2014

# On Kernel-base Multi-Task Learning

Cong Li
*University of Central Florida*

# ON KERNEL-BASED MULTI-TASK LEARNING

by

CONG LI
B.S. Tianjin University, 2009
M.S. University of Central Florida, 2014

A dissertation submitted in partial fulfilment of the requirements
for the degree of Doctor of Philosophy
in the Department of Electrical Engineering and Computer Science
in the College of Engineering and Computer Science
at the University of Central Florida
Orlando, Florida

Fall Term
2014

Major Professor: Michael Georgiopoulos, Georgios C. Anagnostopoulos

# ABSTRACT

Multi-Task Learning (MTL) has been an active research area in machine learning for two decades. By training multiple relevant tasks simultaneously with information shared across tasks, it is possible to improve the generalization performance of each task, compared to training each individual task independently. During the past decade, most MTL research has been based on the Regularization-Loss framework due to its flexibility in specifying various types of information sharing strategies, the opportunity it offers to yield a kernel-based methods and its capability in promoting sparse feature representations.

However, certain limitations exist in both theoretical and practical aspects of Regularization-Loss-based MTL. Theoretically, previous research on generalization bounds in connection to MTL Hypothesis Space (HS)s, where data of all tasks are pre-processed by a (partially) common operator, has been limited in two aspects: First, all previous works assumed linearity of the operator, therefore completely excluding kernel-based MTL HSs, for which the operator is potentially non-linear. Secondly, all previous works, rather unnecessarily, assumed that all the task weights to be constrained within norm-balls, whose radii are equal. The requirement of equal radii leads to significant inflexibility of the relevant HSs, which may cause the generalization performance of the corresponding MTL models to deteriorate. Practically, various algorithms have been developed for kernel-based MTL models, due to different characteristics of the formulations. Most of these algorithms are a burden to develop and end up being quite sophisticated, so that practitioners may face a hard task in interpreting and implementing them, especially when multiple models are involved. This is even more so, when Multi-Task Multiple Kernel Learning (MT-MKL) models are considered.

This research largely resolves the above limitations. Theoretically, a pair of new kernel-based

HSs are proposed: one for single-kernel MTL, and another one for MT-MKL. Unlike previous works, we allow each task weight to be constrained within a norm-ball, whose radius is learned during training. By deriving and analyzing the generalization bounds of these two HSs, we show that, indeed, such a flexibility leads to much tighter generalization bounds, which often results to significantly better generalization performance. Based on this observation, a pair of new models is developed, one for each case: single-kernel MTL, and another one for MT-MKL. From a practical perspective, we propose a general MT-MKL framework that covers most of the prominent MT-MKL approaches, including our new MT-MKL formulation. Then, a general purpose algorithm is developed to solve the framework, which can also be employed for training all other models subsumed by this framework. A series of experiments is conducted to assess the merits of the proposed mode when trained by the new algorithm. Certain properties of our HSs and formulations are demonstrated, and the advantage of our model in terms of classification accuracy is shown via these experiments.

To my beloved parents

# ACKNOWLEDGMENTS

I would like to thank my advisors, Prof. Michael Georgiopoulos and Dr. Georgios Anagnostopoulos, for spending an enormous amount of time and efforts in advising and guiding my doctoral research. With tireless discussion and encouragement, I was able to push my limits and attain fruitful research outcomes, which I would have not imagined to be able to achieve without their tremendous efforts.

I would also like to thank my committee members, Dr. Haiyan Hu, Dr. Marshall Tappen and Dr. Liqiang Ni, who have devoted time to evaluate and assess my research works, and provide helpful suggestions during my proposal examination. Moreover, I appreciate the help of all Machine Learning Lab members for their constructive comments in the preparation of my dissertation defense.

Finally, I would like to thank my parents for their unconditional love and support they showed me, both during my Ph.D. endeavor and other tough periods that I went through.

# TABLE OF CONTENTS

# LIST OF FIGURES

# LIST OF TABLES

# CHAPTER 1: INTRODUCTION

Multi-Task Learning (MTL) [17] [16] is a machine learning paradigm that, instead of treating each prediction task independently, trains multiple related tasks simultaneously with information of each task being shared with the other ones. The hope is that, such an information sharing strategy can serve as positive inductive bias, so that relevant tasks are benefited from the information that is shared by the other tasks. As a result, the training can be conducted more efficiently, which potentially leads to better generalization performance of each task, compared to the traditional single-task learning paradigm.

After near two decades' active research, MTL has received a fruitful achievements. Specifically, it has been shown that MTL is beneficial in many real world application problems, as better generalization performance can be achieved via MTL. For example, MTL has been successfully applied in medical diagnosis [105] [10] [11] [77] [116], computer vision [94] [107] [58], web search ranking [19], brain-computer interface [3], stock selection [34], and bioinformatics [63], to name only a few.

Usually formulated as minimization problems with Regularization-Loss objective function, the benefits of such MTL methods are obvious: First, it is straightforward to encourage information sharing amongst tasks by designing appropriate regularizer or optimization constraint. Second, the regularizer can be flexibly selected so that other useful characteristics can be added into the model; for example, certain MTL regularizers are able to encourage sparse feature selection, when multiple tasks are learned. Third, thank to the representer theorem [84], most linear models can enjoy nonlinearity by incorporating the kernel method [86]. As a result, a better generalization performance can be achieved, compared to pure linear models. This results in the research specifically on kernel-based MTL and, even more sophisticated, Multi-Task Multiple Kernel Learn-

ing (MT-MKL), when Multiple Kernel Learning (MKL) [55] is incorporated with MTL to facilitate kernel selection. Last but not least, it is straightforward to conduct theoretical studies, in terms of generalization bound. This theoretical study is crucial, since it helps people understand *why* MTL is advantageous and *when* we should apply MTL.

However, although having been actively researched recently, certain limitations exist in previous research works, in both theoretical and practical perspectives.

Theoretically, the limitations lie within the research of the generalization bound of MTL models. To be concrete, it is a commonly utilized technique to conducting various types of regularizer for information sharing amongst tasks, as mentioned before. Previously, while most research on generalization bound of MTL Hypothesis Space (HS)s focused on this category [47] [68] [75], another type of information sharing strategy has been largely omitted, which applies a common or partially common pre-processor to map data from all tasks to a (partially) common feature space, and the tasks are learned in the feature space. Such an important technique is not only used in subspace learning [6] [48], but is also naturally adapted by kernel-based MTL and MT-MKL with a (partially) common kernel function used for all tasks [91] [45] [57]. In this case, the data from all tasks are pre-processed by a (partially) common feature map that is specified by the kernel function. It is crucial to conduct a thorough theoretical study on these MTL HSs, as it not only leads to a better understanding of the corresponding MTL methods, but also may inspire the invention of better MTL models.

However, for this category of MTL HSs, the corresponding generalization bound is only studied in two papers [65] [4] with very limited scenario. Take [65] as an example, the authors studied the following HS:

$$\mathcal{F} = \{\boldsymbol{x} \mapsto [\langle \boldsymbol{w}_1, A\boldsymbol{x}\rangle, \cdots, \langle \boldsymbol{w}_T, A\boldsymbol{x}\rangle]' : \|\boldsymbol{w}_t\|^2 \leq R, t = 1, \cdots, T, A \in \mathcal{A}\} \tag{1.1}$$

assuming the $t$-th task features a linear function $\langle \boldsymbol{w}_t, A\boldsymbol{x} \rangle$, and $\mathcal{A}$ is a set of bounded self-adjoint linear operators. The limitation of this work is two-fold. First, the HS does not include the MTL models where a common kernel function is learned for all tasks, such as the MT-MKL models. This is because the authors limited the feature mapping $A$ to be a linear operator, while the feature mapping associated to the kernel function is not necessarily linear. Second, the authors unnecessarily set all the task weights to be constrained in norm balls with the same radius. However, letting each norm ball constraint has its own radius yields a more flexible hypothesis space, and, in practice, may lead to an improved generalization bound, which potentially results in better performance of the corresponding model. Similar limitations also exist in [4].

Practically, certain limitations of previous research on MTL exist in terms of development and application of MTL algorithms. First, since there is no general purpose algorithm for a spectrum of MTL methods, when a new MTL model is developed, it is usually inevitable to invent an algorithm specifically for the proposed model. In fact, this is a common practice for most previous research works. However, developing an algorithm for MTL models is usually a non-trivial task. This is even more true when kernel-based MTL and MT-MKL are considered. As a result, the requirement of model-specific algorithm has become an extra burden for researchers to invent novel MTL models. Second, since most MTL models are accompanied with model-specific algorithms, it is difficult for practitioners to compare a set of MTL methods, as several complicated algorithms have to be interpreted and implemented.

This research is aimed to address the above mentioned limitations, with emphasis on kernel-based MTL and MT-MKL. Specifically, the contributions of this research are as follows:

- We propose a more general MTL HS, where data from all tasks are mapped to a (partially) common feature space. Such a HS embraces kernel-based MTL and MT-MKL, therefore is much more general than the previous two works. Besides, in the proposed HS, each task

weight is constrained in a norm ball, whose radius is task-specific and learned during the training stage, therefore is more flexible than the previous ones, as described above.

- We derive Rademacher complexity-based generalization bound for the proposed HS, which appears to be tighter than the ones that are proposed in [65] [4], due to the introduction of task-specific radius for the norm-ball constraints. Such results justify the invention of our HS. These two contributions addressed the above mentioned theoretical limitation of MTL research.

- We introduce a new MTL model, in both single kernel and MKL scenario, based on the proposed HS. With the tighter generalization bound, it is expected that the new model is able to achieve a better generalization performance.

- We develop a general framework for MTL, which covers not only our proposed MTL model, but also most existing MT-MKL and single-task MKL methods.

