposition 1 indicates that for any $\boldsymbol{\theta} \in \Theta$, $h^t(\boldsymbol{\theta}) \leq \max_{\boldsymbol{\alpha}\in\mathcal{A}} \mathcal{D}(\boldsymbol{\theta}, \boldsymbol{\alpha})$. Hence,

$$\underline{\mathcal{D}}^t = \min_{\boldsymbol{\theta}\in\Theta} h^t(\boldsymbol{\theta}) \leq \min_{\boldsymbol{\theta}\in\Theta} \max_{\boldsymbol{\alpha}\in\mathcal{A}} \mathcal{D}(\boldsymbol{\theta}, \boldsymbol{\alpha}) = \mathcal{D}(\boldsymbol{\theta}^\star, \boldsymbol{\alpha}^\star).$$

Second, since $\mathcal{D}(\boldsymbol{\theta}^t, \boldsymbol{\alpha}^t) = \max_{\boldsymbol{\alpha}\in\mathbf{A}} \mathcal{D}(\boldsymbol{\theta}^t, \boldsymbol{\alpha})$, then we have

$$\begin{aligned} \overline{\mathcal{D}}^t &= \min_{1\leq k\leq t} \mathcal{D}(\boldsymbol{\theta}^k, \boldsymbol{\alpha}^k) = \min_{\boldsymbol{\theta}\in\{\boldsymbol{\theta}_1,\ldots,\boldsymbol{\theta}_t\}} \max_{\boldsymbol{\alpha}\in\mathcal{A}} \mathcal{D}(\boldsymbol{\theta}, \boldsymbol{\alpha}) \\ &\geq \min_{\boldsymbol{\theta}\in\Theta} \max_{\boldsymbol{\alpha}\in\mathcal{A}} \mathcal{D}(\boldsymbol{\theta}, \boldsymbol{\alpha}) = \mathcal{D}(\boldsymbol{\theta}^\star, \boldsymbol{\alpha}^\star). \end{aligned}$$

The above two results conclude 1) of Lemma 1.

Hence, by applying 1) of Lemma 1, we can obtain 1) and 3) of Theorem 11. Combining 2) and 3) of Lemma 1, we have 2) of Theorem 11. □

D.3 Proof of Theorem 12

Proof. Before starting the proof, we first introduce the theorem:

Theorem 13 (Theorem 8.2.1 in Chapter 8, pp. 135–137 of [117])**.** *Let $\mathcal{D}$ be a convex and Lipschitz continuous function defined on the domain Θ of diameter $D(\Theta)$ with the Lipschitz constant being $L(\mathcal{D}) < \infty$. Applying the level method to this convex problem, the gap Δ^T converges to 0; or for any positive δ, one has*

$$T \geq c(\tau) \left(\frac{L(\mathcal{D})D(\Theta)}{\delta} \right)^2, \tag{D.1}$$

where $c(\tau) = \frac{1}{(1-\tau)^2\tau(2-\tau)}$.

We then can derive the result based on the above theorem.

First, let's define $\theta_{\max}$ be the maximum element value of $\boldsymbol{\theta}$. We then have $\theta_{\max} \leq 1$. It can be derived by

$$\begin{aligned} 1 &= v\|\boldsymbol{\theta}\|_1 + (1-v)\|\boldsymbol{\theta}\|_2^2, \quad \text{from Theorem 8} \\ &\geq v\theta_{\max} + (1-v)\theta_{\max}^2, \quad \text{by } \boldsymbol{\theta} \geq \mathbf{0}. \end{aligned} \tag{D.2}$$

The above inequality derives $\theta_{\max} \leq 1$, so as $\boldsymbol{\theta} \leq \mathbf{1}$.

Next, by applying $\boldsymbol{\theta} \leq \mathbf{1}$, we have

$$\boldsymbol{\theta}^\top \boldsymbol{\theta} \leq \mathbf{1}^\top \boldsymbol{\theta} = \|\boldsymbol{\theta}\|_1, \quad \forall \boldsymbol{\theta} \in \Theta. \tag{D.3}$$

Hence, $\forall \boldsymbol{\theta}, \boldsymbol{\theta}' \in \Theta$, we have

$$\begin{aligned} \|\boldsymbol{\theta} - \boldsymbol{\theta}'\|_2^2 \leq & \boldsymbol{\theta}^\top \boldsymbol{\theta} + \boldsymbol{\theta}'^\top \boldsymbol{\theta}' \\ = & v(\boldsymbol{\theta}^\top \boldsymbol{\theta} + \boldsymbol{\theta}'^\top \boldsymbol{\theta}') + (1-v)(\boldsymbol{\theta}^\top \boldsymbol{\theta} + \boldsymbol{\theta}'^\top \boldsymbol{\theta}') \\ \leq & v(\|\boldsymbol{\theta}\|_1 + \|\boldsymbol{\theta}'\|_1) + (1-v)(\|\boldsymbol{\theta}\|_2^2 + \|\boldsymbol{\theta}'\|_2^2) \\ = & 1 + 1 = 2\,. \end{aligned}$$

We then obtain the diameter $D(\Theta)$ as

$$D(\Theta) = \max_{\boldsymbol{\theta}, \boldsymbol{\theta}' \in \Theta} \|\boldsymbol{\theta} - \boldsymbol{\theta}'\|_2 = \sqrt{2}. \tag{D.4}$$

Further, the Lipschitz constant for the GMKL is

$$\begin{aligned} L_{\boldsymbol{\theta}}(\mathcal{D}) &= \max_{\boldsymbol{\theta}\in\Theta,\boldsymbol{\alpha}\in\mathcal{A}} \|\nabla\mathcal{D}_{\boldsymbol{\theta}}(\boldsymbol{\theta},\boldsymbol{\alpha})\|_2 \\ &= \max_{\boldsymbol{\alpha}\in\mathcal{A}} \|[\mathbf{V}_1,\ldots,\mathbf{V}_Q]^\top\|_2 \\ &\leq \frac{1}{2}NC^2\sqrt{Q}\max_{1\leq q\leq Q}\Lambda_{\max}(\mathbf{K}_q), \end{aligned} \tag{D.5}$$

where $\mathbf{V}_q = \frac{1}{2}(\boldsymbol{\alpha}\circ\mathbf{y})^\top\mathbf{K}_q(\boldsymbol{\alpha}\circ\mathbf{y})$, and $\Lambda_{\max}(\mathbf{K}_q)$ defines the maximum eigenvalue of the matrix $\mathbf{K}_q$.

Substituting (D.4) and (D.5) into (D.1) of Theorem 13, we can obtain the result as (6.26) and conclude the proof. □

□ **End of chapter.**

Bibliography

[1] D. A. Aaker, V. Kumar, and G. S. Day. *Marketing Research*. Wiley, 9th edition, 2006.

[2] Y. Amit, S. Shalev-Shwartz, and Y. Singer. Online classification for complex problems using simultaneous projections. In *NIPS*, pages 17–24, 2006.

[3] 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.

[4] A. Argyriou, T. Evgeniou, and M. Pontil. Multi-task feature learning. In *NIPS*, pages 41–48, 2006.

[5] A. Argyriou, T. Evgeniou, and M. Pontil. Convex multi-task feature learning. *Machine Learning*, 73(3):243–272, 2008.

[6] A. Argyriou, M. Herbster, and M. Pontil. Combining graph laplacians for semi–supervised learning. In Y. Weiss, B. Schölkopf, and J. Platt, editors, *Advances in Neural Information Processing Systems 18*, pages 67–74, Cambridge, MA, 2006. MIT Press.

[7] A. Astorino and A. Fuduli. Nonsmooth optimization techniques for semisupervised classification. *IEEE Trans. Pattern Anal. Mach. Intell.*, 29(12):2135–2142, 2007.

[8] F. Bach. Consistency of the group lasso and multiple kernel learning. *Journal of Machine Learning Research*, 9:1179–1225, 2008.

[9] F. R. Bach, G. R. G. Lanckriet, and M. I. Jordan. Multiple kernel learning, conic duality, and the SMO algorithm. In *ICML*, pages 41–48, New York, NY, USA, 2004. ACM.

[10] J. Bai, K. Zhou, G.-R. Xue, H. Zha, G. Sun, B. L. Tseng, Z. Zheng, and Y. Chang. Multi-task learning for learning to rank in web search. In *CIKM*, pages 1549–1552, 2009.

[11] B. Bakker and T. Heskes. Task clustering and gating for bayesian multitask learning. *Journal of Machine Learning Research*, 4:83–99, 2003.

[12] S. Balakrishnan and D. Madigan. Algorithms for sparse linear classifiers in the massive data setting. *Journal of Machine Learning Research*, 9:313–337, 2008.

[13] 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.

[14] S. Ben-David and R. Schuller. Exploiting task relatedness for mulitple task learning. In *COLT*, pages 567–580, 2003.

[15] K. P. Bennett and A. Demiriz. Semi-supervised support vector machines. In *Advances in Neural Information Processing Systems (NIPS)*, pages 1831–1850, 1998.

[16] D. P. Bertsekas. *Nonlinear Programming*. Athena Scientific, Belmont, Massachusetts, 2nd edition, 1999.

[17] C. Blake and C. Merz. UCI repository of machine learning databases, 1998. URL: http://archive.ics.uci.edu/ml/.

[18] L. Bottou. Stochastic learning. In O. Bousquet and U. von Luxburg, editors, *Advanced Lectures on Machine Learning*, Lecture Notes in Artificial Intelligence, LNAI 3176, pages 146–168. Springer Verlag, Berlin, 2004.

[19] L. Bottou and O. Bousquet. The tradeoffs of large scale learning. In *NIPS*, pages 161–168, 2007.

[20] L. Bottou and Y. LeCun. Large scale online learning. In *NIPS*, 2003.

[21] L. Bottou and C.-J. Lin. Support vector machine solvers. In L. Bottou, O. Chapelle, D. DeCoste, and J. Weston, editors, *Large Scale Kernel Machines*, pages 301–320. MIT Press, Cambridge, MA., 2007.

[22] S. Boyd and L. Vandenberghe. *Convex Optimization*. Cambridge University Press, 2004.

[23] E. J. Candès, X. Li, Y. Ma, and J. Wright. Robust principal component analysis? *CoRR*, abs/0912.3599, 2009.

[24] E. J. Candès and Y. Plan. Matrix completion with noise. *CoRR*, abs/0903.3131, 2009.

[25] E. J. Candès and B. Recht. Exact matrix completion via convex optimization. *CoRR*, abs/0805.4471, 2008.