- We invent an easy to implement general purpose algorithm that is able to solve the proposed MTL framework, therefore not only solves our MTL model, but also all the kernel-based methods that are covered by the framework. Such an algorithm addressed the above mentioned practical limitation of MTL research.

The rest of this dissertation is organized as follows: In Chapter 2, we provide an overview of a series of preliminary knowledge, including the concept of Rademacher complexity-based generalization bound, Support Vector Machine (SVM), MKL and MTL. Chapter 3 discusses the literature review associated with MTL generalization bounds, models, and algorithms. Then, in Chapter 4, the proposed HS is introduced, followed by the derivation of the generalization bound. A thorough analysis to the bound is included, which demonstrates the advantages of the proposed HS. Besides, due to the superiority of the HS, a kernel-based MTL model is introduced based on the HS.

Such a model is optimized by the algorithm that is developed in Chapter 5, which also solves all kernel-based MTL models that are covered by the framework that is proposed in the same chapter. The experimental evaluation of all our works is given in Chapter 6. Conclusions and summary are provided in Chapter 7.

# CHAPTER 2: PRELIMINARIES

## Rademacher Complexity-based Generalization Bound

### *Generalization Bound*

Given some prior knowledge for a certain task, a typical machine learning method is aimed to learn useful information from the knowledge, so that, based on what it is learned, it generalizes well in the future, when performing the same task. In this dissertation, we specifically consider the supervised learning paradigm, where the prior knowledge is given in the form of a set of training data $\{(\boldsymbol{x}_i, y_i)\}, i = 1, \cdots, N$, which are drawn identically and independently from an unknown joint probability distribution $P(X, Y)$ with $X \in \mathcal{X}, Y \in \mathcal{Y}$. Here, the output $y_i$ of a given input $\boldsymbol{x}_i$ is usually referred to as *label*. Given the training data, during the training stage, the machine learning algorithm seeks a function $f : \mathcal{X} \to \mathcal{Y}$ from a fixed set of functions $\mathcal{F}$, which is called *Hypothesis Set* or *Hypothesis Space* (HS). The learned function $f$ is expected to generalize well; this means that, we expect $f(\boldsymbol{x}) = y$ for most (hopefully all) future $(\boldsymbol{x}, y)$'s that are drawn identically and independently from $P(X, Y)$.

As a concrete example, consider the task of handwritten digits recognition, where the machine learning method is aimed to learn how to distinguish the $10$ different handwritten digits, namely, from $0$ to $9$. Assuming each digit is represented by a grayscale image of size $h$-by-$w$ pixels, then we can construct a vector $\boldsymbol{x}_i$ for each image $i = 1, \cdots, N$, by concatenating the pixel values into a $h * w$-dimensional vector. Therefore, the training data set can be comprised with a collection of pairs $(\boldsymbol{x}_i, y_i) \in \mathcal{X} \times \mathcal{Y}, i = 1, \cdots, N$, which are assumed to be drawn identically and independently from an unknown probability distribution $P(X, Y)$, and $\mathcal{X} \subseteq \mathbb{R}^{h*w}, \mathcal{Y} = \{0, \cdots, 9\}$. Based on this training set, in the training stage, the machine learning method seeks a function $f : \mathcal{X} \to \mathcal{Y}$

from the HS $\mathcal{F}$, which is designed a priori. The selected function $f$ is expected to correctly assign label to the most, hopefully all, of the future handwritten digits.

For different machine learning methods, the HS $\mathcal{F}$ may be differently defined, and various algorithms can be specified to select the function $f$ from $\mathcal{F}$. For each method that learns $f$ from $\mathcal{F}$, one fundamental question that one should ask is that, how well will the learned function $f$ generalize to future data? In other words, we are interested in the *expected generalization error* for a particular function $f$, or *expected error* for short, as defined below:

$$er(f) \triangleq E_{(X,Y)}\{[f(X) \neq Y]\} \tag{2.1}$$

where $(X, Y) \sim P(X, Y)$, $[f(X) \neq Y] = 1$ if $f(X) \neq Y$ and 0 otherwise. The generalization error tells us that, on average, how much the mis-prediction rate is for a given prediction function $f$ and probability distribution $P(X, Y)$. However, since $P(X, Y)$ is unknown, it is impossible to calculate $er(f)$ directly. Instead, it is a widely accepted approach to estimate it with an upper bound, namely the *generalization bound*.

In this section, we give an introduction to Rademacher Complexity-based generalization bound [8], which is widely utilized in many research areas, such as kernel learning [26], metric learning [14], dictionary learning [67], MTL [65], etc. This section focuses on the single-task binary classification setting, which is further extended to MTL and MT-MKL later.

*Rademacher Complexity-based Generalization Bound for Binary Classification Problems*

This dissertation specifically focuses on binary classification problems, namely $\mathcal{Y} = \{1, -1\}$. Given a function $f : \mathcal{X} \to \mathbb{R}$, usually, a datum $\boldsymbol{x}$ is considered to be classified as class 1 if

$f(\boldsymbol{x}) > 0$, or class $-1$ if $f(\boldsymbol{x}) < 0$. For the marginal case $f(\boldsymbol{x}) = 0$ (although rarely happens), it is always considered as misclassification. In other words, a training data $(\boldsymbol{x}_i, y_i)$ is correctly classified when $y_i f(\boldsymbol{x}_i) > 0$. Therefore, for binary classification, the expected error can be written as

$$er(f) \triangleq E_{(X,Y)}\{[Yf(X) \leq 0]\} = E_{(X,Y)}\{\mathbf{1}_{(-\infty,0]}(Yf(X))\} \tag{2.2}$$

where $\mathbf{1}_{(-\infty,0]}(\cdot)$ is the characteristic function of $(-\infty, 0]$ and referred to as the $0/1$ *loss function*. One drawback of the $0/1$ loss function is that, it is non-convex and non-differentiable, thus a model that directly utilizes the $0/1$ loss is usually difficult to optimize. Therefore, it is common to introduce a surrogate loss function $L$, which upper bounds the $0/1$ loss, in order to measure the loss induced by misclassification. Usually, $L$ is selected to be convex and (or) differentiable, and should return $0$ for a correct classification, and a positive value for a misclassification. Based on the training set $\{\boldsymbol{x}_i, y_i\}, i = 1, \cdots, N$ and the surrogate loss function $L$, the *empirical loss* is defined as follows:

$$\hat{er}_L(f) \triangleq \frac{1}{N}\sum_{i=1}^{N} L(y_i f(\boldsymbol{x}_i)) \tag{2.3}$$

It calculates the average loss induced by the function $f$ on the training set, and is directly accessible after training, as it does not depend on the unknown probability distribution $P(X, Y)$. Therefore, it is natural to estimate the generalization error $er(f)$ with the empirical loss.

Prior to discussing the generalization bound for binary classification problems, we give a theorem below, which serves as a foundation to the derivation of the generalization bounds for binary classification problems.

**Theorem 1** (Theorem 3.1, [71]). *Let $\mathcal{G}$ be a family of functions mapping from an arbitrary space $\mathcal{Z}$ to $[0,1]$, and $\{\boldsymbol{z}_i\}, \boldsymbol{z}_i \in \mathcal{Z}, i = 1, \cdots, N$ be a collection of data that are drawn identically and independently from the probability distribution $P(Z)$. Then, for any $\delta > 0$, with probability at least $1-\delta$, each of the following holds for all $g \in \mathcal{G}$:*

$$E\{g(Z)\} \leq \frac{1}{N}\sum_{i=1}^{N} g(\boldsymbol{z}_i) + \hat{R}(\mathcal{G}) + \sqrt{\frac{9\ln\frac{1}{\delta}}{2N}} \tag{2.4}$$

$$E\{g(Z)\} \leq \frac{1}{N}\sum_{i=1}^{N} g(\boldsymbol{z}_i) + R(\mathcal{G}) + \sqrt{\frac{\ln\frac{1}{\delta}}{2N}} \tag{2.5}$$

*where $Z$ is a random variable with probability distribution $P(Z)$.*

In the above theorem, the quantities $R(\mathcal{Z})$ and $\hat{R}(\mathcal{Z})$ are called the ***Rademacher Complexity (RC)*** and ***Empirical Rademacher Complexity (ERC)*** respectively, which are defined as follows:

$$\hat{R}(\mathcal{G}) \triangleq E_\sigma\{\sup_{g\in\mathcal{G}} \frac{2}{N}\sum_{i=1}^{N} \sigma_i g(\boldsymbol{z}_i)\} \tag{2.6}$$

$$R(\mathcal{G}) \triangleq E_{\boldsymbol{z}} E_\sigma\{\sup_{g\in\mathcal{G}} \frac{2}{N}\sum_{i=1}^{N} \sigma_i g(\boldsymbol{z}_i)\} = E_{\boldsymbol{z}}\{\hat{R}(\mathcal{G})\} \tag{2.7}$$

where $\sigma_i$'s are *i.i.d.* Rademacher-distributed (*i.e.*, Bernoulli $(\frac{1}{2})$-distributed random variables with sample space $\{+1, -1\}$), and $\boldsymbol{z} \triangleq \{\boldsymbol{z}_1, \cdots, \boldsymbol{z}_N\}$.

In Equation (2.6), the $\sigma_i$'s can be recognized as the labels of $\boldsymbol{z}_i$'s in a binary classification problem, since $\sigma_i \in \{1, -1\}$. Therefore, for a given function $g \in \mathcal{G}$, a correct classification occurs when

$\sigma_i g(\boldsymbol{z}_i) > 0$, and the supremum $\sup_{g\in\mathcal{G}} \frac{1}{N}\sum_{i=1}^{N} \sigma_i g(\boldsymbol{z}_i)$ finds the function in the HS $\mathcal{G}$ that gives the best averaged classification performance. It is not hard to see that the ERC measures the richness of the HS $\mathcal{G}$: For a rich function space $\mathcal{G}$, given any random label of $\boldsymbol{z}_i$'s, it is possible to find a function $g$ that gives good classification performance, while a space $\mathcal{G}$ that contains only a few candidate functions may not be able to find a good performing function $g$ for arbitrarily labeled training data. This argument is also valid for the distribution dependent RC $R(\mathcal{Z})$.

In the following *Talagrand's Lemma*, a very useful property of ERC is introduced:

**Lemma 1** (Lemma 4.2, [69])**.** *Let $\psi : \mathbb{R} \to \mathbb{R}$ be a Lipschitz continuous function with Lipschitz constant $\gamma$. Define the function space $\psi \circ \mathcal{G} \triangleq \{\boldsymbol{z} \mapsto \psi(g(\boldsymbol{z})) : g \in \mathcal{G}\}$, then*

$$\hat{R}(\psi \circ \mathcal{G}) \leq \frac{1}{\gamma}\hat{R}(\mathcal{G}) \tag{2.8}$$

Based on Theorem 1 and Lemma 1, we introduce the generalization bound for binary classification problems:

**Theorem 2.** *Let $\{(\boldsymbol{x}_i, y_i)\}, \boldsymbol{x}_i \in \mathcal{X}, y_i \in \mathcal{Y} = \{1, -1\}, i = 1, \cdots, N$ be a set of training data that are drawn identically and independently from a probability distribution $P(X, Y)$, and $L : \mathbb{R} \to [0, 1]$ be a fixed Lipschitz continuous loss function with Lipschitz constant $\gamma$ and upper-bound the $0/1$ loss function. Let the HS $\mathcal{F}$ be a family of functions $f : \mathcal{X} \to \mathbb{R}$. Then, for any $\delta > 0$, with probability at least $1 - \delta$, each of the following holds for all $f \in \mathcal{F}$:*

$$er(f) \leq \hat{er}_L(f) + \frac{1}{\gamma}\hat{R}(\mathcal{F}) + \sqrt{\frac{9\ln\frac{1}{\delta}}{2N}} \tag{2.9}$$

$$er(f) \leq \hat{er}_L(f) + \frac{1}{\gamma} R(\mathcal{F}) + \sqrt{\frac{\ln \frac{1}{\delta}}{2N}} \tag{2.10}$$

Note that although the above theorem requires the range of $L$ to be upper bounded by $1$. In fact, it can be extended to the more general cases where $L : \mathbb{R} \to [0, a]$, $a > 0$, with an additional constant factor multiply to the RC and ERC term. In this research, to keep notations uncluttered, we assume that $a = 1$.

The above theorem tells us that the generalization error for an arbitrary $f \in \mathcal{F}$ is upper bounded by the summation of three terms: the empirical loss, the ERC (or RC) of the HS $\mathcal{F}$, and a constant term. Obviously, in order to get small generalization error, we want the upper bound to be as small as possible. Therefore, it is not hard to see that there exists a trade-off with the capacity of the HS $\mathcal{F}$: On one hand, we want the capacity of $\mathcal{F}$ to be low, which gives a small value of the ERC or the RC. However, a small set of function may not contain a good function $f$ that gives small empirical loss $\hat{er}(f)$ on the training set. On the other hand, we want a rich HS $\mathcal{F}$ so that we can find a $f \in \mathcal{F}$ that gives small empirical loss, but it will generate large ERC (or RC) value. In the ideal case, $\mathcal{F}$ contains only one function $f^* \triangleq \arg\min_f er(f)$. However, since the probability distribution $P(X, Y)$ is unknown, it is impossible to calculate $er(f)$, and thus impractical to find $f^*$. Therefore, it is important to carefully design the HS $\mathcal{F}$, so that a good trade-off can be found between the empirical loss and the ERC (or RC).

It is worth to note that the RC $R(\mathcal{F})$ is dependent on the unknown distribution $P(X, Y)$, which is impossible to calculate, and usually difficult to estimate. Therefore, the generalization bound (2.10) is often only with theoretical interests. On the other hands, it is easier to calculate or estimate the ERC, since it is distribution free. In this dissertation, we focus on the ERC-based distribution free generalization bound.

Based on the above description, the generalization bound not only reveals the reason behind the good or bad performance of a given model, but also provides a way to estimate the expected error $er(f)$, which is the very quantity that practitioners care about. Obviously, the latter can be achieved as soon as the ERC is calculated or estimated. Therefore, the generalization bound is a crucial topic of machine learning research, and deriving the upper bound for the ERC values of specific HSs becomes the center of this research.

## Support Vector Machine

SVM [27] [93] is one of the most important classifiers in the machine learning society. Its reputation is earned primarily due to its strong theoretical foundation, appealing geometrical interpretation, and good generalization performance. Also, although born as a linear classifier, it is able to adapt non-linearity by simply applying the kernel method, which leads to even better generalization performance in many scenarios. As a Quadratic Programming problem, a large amount of algorithms have been proposed specifically for SVM, which improved the training speed dramatically . Moreover, there exists open source library for fast SVM training, which facilitates the usage of SVM for both researchers and practitioners. Due to these advantages, a large amount of MTL models are built based on SVM, and similarly, in this research, our MTL and MT-MKL formulations that are proposed in Chapter 4 are based on the SVM model. In this section, we give an introduction to SVM.

### *Formulation*

Consider a set of training data $\mathcal{D} \triangleq \{(\boldsymbol{x}_i, y_i)\}$ with each datum $(\boldsymbol{x}_i, y_i) \in \mathcal{X} \times \{-1, 1\}, i = 1, \cdots, N$, where $\mathcal{X} \subseteq \mathbb{R}^n$. Assume that the data set is linearly separable, *i.e.*, the data from

the two classes, labeled $1$ and $-1$ respectively, can be completely separated by a hyper-plane $\langle \boldsymbol{w}, \boldsymbol{x}_i \rangle + b = 0$, which is parametrized by $(\boldsymbol{w}, b)$. In other words, we have that $y_i(\langle \boldsymbol{w}, \boldsymbol{x}_i \rangle + b) > 0, \forall i = 1, \cdots, N$. While there may exist an infinite number of such hyper-planes, the SVM seeks the optimal hyper-plane $(\boldsymbol{w}, b)$ in terms of margin maximization, as illustrated in Figure 2.1, where the margin with respect to the hyper-plane and the training set is defined as the minimal distance between the training data and the hyper-plane:

$$\min_{(\boldsymbol{x}_i, y_i) \in \mathcal{D}} \frac{|\langle \boldsymbol{w}, \boldsymbol{x}_i \rangle + b|}{\|\boldsymbol{w}\|_2} \tag{2.11}$$

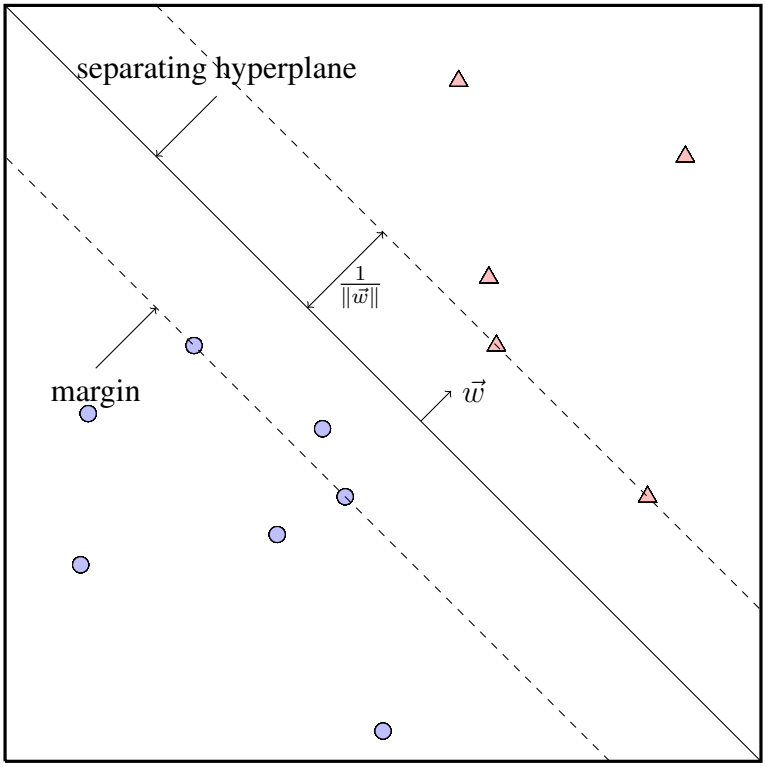



Figure 2.1: The support vector machine seeks a hyperplane that separates data from the two classes by maximizing the margin.