[26] R. Caruana. Multitask learning. *Machine Learning*, 28(1):41–75, 1997.

[27] C.-C. Chang and C.-J. Lin. *LIBSVM: a library for support vector machines*, 2001. Software available at `http://www.csie.ntu.edu.tw/~cjlin/libsvm`.

[28] O. Chapelle, B. Schölkopf, and A. Zien, editors. *Semi-Supervised Learning*. MIT Press, Cambridge, MA, 2006.

[29] O. Chapelle and A. Zien. Semi-supervised classification by low density separation. In *the Tenth International Workshop on Artificial Intelligence and Statistics*, 2005.

[30] J. Chen, J. Liu, and J. Ye. Learning incoherent sparse and low-rank patterns from multiple tasks. In *KDD*, pages 1179–1188, 2010.

[31] J. Chen, L. Tang, J. Liu, and J. Ye. A convex formulation for learning shared structures from multiple tasks. In *ICML*, page 18, 2009.

[32] P.-H. Chen, R.-E. Fan, and C.-J. Lin. A study on SMO-type decomposition methods for support vector machines. *IEEE Transactions on Neural Networks*, 17:893–908, July 2006.

[33] P.-H. Chen, C.-J. Lin, and B. Schölkopf. A tutorial on ν-support vector machines. *Applied Stochastic Models in Business and Industry*, 21:111–136, 2005.

[34] S. Chen and D. L. Donoho. Basis pursuit. In *Proceedings of The Twenty-Eighth Asilomar Conference on Signals, Systems and Computers*, volume 1, pages 41–44, 1994.

[35] Y. Chen, M. R. Gupta, and B. Recht. Learning kernels from indefinite similarities. In *ICML*, page 19, 2009.

[36] R. Collobert, F. H. Sinz, J. Weston, and L. Bottou. Large scale transductive svms. *Journal of Machine Learning Research*, 7:1687–1712, 2006.

[37] R. Collobert, F. H. Sinz, J. Weston, and L. Bottou. Trading convexity for scalability. In *ICML*, pages 201–208, 2006.

[38] C. Cortes, M. Mohri, and A. Rostamizadeh. L2 regularization for learning kernels. In *Proceedings of the 25th*

Conference on Uncertainty in Artificial Intelligence (UAI 2009), 2009.

[39] 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.

[40] D. J. Crisp and C. J. C. Burges. A geometric interpretation of v-svm classifiers. In S. A. Solla, T. K. Leen, and K.-R. Müller, editors, *NIPS*, pages 244–250. The MIT Press, 1999.

[41] N. Cristianini and J. Shawe-Taylor. *An Introduction to Support Vector Machines and Other Kernel-based Learning Methods*. Cambridge University Press, 2000.

[42] N. Cristianini, J. Shawe-Taylor, A. Elisseeff, and J. S. Kandola. On kernel-target alignment. In *NIPS*, pages 367–373, 2001.

[43] N. Cristianini, J. Shawe-Taylor, A. Elisseeff, and J. S. Kandola. On kernel-target alignment. In *Neural Information Processing Systems (NIPS 13)*, pages 367–373, 2001.

[44] M. Culp and G. Michailidis. Graph-based semi-supervised learning. *IEEE Trans. Pattern Anal. Mach. Intell.*, 30(1):174–179, 2008.

[45] A. d'Aspremont, L. E. Ghaoui, M. I. Jordan, and G. R. G. Lanckriet. A direct formulation for sparse pca using semidefinite programming. In *NIPS*, 2004.

[46] O. Dekel, P. M. Long, and Y. Singer. Online multitask learning. In *COLT*, pages 453–467, 2006.

[47] P. S. Dhillon, B. Tomasik, D. P. Foster, and L. H. Ungar. Multi-task feature selection using the multiple inclusion criterion (MIC). In *ECML/PKDD*, pages 276–289, 2009.

[48] D. L. Donoho. Compressed sensing. *IEEE Transactions on Information Theory*, 52(4):1289–1306, 2006.

[49] D. L. Donoho and M. Elad. On the stability of the basis pursuit in the presence of noise. *Signal Processing*, 86(3):511–532, 2006.

[50] D. L. Donoho, M. Elad, and V. N. Temlyakov. Stable recovery of sparse overcomplete representations in the presence of noise. *IEEE Transactions on Information Theory*, 52(1):6–18, 2006.

[51] M. Dredze, K. Crammer, and F. Pereira. Confidence-weighted linear classification. In *ICML*, pages 264–271, 2008.

[52] J. Duchi and Y. Singer. Efficient learning using forward-backward splitting. *Journal of Machine Learning Research*, 10:2873–2898, 2009.

[53] R. O. Duda, P. E. Hart, and D. G. Stork. *Pattern Classification*. John Wiley & Sons, 2nd edition, 2000. 680 pages.

[54] B. Efron, T. Hastie, I. Johnstone, and R. Tibshirani. Least angle regression. *Annals of Statistics*, 32(2):407–499, 2004.

[55] T. Evgeniou, C. A. Micchelli, and M. Pontil. Learning multiple tasks with kernel methods. *Journal of Machine Learning Research*, 6:615–637, 2005.

[56] T. Evgeniou and M. Pontil. Regularized multi–task learning. In *KDD*, pages 109–117, 2004.