To find the hyper-plane $(\boldsymbol{w}, b)$ that separates the training data and maximizes the margin, naturally, the following problem can be proposed:

$$\begin{aligned} \max_{\boldsymbol{w},b} \min_{(\boldsymbol{x}_i,y_i)\in\mathcal{D}} & \frac{|\langle \boldsymbol{w}, \boldsymbol{x}_i\rangle + b|}{\|\boldsymbol{w}\|_2} \\ \text{s.t. } & y_i(\langle \boldsymbol{w}, \boldsymbol{x}_i\rangle + b) > 0, i = 1, \cdots, N. \end{aligned} \tag{2.12}$$

Problem (2.12) is generally difficult to solve. However, note that the objective function is invariant under the transform $(\boldsymbol{w}, b) \to (k\boldsymbol{w}, kb), k > 0$, thus, for any $(\boldsymbol{w}, b)$, it can be assumed without loss of generality that $\min_{(\boldsymbol{x}_i,y_i)\in\mathcal{D}} \ |\langle \boldsymbol{w}, \boldsymbol{x}_i\rangle + b| = 1$, which yields the following new expression for margin:

$$\frac{1}{\|\boldsymbol{w}\|_2} \tag{2.13}$$

Also, Problem (2.12) is equivalent to the following problem:

$$\begin{aligned} \max_{\boldsymbol{w},b} & \frac{1}{\|\boldsymbol{w}\|_2} \\ \text{s.t. } & y_i(\langle \boldsymbol{w}, \boldsymbol{x}_i\rangle + b) \geq 1, i = 1, \cdots, N. \end{aligned} \tag{2.14}$$

or, more often equivalently written as

$$\begin{aligned} \min_{\boldsymbol{w},b} & \frac{\|\boldsymbol{w}\|_2^2}{2} \\ \text{s.t. } & y_i(\langle \boldsymbol{w}, \boldsymbol{x}_i\rangle + b) \geq 1, i = 1, \cdots, N. \end{aligned} \tag{2.15}$$

Problem (2.15) is usually referred to as hard-margin SVM for the separable case, *i.e.*, the training data from the two classes can be separated by a hyper-plane. However, in practice, the data are

not usually linearly separable, which makes Problem (2.15) infeasible. To tackle this issue, the so-called soft-margin SVM allows some data samples to violate the constraint $y_i(\langle \boldsymbol{w}, \boldsymbol{x}_i \rangle + b) \geq 1$, by adding the slack variables $\xi_i$'s:

$$\begin{aligned} \min_{\boldsymbol{w},b,\boldsymbol{\xi}} \ & \frac{\|\boldsymbol{w}\|_2^2}{2} + C\sum_{i=1}^{N} \xi_i \\ \text{s.t. } & y_i(\langle \boldsymbol{w}, \boldsymbol{x}_i \rangle + b) \geq 1 - \xi_i, i = 1, \cdots, N \\ & \xi_i \geq 0, i = 1, \cdots, N. \end{aligned} \tag{2.16}$$

where $\boldsymbol{\xi} \triangleq [\xi_1, \cdots, \xi_N]'$. The slack variables reflect the degree of violation of the original hard-margin constraint $y_i(\langle \boldsymbol{w}, \boldsymbol{x}_i \rangle + b) \geq 1$, while the violation is penalized via the minimization of $\sum_{i=1}^{N} \xi_i$. The hyper-parameter $C$ is usually chosen by the user via the cross-validation strategy, which specifies the trade-off between the penalty over the margin and the violations. Problem (2.16) can be equivalently transformed to the following more compact form:

$$\min_{\boldsymbol{w},b} \ \frac{\|\boldsymbol{w}\|_2^2}{2} + C\sum_{i=1}^{N} L_{hinge}(y_i(\langle \boldsymbol{w}, \boldsymbol{x}_i \rangle + b)) \tag{2.17}$$

where $L_{hinge}(\cdot)$ is referred to as the *hinge loss*, which is defined as $L_{hinge}(y_i(\langle \boldsymbol{w}, \boldsymbol{x}_i \rangle + b)) \triangleq \mathbf{1}_{[0,+\infty)}(1 - y_i(\langle \boldsymbol{w}, \boldsymbol{x}_i \rangle + b))$.

In practice, the above soft-margin SVM model is the most frequently used, instead of the hard-margin formulation, since as stated above, the linearly separable data set is barely encountered.

*The Dual Formulation and Kernel Methods*

The soft-margin SVM is usually equivalently formulated in the dual domain. Applying the Lagrangian multiplier method on Problem (2.16) gives the following dual formulation:

$$\begin{aligned} \max_{\boldsymbol{\alpha}} \ & \sum_{i=1}^{N} \alpha_i - \frac{1}{2} \sum_{i=1}^{N} \sum_{j=1}^{N} \alpha_i \alpha_j y_i y_j \langle \boldsymbol{x}_i, \boldsymbol{x}_j \rangle \\ \text{s.t.} \ & \sum_{i=1}^{N} \alpha_i y_i = 0 \\ & 0 \leq \alpha_i \leq C, i = 1, \cdots, N. \end{aligned} \tag{2.18}$$

where $\boldsymbol{\alpha} \triangleq [\alpha_1, \cdots, \alpha_N]'$, and the $\alpha_i$'s are the Lagrangian multipliers. Note that Problem (2.18) is a Quadratic Programming problem, and all the calculations involving the training data $\boldsymbol{x}_i$'s are solely within the inner product term $\langle \boldsymbol{x}_i, \boldsymbol{x}_j \rangle$. This characteristic of the dual problem allows the application of the kernel method, which transfers the linear SVM to a non-linear model.

**Definition 1** (Definition 5.1, [71]). *A function $k : \mathcal{X} \times \mathcal{X} \to \mathbb{R}$ is called a kernel over $\mathcal{X}$.*

**Definition 2** (Definition 5.2, [71]). *A kernel function $k : \mathcal{X} \times \mathcal{X} \to \mathbb{R}$ is said to be Positive Definite Symmetric (PDS) if for any $(\boldsymbol{x}_1, \cdots, \boldsymbol{x}_N), \boldsymbol{x}_i \in \mathcal{X}, \forall i = 1, \cdots, N$, the matrix $\boldsymbol{K} \in \mathbb{R}^{N \times N}$, whose $(i, j)$-th element calculated as $k(\boldsymbol{x}_i, \boldsymbol{x}_j)$, is symmetric positive semidefinite.*

Note that, here, there is no particular assumption about the input space $\mathcal{X}$, thus it is not limited to the $\mathbb{R}^n$ space.

**Theorem 3** (Theorem 5.2, [71]). *Let $k : \mathcal{X} \times \mathcal{X} \to \mathbb{R}$ be a PDS kernel. Then, there exists a Hilbert space $\mathcal{H}$ and a mapping $\phi : \mathcal{X} \to \mathcal{H}$, such that*

$$\forall \boldsymbol{x}_i, \boldsymbol{x}_j \in \mathcal{X}, k(\boldsymbol{x}_i, \boldsymbol{x}_j) = \langle \phi(\boldsymbol{x}_i), \phi(\boldsymbol{x}_j) \rangle_{\mathcal{H}}. \tag{2.19}$$

*Furthermore, $\mathcal{H}$ has the following property known as the reproducing property:*

$$\forall h \in \mathcal{H}, \forall \boldsymbol{x} \in \mathcal{X}, h(\boldsymbol{x}) = \langle h, k(\boldsymbol{x}, \cdot) \rangle_{\mathcal{H}}. \tag{2.20}$$

*$\mathcal{H}$ is called a Reproducing Kernel Hilbert Space (RKHS) associated to $k$.*

Based on the above theorem, for any PDS kernel $k$, there exist an associated mapping $\phi$, which is usually referred to as the *feature mapping*, and an associated Reproducing Kernel Hilbert Space (RKHS) $\mathcal{H}$, which is usually referred to as the *feature space*. By applying the feature mapping on the training data $\boldsymbol{x}_i$'s, Problem (2.18) now becomes

$$\begin{aligned} \max_{\boldsymbol{\alpha}} \ & \sum_{i=1}^{N} \alpha_i - \frac{1}{2} \sum_{i=1}^{N} \sum_{j=1}^{N} \alpha_i \alpha_j y_i y_j \langle \phi(\boldsymbol{x}_i), \phi(\boldsymbol{x}_j) \rangle_{\mathcal{H}} \\ \text{s.t.} \ & \sum_{i=1}^{N} \alpha_i y_i = 0 \\ & 0 \leq \alpha_i \leq C, i = 1, \cdots, N. \end{aligned} \tag{2.21}$$

where the inner product $\langle \phi(\boldsymbol{x}_i), \phi(\boldsymbol{x}_j) \rangle_{\mathcal{H}}$ is computed in the RKHS $\mathcal{H}$. Since the RKHS $\mathcal{H}$ usually has a higher dimension than the input space $\mathcal{X}$, and may be infinite dimensional, the inner product may be intractable, if computed directly. However, since $k(\boldsymbol{x}_i, \boldsymbol{x}_j) = \langle \phi(\boldsymbol{x}_i), \phi(\boldsymbol{x}_j) \rangle_{\mathcal{H}}$, we can substitute the inner product with kernel computation. Therefore, Problem (2.21) is further

transformed to

$$\begin{aligned} \max_{\boldsymbol{\alpha}} \; & \boldsymbol{\alpha}'\mathbf{1} - \frac{1}{2}\boldsymbol{\alpha}'\boldsymbol{Y}\boldsymbol{K}\boldsymbol{Y}\boldsymbol{\alpha} \\ \text{s.t. } & \boldsymbol{\alpha}'\boldsymbol{y} = 0 \\ & \mathbf{0} \preceq \boldsymbol{\alpha} \preceq C\mathbf{1}. \end{aligned} \tag{2.22}$$

where $\boldsymbol{K}$ is the kernel matrix evaluated on the training set. A major advantage of the above kernelized SVM formulation is that it brings non-linearity into the linear SVM model. This can be more clearly observed via the corresponding primal formulation

$$\min_{\boldsymbol{w},b} \; \frac{\|\boldsymbol{w}\|_{\mathcal{H}}^2}{2} + C\sum_{i=1}^{N} L_{hinge}(y_i(\langle \boldsymbol{w}, \phi(\boldsymbol{x}_i)\rangle_{\mathcal{H}} + b)) \tag{2.23}$$

where $\boldsymbol{w} \in \mathcal{H}$, $\|\boldsymbol{w}\|_{\mathcal{H}} \triangleq \sqrt{\langle \boldsymbol{w}, \boldsymbol{w}\rangle_{\mathcal{H}}}$. It is not difficult to see that, the kernelized SVM trains the linear SVM in the RKHS $\mathcal{H}$, which is potentially non-linear in the original input space $\mathcal{X}$, due to the non-linear feature mapping $\phi$. By appropriately choosing the kernel function $k$ (thus the feature mapping $\phi$ and the RKHS $\mathcal{H}$), it is possible that the training data in the feature space $\mathcal{H}$ is better distributed, compared to that in the original input space $\mathcal{X}$, so that a better classification performance can be expected. A concrete example that justifies this argument is given in Figure 5.2 of [71]. Some commonly used kernel functions are listed in Table 2.1. Usually, the kernel parameters, *e.g.*, $\sigma$ in the Gaussian kernel, are selected via cross-validation.

### *Generalization Bound for SVM*

Before discussing the generalization bound for SVM, we first present the following proposition, which was introduced as the first part of Proposition 12 in [52].

Table 2.1: Commonly used kernel functions

| Kernel Name | Mathematical Expression |
|---|---|
| Linear Kernel | $\langle \boldsymbol{x}_1, \boldsymbol{x}_2 \rangle$ |
| Polynomial Kernel | $(\langle \boldsymbol{x}_1, \boldsymbol{x}_2 \rangle + c)^d$ |
| Gaussian Kernel | $\exp(\frac{\|\boldsymbol{x}_1 - \boldsymbol{x}_2\|^2}{2\sigma^2})$ |
| Sigmoid Kernel | $\tanh(\kappa \langle \boldsymbol{x}_1, \boldsymbol{x}_2 \rangle - \delta)$ |

**Proposition 1** (Proposition 12, [52]). *Let $\mathcal{A} \subseteq \mathcal{X}$, and let $f, g : \mathcal{A} \mapsto \mathbb{R}$ be two functions. For any $\sigma > 0$, there must exist a $\tau > 0$, such that the following two problems are equivalent*

$$\min_{x \in \mathcal{A}} \ f(x) + \sigma g(x) \tag{2.24}$$

$$\min_{x \in \mathcal{A}, g(x) \leq \tau} \ f(x) \tag{2.25}$$

Based on the above proposition, we immediately have that, for any $C$, there must exist a $r > 0$, such that the soft-margin SVM problem (2.23) is equivalent to

$$\begin{aligned} &\min_{\boldsymbol{w}, b} \ \sum_{i=1}^{N} L_{hinge}(y_i(\langle \boldsymbol{w}, \phi(\boldsymbol{x}_i) \rangle_{\mathcal{H}} + b)) \\ &\text{s.t. } \|\boldsymbol{w}\|_{\mathcal{H}}^2 \leq r^2 \end{aligned} \tag{2.26}$$

Therefore, the essence of SVM is to seek a function $f : \mathcal{H} \to \mathbb{R}$, parametrized with $\boldsymbol{w} \in \mathcal{H}$ and

$b \in \mathbb{R}$, from the HS

$$\mathcal{F} = \{\boldsymbol{x} \mapsto \langle \boldsymbol{w}, \phi(\boldsymbol{x}) \rangle_{\mathcal{H}} + b : \|\boldsymbol{w}\|_{\mathcal{H}}^2 \leq r^2, b \in \mathbb{R}\} \tag{2.27}$$

by minimizing the empirical hinge loss. Given the loss function and HS of the SVM formulation, we are ready to present its generalization bound.

**Theorem 4.** *Let* $\{(\boldsymbol{x}_i, y_i)\}, \boldsymbol{x}_i \in \mathcal{X}, y_i \in \mathcal{Y} = \{1, -1\}, i = 1, \cdots, N$ *be a set of training data that are drawn identically and independently from a probability distribution* $P(X, Y)$*. Let the HS* $\mathcal{F}$ *be defined as* $\mathcal{F} \triangleq \{\boldsymbol{x} \mapsto \langle \boldsymbol{w}, \phi(\boldsymbol{x}) \rangle_{\mathcal{H}} + b : \|\boldsymbol{w}\|_{\mathcal{H}}^2 \leq r^2, b \in \mathbb{R}\}$*, where* $\mathcal{H}$ *is a RKHS associated with a kernel function* $k$*. Then, for any* $\delta > 0$*, with probability at least* $1 - \delta$*, each of the following holds for all* $f \in \mathcal{F}$*:*

$$\begin{aligned} er(f) &\leq \hat{er}_{L_{hinge}}(f) + \hat{R}(\mathcal{F}) + \sqrt{\frac{9 \ln \frac{1}{\delta}}{2N}} \\ &\leq \hat{er}_{L_{hinge}}(f) + \frac{2r}{N}\sqrt{tr(\boldsymbol{K})} + \sqrt{\frac{9 \ln \frac{1}{\delta}}{2N}} \end{aligned} \tag{2.28}$$

*where* $\boldsymbol{K}$ *is the kernel matrix calculated by* $k$ *on the training set.*

According to Theorem 4, it is clear that the generalization error of SVM is upper-bounded by the empirical hinge loss, the ERC, and a constant term. It is important to note that, the last two terms decreases to $0$ when the number of training samples $N$ goes to infinity. This means that, the more training samples we have, the closer the generalization error is to the empirical loss, and thus we can more accurately estimate the generalization error based on the empirical loss. Besides, since a tighter generalization bound means a potentially smaller expected error, the above bound indicates that as the number of training data increases, a higher classification accuracy is

likely to be achieved. Further more, if we assume that $k(\boldsymbol{x}, \boldsymbol{x}) = 1, \forall \boldsymbol{x} \in \mathcal{X}$, then the ERC is upper-bounded by $\frac{2r}{\sqrt{N}}$. In this case, the last two terms decrease in a speed of order $O(\frac{1}{\sqrt{N}})$.

Based on the above generalization bound, heuristically, we want $r$ to be small, which decreases the value of the ERC. However, as we stated before, this choice shrinks the size of HS $\mathcal{F}$, thus there may not exist a function $f \in \mathcal{F}$ that gives small empirical hinge loss. Clearly, it is crucial to carefully choose $r$, which is equivalently to selecting the parameter $C$ in the SVM model.

Finally, we point out that the ERC of $\mathcal{F}$ equals the ERC of the following HS:

$$\mathcal{F}' = \{\boldsymbol{x} \mapsto \langle \boldsymbol{w}, \phi(\boldsymbol{x}) \rangle_{\mathcal{H}} : \|\boldsymbol{w}\|_{\mathcal{H}}^2 \leq r^2\} \tag{2.29}$$

where the difference between $\mathcal{F}$ and $\mathcal{F}'$ is that, in $\mathcal{F}'$, the bias $b$ is equal to $0$. Such an equal value of ERC between these two HSs can be directly observed from the proof of Theorem 4, which is given in the appendix. Therefore, when considering the HS, it is a common practice to omit $b$. In this dissertation, we follow the same convention.

### Multiple Kernel Learning

Kernel methods [86] play an important role in machine learning. By implicitly mapping samples from the original space into a potentially infinite-dimensional feature space, it is hoped that samples are appropriately distributed in the feature space, such that kernel machines, like SVM, could perform better in the feature space than in the original space. As introduced before, an elegant property that kernel methods have is that, the inner product in the feature space, which may not be feasible to calculate due to potential infinite dimensionality of the feature space, can be calculated directly via a kernel function. Furthermore, kernel methods are able to bring non-linearity to a

linear model, due to the potential non-linearity of the feature mapping $\phi$. Since the feature mapping $\phi$, and thus the RKHS $\mathcal{H}$, is implicitly defined via the kernel function, it is important to choose the kernel appropriately for a given task.

A traditional approach for kernel selection is cross-validation. The "optimal" kernel function is chosen from a set of pre-selected kernel functions based on the performance on the validation set: the one that leads to the best performance (*e.g.*, classification accuracy for classification problems) on the validation set is used. However, there are a few disadvantages to this strategy. For example, it requires training of the model on each candidate kernel function, which significantly increases computational cost. Also, the selected kernel function is limited in the set of pre-selected kernel functions, which may not contain good candidates at all.

In recent years, substantial effort has been devoted on how to learn the kernel function (or equivalently, kernel matrix) from the available data, thus cross-validation over a collection of kernel functions can be avoided. Some kernel learning methods include Kernel Target Alignment [87], MKL [55], optimal neighborhood kernel [61], etc. Out of the abundant achievements in kernel learning methods, MKL is one of the most popular strategies, since it is easy to interpret, relatively simple to optimize, and is well studied theoretically. In this section, we give an introduction to MKL, which is an important building block of this research.

### *Formulation*

MKL and cross-validation for kernel selection share the commonality that they both require a set of pre-specified kernel functions, say $k_1, \cdots, k_M$. However, unlike cross-validation, MKL linearly combine the $M$ kernel functions with coefficients $\boldsymbol{\theta} = [\theta_1, \cdots, \theta_M]'$: $k = \sum_{m=1}^{M} \theta_m k_m$, where the coefficients are learned during the training stage. Therefore, the kernel function is selected during training in a data-driven manner, instead of depending on the validation set. It is worth to mention

that, although there are MKL models that consider non-linearly combined kernel functions, such as [25], such methods are usually more complicated, and difficult to optimize and analyze, without significant benefit on generalization performance. Therefore, in this research, we focus on the MKL approaches that linearly combine the pre-specified kernel functions.

For different MKL formulations, various types of constraints may be added to $\boldsymbol{\theta}$. As an example, one of the most widely utilized constraint is the $L_p$-norm constraint [52], which gives the so-called $L_p$-norm MKL model. Its feasible region $\Omega(\boldsymbol{\theta})$ is defined as $\Omega(\boldsymbol{\theta}) \triangleq \{\boldsymbol{\theta} : \boldsymbol{\theta} \succeq \mathbf{0}, \|\boldsymbol{\theta}\|_p \leq 1\}$ with $p \geq 1$ selected a priori. Therefore, the candidate kernel functions are all possible linear combinations of the pre-selected kernel functions, when the coefficients are in the feasible region $\Omega(\boldsymbol{\theta})$. This formulates a much richer space of candidate kernel functions, compared to cross-validation, which only considers each single kernel functions $k_1, \cdots, k_M$.