[57] M. Fink, S. Shalev-Shwartz, Y. Singer, and S. Ullman. Online multiclass learning by interclass hypothesis sharing. In *ICML*, pages 313–320, 2006.

[58] I. Frank and J. Friedman. A statistical view of some chemometrics regression tools (with discussion). *Technometrics*, 35:109–148, 1993.

[59] J. Friedman, T. Hastie, and R. Tibshirani. A note on the group lasso and a sparse group lasso, 2010.

[60] J. Friedman, T. Hastie, and R. Tibshirani. Regularized paths for generalized linear models via coordinate descent. *Journal of Statistical Software*, 33(1), 2010.

[61] D. Goldfarb and S. Liu. An o(n^3l) primal interior point algorithm for convex quadratic programming. *Math. Program.*, 49(3):325–340, 1991.

[62] M. Gönen and E. Alpaydin. Localized multiple kernel learning. In *ICML*, pages 352–359, 2008.

[63] Y. Han, F. Wu, J. Jia, Y. Zhuang, and B. Yu. Multi-task sparse discriminant analysis (mtsda) with overlapping categories. In *AAAI*, 2010.

[64] Z. Harchaoui and F. Bach. Image classification with segmentation graph kernels. In *CVPR*, 2007.

[65] W. Härdle, M. Müller, S. Sperlich, and A. Werwatz. *Nonparametric and Semiparametric Models*. Springer-Verlag Inc., 2004.

[66] D. J. Harrison and D. L. Rubinfeld. Hedonic housing prices and the demand for clean air. *Journal of Environmental Economics and Management*, 5(1):81–102, March 1978.

[67] T. Hastie, R. Tibshirani, and J. Friedman. *The Elements of Statistical Learning: Data Mining, Inference, and Prediction*. Springer, 2nd edition, Februray 2009.

[68] E. Hazan, A. Agarwal, and S. Kale. Logarithmic regret algorithms for online convex optimization. *Machine Learning*, 69(2-3):169–192, 2007.

[69] G. Hinton, P. Dayan, and M. Revow. Modeling the manifolds of images of handwritten digits. *IEEE Transactions on Neural Networks*, 8:65–74, 1997.

[70] B. K. P. Horn. *Robot Vision*. MIT Press/McGraw-Hill, Cambridge, MA, March 1986.

[71] D. W. Hosmer and S. Lemeshow. *Applied logistic regression*. Wiley-Interscience Publication, 2nd edition, 2000.

[72] C. Hu, J. Kwok, and W. Pan. Accelerated gradient methods for stochastic optimization and online learning. In *NIPS*, pages 781–789, 2009.

[73] M. Hu, Y. Chen, and J. T.-Y. Kwok. Building sparse multiple-kernel svm classifiers. *IEEE Transactions on Neural Networks*, 20(5):827–839, May 2009.

[74] K. Huang, Z. Xu, I. King, and M. R. Lyu. Semi-supervised learning from general unlabeled data. In *the IEEE International Conference on Data Mining, ICDM 2008*, pages 273–282, 2008.

[75] K. Huang, H. Yang, I. King, and M. R. Lyu. Imbalanced learning with biased minimax probability machine,. *IEEE Transactions on System, Man, and Cybernetics Part B*, 36:913–923, 2006. SCI(2006)=1.538.

[76] K. Huang, H. Yang, I. King, and M. R. Lyu. Maximizing sensitivity in medical diagnosis using biased minimax probability machine. *IEEE Transactions on Biomedical Engineering*, 53:821–831, 2006. SCI(2006)=2.302.

[77] K. Huang, H. Yang, I. King, and M. R. Lyu. Maxi-min margin machine: Learning large margin classifiers locally and globally. *IEEE Transactions on Neural Networks*, 19(2):260–272, February 2008. SCI(2007)=2.769.

[78] K. Huang, H. Yang, I. King, and M. R. Lyu. *Modeling Data Locally and Globally.* Advanced Topics in Science and Tecnology in China: Machine Learning. Zhejiang University Press with Springer Verlag, first edition, April 2008.

[79] J. J. Hull. A database for handwritten text recognition research. *IEEE Trans. Pattern Anal. Mach. Intell.*, 16(5):550–554, 1994.

[80] L. Jacob, G. Obozinski, and J.-P. Vert. Group lasso with overlap and graph lasso. In *ICML*, page 55, 2009.

[81] S. N. Jagarlapudi, D. Govindaraj, R. S, C. Bhattacharyya, A. Ben-Tal, and K. R. Ramakrishnan. On the algorithmics and applications of a mixed-norm based kernel learning formulation. In Y. Bengio, D. Schuurmans, J. Lafferty, C. K. I. Williams, and A. Culotta, editors, *Advances in Neural Information Processing Systems 22 (NIPS)*, pages 844–852, Vancouver, B.C., Canada, 2010.

[82] T. Jebara. Multi-task feature and kernel selection for svms. In *ICML*, 2004.

[83] S. Ji, L. T. Watson, and L. Carin. Semisupervised learning of hidden markov models via a homotopy method. *IEEE Trans. Pattern Anal. Mach. Intell.*, 31(2):275–287, 2009.

[84] T. Joachims. Transductive inference for text classification using support vector machines. In *ICML*, pages 200–209, Bled, Slowenien, 1999.

[85] T. Joachims. Optimizing search engines using clickthrough data. In *ACM SIGKDD Conference on Knowledge Discovery and Data Mining (KDD)*, pages 133–142, 2002.

[86] T. Joachims. Transductive learning via spectral graph partitioning. In *ICML*, pages 290–297, 2003.

[87] N. Jones and P. Pevzner. *An Introduction to Bioinformatics Algorithms*. MIT Press, Cambridge, MA, 2004.

[88] J. Kiefer and J. Wolfowitz. Stochastic estimation of the maximum of a regression function. *The Annuals of Mathematical Statistics*, 23:462–466, 1952.

[89] M. Kloft, U. Brefeld, P. Laskov, and S. Sonnenburg. Non-sparse multiple kernel learning. In *NIPS workshop on Kernel Learning: Automatic Selection of Optimal Kernels*, 2008.

[90] M. Kloft, U. Brefeld, S. Sonnenburg, P. Laskov, K.-R. Müller, and A. Zien. Efficient and accurate lp-norm multiple kernel learning. In Y. Bengio, D. Schuurmans, J. Lafferty, C. K. I. Williams, and A. Culotta, editors, *Advances in Neural Information Processing Systems 22 (NIPS)*, pages 997–1005, Vancouver, B.C., Canada, 2010.

[91] M. Kloft, U. Rückert, and P. L. Bartlett. A unifying view of multiple kernel learning. In *Proceedings of the European Conference on Machine Learning (ECML)*, 2010.

[92] K. Koh, S.-J. Kim, and S. Boyd. An interior-point method for large-scale l1-regularized logistic regression. *Journal of Machine Learning Research*, 8:1519–1555, 2007.

[93] D. Koller and N. Friedman. *Probabilistic Graphical Models: Principles and Techniques*. MIT Press, 2009.

[94] M. Kowalski, M. Szafranski, and L. Ralaivola. Multiple indefinite kernel learning with mixed norm regularization. In *ICML*, page 69, 2009.

[95] G. R. G. Lanckriet, T. D. Bie, N. Cristianini, M. I. Jordan, and W. S. Noble. A statistical framework for genomic data fusion. *Bioinformatics*, 20(16):2626–2635, 2004.

[96] G. R. G. Lanckriet, N. Cristianini, P. Bartlett, L. E. Ghaoui, and M. I. Jordan. Learning the kernel matrix with semidefinite programming. *Journal of Machine Learning Research*, 5:27–72, 2004.

[97] J. Langford, L. Li, and T. Zhang. Sparse online learning via truncated gradient. *Journal of Machine Learning Research*, 10:777–801, 2009.

[98] N. D. Lawrence and M. I. Jordan. Semi-supervised learning via gaussian processes. In L. K. Saul, Y. Weiss, and L. Bottou, editors, *Advances in Neural Information Processing Systems 17*, pages 753–760, Cambridge, MA, 2005. MIT Press.

[99] Y. Lecun, L. Bottou, Y. Bengio, and P. Haffner. Gradient-based learning applied to document recognition. In *Proceedings of the IEEE*, pages 2278–2324, november 1998.

[100] Y. LeCun and C. Cortes. The mnist database of handwritten digits.

[101] H. Lee, A. Battle, R. Raina, and A. Y. Ng. Efficient sparse coding algorithms. In *NIPS*, pages 801–808, 2006.

[102] P. J. Lenk, W. S. DeSarbo, P. E. Green, and M. R. Young. Hierarchical bayes conjoint analysis: Recovery of part-worth heterogeneity from reduced experimental designs. *Marketing Science*, 15(2):173–191, 1996.

[103] Y. Lin and H. Zhang. Component selection and smoothing in multivariate nonparametric regression. *Annals of Statistics*, 34:2272–2297, 2006.

[104] H. Liu, M. Palatucci, and J. Zhang. Blockwise coordinate descent procedures for the multi-task lasso, with applications to neural semantic basis discovery. In *ICML*, page 82, 2009.

[105] J. Liu, J. Chen, S. Chen, and J. Ye. Learning the optimal neighborhood kernel for classification. In *IJCAI*, pages 1144–1149, 2009.

[106] J. Liu, J. Chen, and J. Ye. Large-scale sparse logistic regression. In *KDD*, pages 547–556, 2009.

[107] J. Liu, S. Ji, and J. Ye. Multi-task feature learning via efficient $l_{2,1}$ norm minimization. In *UAI*, 2009.

[108] M. S. Lobo, M. Fazel, and S. Boyd. Portfolio optimization with linear and fixed transaction costs. *Annals OR*, 152(1):341–365, 2007.

[109] C. Longworth and M. J. F. Gales. Combining derivative and parametric kernels for speaker verification. *IEEE Transactions on Audio, Speech & Language Processing*, 17(4):748–757, 2009.

[110] J. Mairal, F. Bach, J. Ponce, and G. Sapiro. Online learning for matrix factorization and sparse coding. *Journal of Machine Learning Research*, 11:19–60, 2010.

[111] D. Matterra and S. Haykin. Support vector machines for dynamic reconstruction of a chaotic system. In B. Schölkopf, C. J. C. Burges, and A. J. Smola, editors, *Advances in Kernel Methods – Support Vector Learning*, pages 211–241. MIT Press, 1999.