Suppose that each of the $M$ kernel functions are associated to a feature mapping $\phi_1, \cdots, \phi_M$, where $\phi_m : \mathcal{X} \to \mathcal{H}_m, m = 1, \cdots, M$ and $\mathcal{H}_m$ is a RKHS that is implicitly determined by $k_m$. Then, we can easily observe the following relation:

$$\begin{aligned} k(\boldsymbol{x}_i, \boldsymbol{x}_j) &= \sum_{i=1}^{M} \theta_m k_m(\boldsymbol{x}_i, \boldsymbol{x}_j) \\ &= \sum_{i=1}^{M} \theta_m \langle \phi_m(\boldsymbol{x}_i), \phi_m(\boldsymbol{x}_j) \rangle_{\mathcal{H}_m} \\ &= \langle \phi(\boldsymbol{x}_i), \phi(\boldsymbol{x}_j) \rangle_{\mathcal{H}_{\boldsymbol{\theta}}} \end{aligned} \tag{2.30}$$

where $\phi : \mathcal{X} \to \mathcal{H}_{\boldsymbol{\theta}}$, $\mathcal{H}_{\boldsymbol{\theta}} \triangleq \mathcal{H}_1 \times \cdots \times \mathcal{H}_M$ and $\phi(\boldsymbol{x}) \triangleq (\sqrt{\theta_1}\phi_1(\boldsymbol{x}), \cdots, \sqrt{\theta_M}\phi_M(\boldsymbol{x}))$. Therefore, in essence, by linearly combining multiple kernel functions, MKL constructs a new RKHS $\mathcal{H}_{\boldsymbol{\theta}}$, which is the direct product of the $M$ RKHSs that are defined by the pre-selected kernel functions. Given a linear model, such as SVM, instead of training it in the original space $\mathcal{X}$, we can now train

it in the feature space $\mathcal{H}_{\boldsymbol{\theta}}$, which, parametrized by $\boldsymbol{\theta}$, is also learned in the training stage. Below we introduce the SVM-based MKL formulation.

Let $\boldsymbol{w} \triangleq (\sqrt{\theta_1}\boldsymbol{w}_1, \cdots, \sqrt{\theta_M}\boldsymbol{w}_M) \in \mathcal{H}_{\boldsymbol{\theta}}$, where $\boldsymbol{w}_m \in \mathcal{H}_m, \forall m = 1, \cdots, M$, the SVM-based MKL method is a simple variation of the original SVM model:

$$\min_{\boldsymbol{w}, b, \boldsymbol{\theta} \in \Omega(\boldsymbol{\theta})} \frac{\|\boldsymbol{w}\|_{\mathcal{H}_{\boldsymbol{\theta}}}^2}{2} + C \sum_{i=1}^{N} L_{hinge}(y_i(\langle \boldsymbol{w}, \phi(\boldsymbol{x}_i)\rangle_{\mathcal{H}_{\boldsymbol{\theta}}} + b)) \tag{2.31}$$

Note that, here, $\boldsymbol{\theta}$ is optimized in the feasible region $\Omega(\boldsymbol{\theta})$ such that the objective function is minimized. However, one disadvantage of this model is that it is not jointly convex with respect to all the variables. This is because that neither the norm nor the inner product term is jointly convex with respect to $\boldsymbol{w}$ and $\boldsymbol{\theta}$, as shown below:

$$\|\boldsymbol{w}\|_{\mathcal{H}_{\boldsymbol{\theta}}}^2 \triangleq \sum_{m=1}^{M} \theta_m \|\boldsymbol{w}_m\|_{\mathcal{H}_m}^2 \tag{2.32}$$

$$\langle \boldsymbol{w}, \phi(\boldsymbol{x}_i)\rangle_{\mathcal{H}_{\boldsymbol{\theta}}} = \sum_{m=1}^{M} \theta_m \langle \boldsymbol{w}_m, \phi_m(\boldsymbol{x}_i)\rangle_{\mathcal{H}_m} \tag{2.33}$$

Therefore, directly optimizing Problem (2.31) may converge to a local minimum, which is less preferred. The remedy to this issue is simple: By applying the variable change $\boldsymbol{w}_m \leftarrow \boldsymbol{w}_m\theta_m, \forall m = 1, \cdots, M$, we reach the following model

$$\min_{\boldsymbol{w}, b, \boldsymbol{\theta} \in \Omega(\boldsymbol{\theta})} \sum_{m=1}^{M} \frac{\|\boldsymbol{w}_m\|_{\mathcal{H}_m}^2}{2\theta_m} + C \sum_{i=1}^{N} L_{hinge}(y_i(\sum_{m=1}^{M} \langle \boldsymbol{w}_m, \phi_m(\boldsymbol{x}_i)\rangle_{\mathcal{H}_m} + b)) \tag{2.34}$$

which is jointly convex with respect to all variables.

Besides the convexity, another advantage of Problem (2.34) is that it is easy to solve by using block-coordinate descent method. To be more concrete, it is convenient to split the variables into two groups, namely $\{\boldsymbol{w}, b\}$ as a group and $\boldsymbol{\theta}$ as another, and then alternate the optimization with respect to one block of variables with the other block fixed. When $\boldsymbol{\theta}$ is fixed, Problem (2.34) is simply an SVM problem, which has the following dual form:

$$\begin{aligned} \max_{\boldsymbol{\alpha}} \ & \boldsymbol{\alpha}'\mathbf{1} - \frac{1}{2}\boldsymbol{\alpha}'\boldsymbol{Y}(\sum_{m=1}^{M}\theta_m \boldsymbol{K}_m)\boldsymbol{Y}\boldsymbol{\alpha} \\ \text{s.t. } & \boldsymbol{\alpha}'\boldsymbol{y} = 0 \\ & \mathbf{0} \preceq \boldsymbol{\alpha} \preceq C\mathbf{1} \end{aligned} \tag{2.35}$$

where $\boldsymbol{\alpha} \triangleq [\alpha_1, \cdots, \alpha_N]'$, $\boldsymbol{y} \triangleq [y_1, \cdots, y_N]'$, $\boldsymbol{Y} \triangleq \mathit{diag}(\boldsymbol{y}) \in \mathbb{R}^{N\times N}$, and the *diag* function maps the vector $\boldsymbol{y}$ to a diagonal matrix $\boldsymbol{Y}$, whose diagonal elements are given by $\boldsymbol{y}$. $\boldsymbol{K}_m \in \mathbb{R}^{N\times N}$ is the kernel matrix calculated by the kernel function $k_m$ on the training set.

Compared to the original SVM dual formulation (2.22), we observe that Problem (2.35) is the same as Problem (2.22), except the kernel matrix is substituted by the linear combination of the $M$ kernel matrices; therefore it can be efficiently solved via SVM solver, such as `LIBSVM` [18].

On the other hand, when $\{\boldsymbol{w}, b\}$ is fixed, optimization on $\boldsymbol{\theta}$ is simply the following problem:

$$\min_{\boldsymbol{\theta}\in\Omega(\boldsymbol{\theta})} \sum_{m=1}^{M} \frac{\|\boldsymbol{w}_m\|_{\mathcal{H}_m}^2}{2\theta_m} \tag{2.36}$$

while it is straightforward to prove that $\|\boldsymbol{w}_m\|_{\mathcal{H}_m}^2 = \boldsymbol{\alpha}'\boldsymbol{Y}\boldsymbol{K}_m\boldsymbol{Y}\boldsymbol{\alpha}$. This problem is usually easy to solve. Take the most commonly encountered $L_p$-norm MKL as an example, $\Omega(\boldsymbol{\theta})$ is defined as $\Omega(\boldsymbol{\theta}) \triangleq \{\boldsymbol{\theta} : \boldsymbol{\theta} \succeq \mathbf{0}, \|\boldsymbol{\theta}\|_p \leq 1\}$ with $p \geq 1$. In this scenario, the following theorem tells us that

Problem (2.36) has closed-form solution.

**Theorem 5** (Lemma 2, [57]). *For $q \in [\frac{1}{2}, 1)$, $\boldsymbol{\theta} \in \mathbb{R}^N$, $\boldsymbol{c} \in \mathbb{R}^N$, $\boldsymbol{c} \succeq \mathbf{0}$ and $\boldsymbol{c} \neq \mathbf{0}$, we have*

$$\min_{\boldsymbol{\theta} \in \Omega(\boldsymbol{\theta})} \boldsymbol{c}' \boldsymbol{\theta}^{-1} = (\sum_{n=1}^{N} c_n^q)^{\frac{1}{q}} \tag{2.37}$$

*with $\Omega(\boldsymbol{\theta}) \triangleq \{\boldsymbol{\theta} : \boldsymbol{\theta} \succeq \mathbf{0}, \|\boldsymbol{\theta}\|_p \leq 1\}$, $p \triangleq \frac{q}{1-q}$, and the optimal $\boldsymbol{\theta}^*$ can be calculated as follows:*

$$\boldsymbol{\lambda}^* = \left( \frac{\boldsymbol{c}}{(\sum_{m=1}^{M} c_m^q)^{\frac{1}{q}}} \right)^{1-q} \tag{2.38}$$

To summarize, the algorithm for solving Problem (2.34) involves iterating over two simple steps: An SVM problem and a minimization problem with closed-form solution.