[112] L. Meier, S. van de Geer, and P. Bühlmann. The group lasso for logistic regression. *Journal of the Royal Statistical Society, Series B*, 70(1):53–71, 2008.

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

[114] T. Mitchell. *Machine Learning*. McGraw Hill, 1997.

[115] M. Mojdeh and G. V. Cormack. Semi-supervised spam filtering: does it work? In *SIGIR*, pages 745–746, 2008.

[116] K. R. Müller, A. Smola, G. Rätsch, B. Schölkopf, J. Kohlmorgen, and V. Vapnik. Predicting Time Series with Support Vector Machines. In W. Gerstner, A. Germond, M. Hasler, and J. D. Nicoud, editors, *ICANN*, pages 999–1004. Springer, 1997.

[117] A. Nemirovski. Efficient methods in convex programming. Lecture Notes, 1994.

[118] Y. Nesterov. Primal-dual subgradient methods for convex problems. *Mathematical Programming*, 120(1):221–259, 2009.

[119] K. Nigam and R. Ghani. Analyzing the effectiveness and applicability of co-training. In *CIKM*, pages 86–93, 2000.

[120] K. Nigam, A. McCallum, S. Thrun, and T. M. Mitchell. Text classification from labeled and unlabeled documents using EM. *Machine Learning*, 39(2/3):103–134, 2000.

[121] G. Obozinski, B. Taskar, and M. I. Jordan. Joint covariate selection and joint subspace selection for multiple classification problems. *Statistics and Computing*, 2009.

[122] C. S. Ong, A. J. Smola, and R. C. Williamson. Learning the kernel with hyperkernels. *Journal of Machine Learning Research*, 6:1043–1071, 2005.

[123] N. Paragios, Y. Chen, and O. Faugeras. *Handbook of Mathematical Models in Computer Vision*. Springer, 2005.

[124] J. C. Platt. Fast training of support vector machines using sequential minimal optimization. In B. Schölkopf, C. J. C. Burges, and A. J. Smola, editors, *Advances in Kernel Methods – Support Vector Learning*. MIT Press, 1999.

[125] J. C. Platt. Probabilistic outputs for support vector machines and comparisons to regularized likelihood methods. In *Advances in Large Margin Classifiers*, pages 61–74. MIT Press, 1999.

[126] T. K. Pong, P. Tseng, S. Ji, and J. Ye. Trace norm regularization: Reformulations, algorithms, and multi-task learning. *SIAM Journal on Optimization*, 2010.

[127] A. Quattoni, X. Carreras, M. Collins, and T. Darrell. An efficient projection for $l_{1,\infty}$ regularization. In *ICML*, page 108, 2009.

[128] F. Radlinski and T. Joachims. Query chains: Learning to rank from implicit feedback. In *ACM SIGKDD International Conference On Knowledge Discovery and Data Mining (KDD)*, pages 239–248, 2005.

[129] R. Raina, A. Battle, H. Lee, B. Packer, and A. Y. Ng. Self-taught learning: transfer learning from unlabeled data. In *ICML*, pages 759–766, 2007.

[130] A. Rakotomamonjy, F. R. Bach, S. Canu, and Y. Grandvalet. SimpleMKL. *Journal of Machine Learning Research*, 9:1179–1225, 2008.

[131] S. Raman, T. J. Fuchs, P. J. Wild, E. Dahl, and V. Roth. The bayesian group-lasso for analyzing contingency tables. In *ICML*, page 111, 2009.

[132] C. E. Rasmussen and C. K. I. Williams. *Gaussian Processes for Machine Learning*. The MIT Press, 2006.

[133] H. Robbins and S. Monro. A stochastic approximation model. *The Annuals of Mathematical Statistics*, 22:400–407, 1951.

[134] V. Roth and B. Fischer. The group-lasso for generalized linear models: uniqueness of solutions and efficient algorithms. In *ICML*, pages 848–855, 2008.

[135] K. Scheinberg. An efficient implementation of an active set method for svms. *Journal of Machine Learning Research*, 7:2237–2257, 2006.

[136] B. Schölkopf, J. C. Platt, J. Shawe-Taylor, A. J. Smola, and R. C. Williamson. Estimating the support of a high-dimensional distribution. *Neural Computation*, 13(7):1443–1471, 2001.

[137] B. Schölkopf and A. Smola. *Learning with Kernels*. MIT Press, Cambridge, MA, 2002.

[138] B. Schölkopf, A. J. Smola, R. C. Williamson, and P. L. Bartlett. New support vector algorithms. *Neural Computation*, 12(5):1207–1245, 2000.

[139] B. Schölkopf, R. C. Williamson, A. J. Smola, J. Shawe-Taylor, and J. C. Platt. Support vector method for novelty detection. In *NIPS*, pages 582–588, 1999.

[140] S. Shalev-Shwartz and S. M. Kakade. Mind the duality gap: Logarithmic regret algorithms for online optimization. In *NIPS*, pages 1457–1464, 2008.

[141] S. Shalev-Shwartz and Y. Singer. Online learning meets optimization in the dual. In *COLT*, pages 423–437, 2006.

[142] S. Shalev-Shwartz and N. Srebro. SVM optimization: inverse dependence on training set size. In *ICML*, pages 928–935, 2008.

[143] J. Shawe-Taylor. Kernel learning for novelty detection. In *NIPS workshop on Kernel Learning: Automatic Selection of Optimal Kernels*, 2008.

[144] J. Shawe-Taylor and N. Cristianini. *Kernel Methods for Pattern Analysis*. Cambridge University Press, UK, 2004.

[145] A. Singh, R. D. Nowak, and X. Zhu. Unlabeled data: Now it helps, now it doesn't. In *NIPS*, pages 1513–1520, 2008.

[146] F. H. Sinz, O. Chapelle, A. Agarwal, and B. Schölkopf. An analysis of inference with the universum. In J. C. Platt, D. Koller, Y. Singer, and S. Roweis, editors, *Advances in Neural Information Processing Systems 20*, pages 1369–1376, Cambridge, MA, 2008. MIT Press.

[147] A. J. Smola and B. Schölkopf. A tutorial on support vector regression. *Statistics and Computing*, 14(3):199–222, 2004.

[148] A. J. Smola, S. V. N. Vishwanathan, and T. Hofmann. Kernel methods for missing variables. In *Proceedings of the tenth International Workshop on Artificial Intelligence and Statistics*, 2005.

[149] A. J. Smola, S. V. N. Vishwanathan, and Q. V. Le. Bundle methods for machine learning. In *NIPS*, 2007.

[150] S. Sonnenburg, G. Rätsch, and C. Schäfer. Learning interpretable SVMs for biological sequence classification. In *Research in Computational Mloecular Biology, LNBI 3500*, pages 389–407. Springer-Verlag Berlin Heidelberg, 2005.

[151] S. Sonnenburg, G. Rätsch, C. Schäfer, and B. Schölkopf. Large scale multiple kernel learning. *Journal of Machine Learning Research*, 7:1531–1565, 2006.

[152] I. Steinwart and A. Christmann. *Support Vector Machines*. Springer, New York, 2008. 602 pages.

[153] M. Szafranski, Y. Grandvalet, and A. Rakotomamonjy. Composite kernel learning. In *ICML '08: Proceedings of the 25th international conference on Machine learning*, pages 1040–1047, New York, NY, USA, 2008. ACM.

[154] D. M. J. Tax and R. P. W. Duin. Support vector domain description. *Pattern Recognition Letters*, 20(11-13):1191–1199, 1999.

[155] R. Tibshirani. Regression shrinkage and selection via the lasso. *J. Roy. Statist. Soc. Ser. B*, 58(1):267–288, 1996.

[156] R. Tibshirani, M. Saunders, S. Rosset, J. Zhu, and K. Knight. Sparsity and smoothness via the fused lasso. *Journal of the Royal Statistical Society, Series B*, 67(1):91–108, 2005.

[157] R. Tomioka and T. Suzuki. Sparsity-accuracy trade-off in MKL. arXiv:1001.2615, 2010.

[158] I. W.-H. Tsang and J. T.-Y. Kwok. Efficient hyperkernel learning using second-order cone programming. *IEEE Transactions on Neural Networks*, 17(1):48–58, 2006.

[159] A. N. Tychonoff and V. Y. Arsenin. *Solution of Ill-posed Problems*. Winston & Sons, Washington, 1977.

[160] V. Vapnik. *The Nature of Statistical Learning Theory*. Springer, New York, 2nd edition, 1999.

[161] V. Vapnik and S. Kotz. *Estimation of Dependences Based on Empirical Data: Empirical Inference Science (Information Science and Statistics)*. Springer-Verlag New York, Inc., Secaucus, NJ, USA, 2nd edition, 2006.

[162] M. Varma and B. R. Babu. More generality in efficient multiple kernel learning. In A. P. Danyluk, L. Bottou, and M. L. Littman, editors, *ICML*, volume 382 of *ACM International Conference Proceeding Series*, page 134. ACM, 2009.

[163] J. von Neumann and O. Morgenstern. *Theory of Games and Economic Behavior*. Princeton University Press, 3rd edition, 1953. First published in 1944.

[164] J. Wang, X. Shen, and W. Pan. On efficient large margin semisupervised learning: Method and theory. *Journal of the Royal Statistical Society, Series B*, 10(Mar):719–742, 2009.

[165] J. Weston, R. Collobert, F. H. Sinz, L. Bottou, and V. Vapnik. Inference with the universum. In *ICML*, pages 1009–1016, 2006.

[166] J. Weston, A. Elisseeff, B. Schölkopf, and M. Tipping. Use of the zero-norm with linear models and kernel methods. *Journal of Machine Learning Research*, 3:1439–1461, 03 2003.

[167] L. A. Wolsey. *Integer programming*. Wiley-Interscience, 1 edition, September 1998.

[168] T. T. Wu, Y. F. Chen, T. Hastie, E. Sobel, and K. Lange. Genomewide association analysis by lasso penalized logistic regression. *Bioinformatics*, 25(6):714–721, 2009.

[169] L. Xiao. Dual averaging method for regularized stochastic learning and online optimization. *Journal of Machine Learning Research*, 11:2543–2596, October 2010.

[170] Z. Xu, R. Jin, I. King, and M. R. Lyu. An extended level method for efficient multiple kernel learning. In D. Koller, D. Schuurmans, Y. Bengio, and L. Bottou, editors, *Advances in Neural Information Processing Systems 21 (NIPS)*, pages 1825–1832, 2009.