*Multiple Kernel Learning as a Min-Max Problem*

The classical SVM-based MKL problem (2.34) can be converted to an equivalent min-max problem, which is of particular interest. This is achieved by simply applying the Lagrangian multiplier method to Problem (2.34) on variables $\{\boldsymbol{w}, b\}$, which gives the following min-max problem:

$$\begin{aligned} \min_{\boldsymbol{\theta} \in \Omega(\boldsymbol{\theta})} \max_{\boldsymbol{\alpha}} \; & \boldsymbol{\alpha}' \mathbf{1} - \frac{1}{2} \boldsymbol{\alpha}' \boldsymbol{Y} (\sum_{m=1}^{M} \theta_m \boldsymbol{K}_m) \boldsymbol{Y} \boldsymbol{\alpha} \\ & \text{s.t. } \boldsymbol{\alpha}' \boldsymbol{y} = 0 \\ & \mathbf{0} \preceq \boldsymbol{\alpha} \preceq C \mathbf{1} \end{aligned} \tag{2.39}$$

Compared to the original primal domain formulation (2.34), the above dual domain min-max formulation has the following characteristics:

First, the principal of MKL is clearly shown in Problem (2.39): The difference between traditional single kernel learning and MKL is that, MKL linearly combine the $M$ pre-selected kernel functions (thus matrices), and the combination coefficients are learned during the training stage. This is not directly observed from the primal domain formulation (2.34).

Second, compared to the primal formulation, it is more straightforward to formulate different variations of MKL models in the dual domain. For example, given the dual formulation of the Kernel Ridge Regression (KRR) model

$$\max_{\boldsymbol{\alpha}} \ \boldsymbol{\alpha}'(2\boldsymbol{y} - \lambda\boldsymbol{\alpha}) - \boldsymbol{\alpha}'\boldsymbol{K}\boldsymbol{\alpha} \tag{2.40}$$

it is easy to incorporate MKL with KRR in the dual domain by simply substituting the kernel matrix with linear combination of multiple kernel matrices, which gives the following formulation,

$$\min_{\boldsymbol{\theta}\in\Omega(\boldsymbol{\theta})} \max_{\boldsymbol{\alpha}} \ \boldsymbol{\alpha}'(2\boldsymbol{y} - \lambda\boldsymbol{\alpha}) - \boldsymbol{\alpha}'(\sum_{m=1}^{M} \theta_m \boldsymbol{K}_m)\boldsymbol{\alpha} \tag{2.41}$$

while it is more mathematically involved to formulate the above problem in the primal domain. To give another example, the optimal neighborhood kernel method was proposed in [61]:

$$\begin{aligned} \min_{\boldsymbol{K}} \max_{\boldsymbol{\alpha}} \ & \boldsymbol{\alpha}'\mathbf{1} - \frac{1}{2}\boldsymbol{\alpha}'\boldsymbol{Y}\boldsymbol{K}\boldsymbol{Y}\boldsymbol{\alpha} + \rho\|\boldsymbol{K} - \boldsymbol{K}_0\|_F^2 \\ & \text{s.t. } \mathbf{0} \preceq \boldsymbol{\alpha} \preceq C\mathbf{1}, \ \boldsymbol{\alpha}'\boldsymbol{y} = 0, \\ & \boldsymbol{K} \succeq \mathbf{0} \end{aligned} \tag{2.42}$$

It is straightforward to incorporate MKL with the above model, which gives the following min-max

problem

$$\begin{aligned}\min_{\boldsymbol{\theta}\in\Omega(\boldsymbol{\theta})} \max_{\boldsymbol{\alpha}} \; & \boldsymbol{\alpha}'\mathbf{1} - \frac{1}{2}\boldsymbol{\alpha}'\boldsymbol{Y}(\sum_{m=1}^{M}\theta_m\boldsymbol{K}_m)\boldsymbol{Y}\boldsymbol{\alpha} + \rho\|\sum_{m=1}^{M}\theta_m\boldsymbol{K}_m - \boldsymbol{K}_0\|_F^2 \\ & \text{s.t. } \mathbf{0} \preceq \boldsymbol{\alpha} \preceq C\mathbf{1},\ \boldsymbol{\alpha}'\boldsymbol{y} = 0\end{aligned} \tag{2.43}$$

while it is not immediately clear how to formulate its corresponding primal problem.

Albeit the above mentioned advantages of min-max MKL formulations, a common drawback of these models is that they are usually difficult to optimize, and it is not uncommon to require model-specific complicated algorithms to solve the min-max problems. This is even more true, when MTL is combined with MKL, which is called MT-MKL and is discussed later. Such disadvantage inspired us to develop a general purpose algorithm to solve the most min-max MKL, kernel-based MTL and MT-MKL problems. This algorithm is introduced in Chapter 5.

*Generalization Bound of SVM-based $L_p$-norm Multiple Kernel Learning*

Based on the discussion of the generalization bound of SVM, it can be observed from Problem (2.31) that the HS of MKL is

$$\mathcal{F} = \{\boldsymbol{x} \mapsto \langle \boldsymbol{w}, \phi(\boldsymbol{x})\rangle_{\mathcal{H}_{\boldsymbol{\theta}}} : \|\boldsymbol{w}\|_{\mathcal{H}_{\boldsymbol{\theta}}}^2 \leq r^2, \boldsymbol{\theta} \in \Omega(\boldsymbol{\theta})\} \tag{2.44}$$

Therefore, the ERC of the HS $\mathcal{F}$ is given as

$$\begin{aligned}
\hat{R}(\mathcal{F}) &= \frac{2}{N} E_{\boldsymbol{\sigma}} \left\{ \sup_{\|\boldsymbol{w}\|_{\mathcal{H}_{\boldsymbol{\theta}}} \leq r, \boldsymbol{\theta} \in \Omega(\boldsymbol{\theta})} \sum_{i=1}^{N} \sigma_i \langle \boldsymbol{w}, \phi(\boldsymbol{x}_i) \rangle_{\mathcal{H}_{\boldsymbol{\theta}}} \right\} \\
&\leq \frac{2}{N} E_{\boldsymbol{\sigma}} \left\{ \sup_{\|\boldsymbol{w}\|_{\mathcal{H}_{\boldsymbol{\theta}}} \leq r, \boldsymbol{\theta} \in \Omega(\boldsymbol{\theta})} \|\boldsymbol{w}\|_{\mathcal{H}_{\boldsymbol{\theta}}} \, \| \sum_{i=1}^{N} \sigma_i \phi(\boldsymbol{x}_i) \|_{\mathcal{H}_{\boldsymbol{\theta}}} \right\} \\
&= \frac{2r}{N} E_{\boldsymbol{\sigma}} \left\{ \sup_{\boldsymbol{\theta} \in \Omega(\boldsymbol{\theta})} \| \sum_{i=1}^{N} \sigma_i \phi(\boldsymbol{x}_i) \|_{\mathcal{H}_{\boldsymbol{\theta}}} \right\} \\
&= \frac{2r}{N} E_{\boldsymbol{\sigma}} \left\{ \sup_{\boldsymbol{\theta} \in \Omega(\boldsymbol{\theta})} \left( \boldsymbol{\sigma}' \left( \sum_{m=1}^{M} \theta_m \boldsymbol{K}_m \right) \boldsymbol{\sigma} \right)^{\frac{1}{2}} \right\}
\end{aligned} \tag{2.45}$$

Consider the most commonly encountered case, the $L_p$-norm MKL, we have $\Omega(\boldsymbol{\theta}) \triangleq \{\boldsymbol{\theta} : \boldsymbol{\theta} \succeq \mathbf{0}, \|\boldsymbol{\theta}\|_p \leq 1\}$ with $p \geq 1$. By letting $\boldsymbol{u} \triangleq [u_1, \cdots, u_M]'$, $u_m \triangleq \boldsymbol{\sigma}' \boldsymbol{K}_m \boldsymbol{\sigma}, \forall m = 1, \cdots, M$, and $p^* \geq 1$ such that $\frac{1}{p} + \frac{1}{p^*} = 1$, the ERC can be further upper bounded as follows:

$$\begin{aligned}
\hat{R}(\mathcal{F}) &\leq \frac{2r}{N} E_{\boldsymbol{\sigma}} \left\{ \sup_{\boldsymbol{\theta} \in \Omega(\boldsymbol{\theta})} (\boldsymbol{\theta}' \boldsymbol{u})^{\frac{1}{2}} \right\} \\
&\leq \frac{2r}{N} E_{\boldsymbol{\sigma}} \left\{ \sup_{\boldsymbol{\theta} \in \Omega(\boldsymbol{\theta})} (\|\boldsymbol{\theta}\|_p \|\boldsymbol{u}\|_{p^*})^{\frac{1}{2}} \right\} \\
&= \frac{2r}{N} E_{\boldsymbol{\sigma}} \left\{ \|\boldsymbol{u}\|_{p^*}^{\frac{1}{2}} \right\} \\
&\leq \frac{2r}{N} \left\{ \sum_{m=1}^{M} E_{\boldsymbol{\sigma}} \left( \boldsymbol{\sigma}' \boldsymbol{K}_m \boldsymbol{\sigma} \right)^{p^*} \right\}^{\frac{1}{2p^*}} \\
&= \frac{2r}{N} \left\{ \sum_{m=1}^{M} E_{\boldsymbol{\sigma}} \left\| \sum_{i=1}^{N} \sigma_i \phi(\boldsymbol{x}_i) \right\|_{\mathcal{H}_{\boldsymbol{\theta}}}^{2p^*} \right\}^{\frac{1}{2p^*}} \\
&\leq \frac{2r}{N} \left\{ \sum_{m=1}^{M} \left( 2p^* \mathrm{tr}(\boldsymbol{K}_m) \right)^{p^*} \right\}^{\frac{1}{2p^*}} \\
&= \frac{2r}{N} \sqrt{2p^* \| \mathrm{tr}(\boldsymbol{K}_m)_{m=1}^{M} \|_{p^*}}
\end{aligned} \tag{2.46}$$

where the second inequality is based on Hölder's inequality, the third inequality is based on Jensen's inequality, and the last inequality is based on the following theorem:

**Theorem 6.** *Let $\boldsymbol{x}_1, \cdots, \boldsymbol{x}_n \in \mathcal{H}$, then we have that*

$$E_\sigma \| \sum_{i=1}^{n} \sigma_i \boldsymbol{x}_i \|^p \leq (p \sum_{i=1}^{n} \|\boldsymbol{x}_i\|^2)^{\frac{p}{2}} \tag{2.47}$$

*for any $p \geq 1$, where $\sigma_i$'s are the Rademacher-distributed random variables.*

For $1 \leq p < 2$, the above result can be simply proved using Lyapunov's inequality. When $p \geq 2$, the lemma can be proved by using Proposition $3.3.1$ and $3.4.1$ in [54].