[171] Z. Xu, R. Jin, H. Yang, I. King, and M. R. Lyu. Simple and efficient multiple kernel learning by group lasso. In *ICML*, Haifa, Israel, 2010.

[172] Z. Xu, R. Jin, J. Ye, H. Yang, I. King, and M. R. Lyu. Non-monotonic feature selection, 2009.

[173] H. Yang. Multi-task learning with irrelevant tasks. Technical report, The Chinese University of Hong Kong, 2009.

[174] H. Yang, L. Chan, and I. King. Support vector machine regression for volatile stock market prediction. In H. Yin, N. M. Allinson, R. T. Freeman, J. A. Keane, and S. J. Hubbard, editors, *IDEAL*, volume 2412 of *Lecture Notes in Computer Science*, pages 391–396. Springer, 2002.

[175] H. Yang, K. Huang, I. King, and M. R. Lyu. Efficient minimax clustering probability machine by generalized probability product kernel. In J. M. Zurada, G. G. Yen, and J. Wang, editors, *Proceedings to the World Congress on Computational Intelligence (WCCI2008)*, Hong Kong, June 2-6 2008. IEEE.

[176] H. Yang, K. Huang, I. King, and M. R. Lyu. Localized support vector regression for time series prediction. *Neurocomputing*, 72(10-12):2659–2669, 2009.

[177] H. Yang and I. King. Sprinkled latent semantic indexing for text classification with background knowledge. In *Proceedings to the 15th International Conference on Neural Information Processing (ICONIP'08)*, Auckland, New Zealand, November 25–28 2008.

[178] H. Yang and I. King. Ensemble learning for imbalanced e-commerce transaction anomaly classification. In *Proceedings to the 16th International Conference on Neural Information Processing (ICONIP'09)*, Bangkok, Thailand, 2009.

[179] H. Yang, I. King, and M. R. Lyu. Multi-task learning for one-class classification. In *IJCNN*, Barcelona, Spain, 2010.

[180] H. Yang, I. King, and M. R. Lyu. Online learning for multi-task feature selection. In *Proceedings of the 19th ACM Conference on Information and Knowledge Management (CIKM2010)*, pages 1693–1696, Toronto, Canada, 2010.

[181] H. Yang, Z. Xu, I. King, and M. R. Lyu. Online learning for group lasso. In *Proceedings of the 27th International Conference on Machine Learning (ICML2010)*, Haifa, Israel, 2010.

[182] J. Ye, J. Chen, and S. Ji. Discriminant kernel and regularization parameter learning via semidefinite programming. In *ICML*, pages 1095–1102, 2007.

[183] G. W. Yeo and C. B. Burge. Maximum entropy modeling of short sequence motifs with applications to rna splicing signals. *Journal of Computational Biology*, 11(2/3):377–394, 2004.

[184] M. Yuan and Y. Lin. Model selection and estimation in regression with grouped variables. *Journal of the Royal Statistical Society, Series B*, 68(1):49–67, 2006.

[185] A. L. Yuille and A. Rangarajan. The concave-convex procedure. *Neural Computation*, 15(4):915–936, 2003.

[186] R. Zass and A. Shashua. Nonnegative sparse pca. In *NIPS*, pages 1561–1568, 2006.

[187] D. Zhang, J. Wang, F. Wang, and C. Zhang. Semi-supervised classification with universum. In *SDM*, pages 323–333, 2008.

[188] T. Zhang. Solving large scale linear prediction problems using stochastic gradient descent algorithms. In *ICML*, Banff, Alberta, Canada, 2004.

[189] Y. Zhang. Multi-task active learning with output constraints. In *AAAI*, 2010.

[190] P. Zhao, S. C. Hoi, and R. Jin. Duol: A double updating approach for online learning. In *NIPS*, pages 2259–2267, 2009.

[191] P. Zhao, G. Rocha, and B. Yu. Grouped and hierarchical model selection through composite absolute penalties. *Annals of Statistics*, 37(6A):3468–3497, 2009.

[192] P. Zhong and M. Fukushima. A new multi-class support vector algorithm. *Optimization Methods and Software*, 21:359–372, 2006.

[193] D. Zhou, O. Bousquet, T. N. Lal, J. Weston, and B. Schölkopf. Learning with local and global consistency. In *NIPS*, 2003.

[194] Y. Zhou, R. Jin, and S. C. Hoi. Exclusive lasso for multi-task feature selection. In *AISTATS*, 2010.

[195] X. Zhu. Semi-supervised learning literature survey. Technical Report 1530, Department of Computer Sciences, University of Wisconsin, Madison, 2005.

[196] X. Zhu, Z. Ghahramani, and J. D. Lafferty. Semi-supervised learning using gaussian fields and harmonic functions. In *ICML*, pages 912–919, 2003.

[197] X. Zhu and A. B. Goldberg. *Introduction to Semi-Supervised Learning*. Morgan & Claypool, 2009.

[198] A. Zien and C. S. Ong. Multiclass multiple kernel learning. In *ICML '07: Proceedings of the 24th international conference on Machine learning*, pages 1191–1198, New York, NY, USA, 2007. ACM.

[199] M. Zinkevich. Online convex programming and generalized infinitesimal gradient ascent. In *ICML*, pages 928–936, 2003.

[200] H. Zou and T. Hastie. Regularization and variable selection via the elastic net. *Journal of the Royal Statistical, Society B*, 67:301–320, 2005.