It can be seen from (2.46) that the key ingredient of the upper bound of the ERC is the value of $p$ and the trace of the kernel matrices. If we assume that $k_m(\boldsymbol{x}_i, \boldsymbol{x}_i) \leq 1, \forall m = 1, \cdots, M, i = 1, \cdots, N$, then we have that

$$\hat{R}(\mathcal{F}) \leq \frac{2r}{N} \sqrt{2p^* M^{\frac{1}{p^*}}} \tag{2.48}$$

which is minimized with respect to $p$ when $p^* = \ln M$. This gives the following bound of the ERC

$$\hat{R}(\mathcal{F}) \leq \frac{2r}{N} \sqrt{2e \ln M} \tag{2.49}$$

Clearly, the optimal bound of the ERC is of order $O(\frac{\sqrt{\ln M}}{N})$, which increases very slowly with respect to $M$, the number of kernel functions. Therefore, by increasing the number of kernel functions, we significantly increased the volume of the space of candidate kernel functions, thus we may be able to find better kernel function with much smaller empirical loss $\hat{er}(f)$. At the same

time, conducting MKL has only a minor effect on increasing the generalization bound, thus the generalization error $er(f)$ can be upper-bounded by a small value, and therefore, guarantee a good generalization performance.

## Multi-Task Learning

The concept of learning multiple tasks simultaneously is inspired from the learning process of human being:

*"A child trained from birth on a single, isolated, complex task – and nothing else – would be unlikely to learn it. If you were trained from birth to play tennis – and nothing else – you would probably not learn tennis. Instead, you learn to play tennis in a world that tasks you to learn many other things. You also learn to walk, to run, to jump, to exercise, to grasp, to throw, to swing, to recognize objects, to predict trajectories, to rest, to talk, to study, to practice, etc. These tasks are not all the same: running in tennis is different from running on a track, yet they are related. Perhaps the similarities between the thousands of tasks you learn are what enable you to learn any one of them – including tennis."*

The above description is cited from [17], which for the first time introduced the concept of MTL, based on artificial neural networks. Since then, many research works have been devoted to MTL, and a large amount of MTL models have been developed. It has been shown that, indeed, in many scenarios, learning multiple related tasks simultaneously can be advantageous, *i.e.*, better generalization performance can be achieved, compared to learning each task separately. This fact has been also verified in many real world applications, such as stock selection, web search ranking, medical diagnosis, computer vision, etc. Besides, several researchers tried to theoretically explain the benefits and good performance of MTL, in order to achieve a better understanding of MTL.

These research works derive and analyze the generalization bound for MTL HSs, and show the advantage of MTL in terms of sample complexity. A solid theoretical foundation of MTL is gained because of these research works.

Due to the good generalization performance, wide applicability, and sound theoretical foundation, MTL has been becoming an active research area for two decades. In this section, we give an introduction to MTL, and a literature review is given in Chapter 3.

### *Formulation*

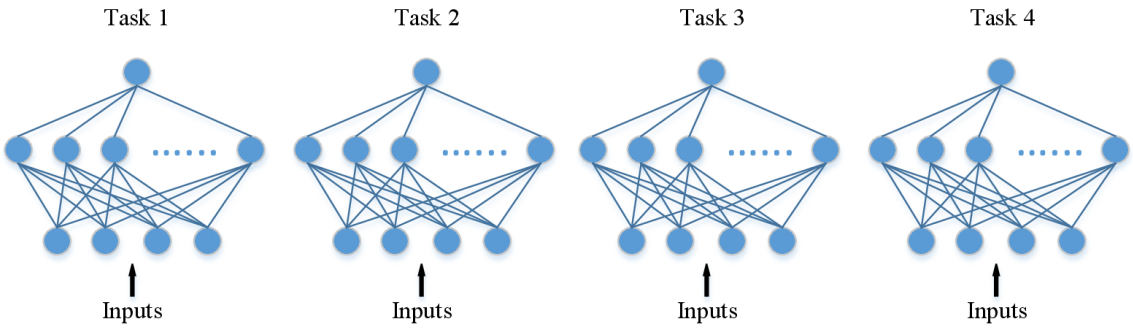
The inception of MTL methods started with the invention of artificial neural network-based model, as proposed in [16]. Given $T$ tasks that can be learned separately via $T$ individual neural networks, the MTL method learns the tasks simultaneously by sharing the hidden layer amongst tasks, as shown in Figure 2.2.

It is not difficult to see that MTL is an inductive transfer method: The information that is gained by training each task is transfered to the other tasks via the shared hidden layer. It is hoped that, by sharing the feature representation in the hidden layer, tasks can be benefited by the training of other tasks, so that better generalization performance can be achieved for each task, compared to the independent training method.

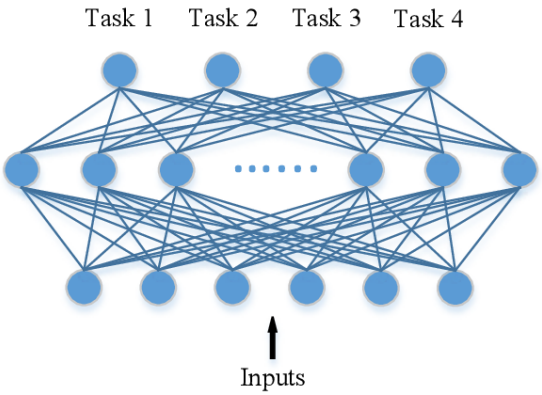
In general, MTL models can be formulated as follows:

Given a model featured with a function $f : \mathcal{X} \to \mathcal{Y}$ and a set of training data $\{\boldsymbol{x}_i, y_i\}, i = 1, \cdots, N$, the Regularization-Loss framework considers the following optimization problem:

$$\min_f \; R(f) + C \sum_{i=1}^{N} L(f(\boldsymbol{x}_i), y_i) \tag{2.50}$$




(a) Four single tasks that are learned separately via neural networks




(b) Four tasks that are learned simultaneously via neural networks with shared hidden layer

Figure 2.2: Multi-Task Learning via neural networks

One important property of $R(f)$ is to prevent over-fitting: Based on Proposition 1, the appearance of the regularizer controls the capacity of the HS $\mathcal{F} = \{f : R(f) \leq r\}$, such that the ERC of the HS is not too large, which, according to Theorem 1 and Theorem 2, is critical for good generalization performance. Without the regularizer, we may get a function $f$ that provides very small empirical loss, but deteriorated generalization performance, due to the large value of the ERC term. On the other hand, $L : \mathcal{Y} \times \mathcal{Y} \to \mathbb{R}$ is the loss function that measures the degree

of disagreement between $f(\boldsymbol{x})$ and the true label $y$. Obviously, many classical machine learning models fall into this framework, including SVM, ridge regression [83], One-class SVM [85], Support Vector Domain Description (SVDD) [92], etc.

Given $T$ tasks, unlike the single task scenario, which contains only one function $f$, in this case, we have a vector of functions $\boldsymbol{f} \triangleq [f_1, \cdots, f_T]'$, where each individual function $f_t$ corresponds to the $t$-th task. Besides, we are provided with a set of training data for the $T$ tasks: $\{\boldsymbol{x}_t^i, y_t^i\} \in \mathcal{X} \times \mathcal{Y}, i = 1, \cdots, N_t, t = 1, \cdots, T$, where $N_t$ is the number of training samples of the $t$-th task, and it is assumed that the training data from the $t$-th task are drawn identically and independently from a probability distribution $P_t(X, Y)$. Based on this setting, the general MTL framework can be defined as

$$\min_{\boldsymbol{f}} \; R(\boldsymbol{f}) + C \sum_{t=1}^{T} \sum_{i=1}^{N_t} L(f(\boldsymbol{x}_t^i), y_t^i) \tag{2.51}$$

In order to learn the $T$ functions simultaneously, while at the same time encourages information sharing and inductive transfer, it is necessary for the model to capture the underlying commonality and to allow information sharing among the $T$ tasks, which is usually achieved by careful design of the regularizer.

To give a concrete example, assume we have a linear model $f_t(\boldsymbol{x}) \triangleq \langle \boldsymbol{w}_t, \boldsymbol{x} \rangle, t = 1, \cdots, T$. Then, the authors in [31] proposed the following MTL model by assuming that the weight vector $\boldsymbol{w}_t$'s of all tasks are close to their mean vector $\bar{\boldsymbol{w}} \triangleq \frac{1}{T} \sum_{t=1}^{T} \boldsymbol{w}_t$:

$$\min_{\boldsymbol{w}} \sum_{t=1}^{T} \|\boldsymbol{w}_t - \bar{\boldsymbol{w}}\| + C \sum_{t=1}^{T} \sum_{i=1}^{N_t} L(f_t(\boldsymbol{x}_t^i), y_t^i) \tag{2.52}$$

where $\boldsymbol{w}$ is the collection of all the weight vectors $(\boldsymbol{w}_1, \cdots, \boldsymbol{w}_T)$. Clearly, in this MTL model, the

prior assumption of the $T$ tasks is adapted to Problem (2.52) via using this specifically designed regularizer.

To give another example, in [72], the authors assumed that all tasks share a common feature representation, by proposing the following MTL model:

$$\min_{\boldsymbol{w}} \sum_{d=1}^{D} \|\boldsymbol{w}^d\|_p + C \sum_{t=1}^{T} \sum_{i=1}^{N_t} L(f_t(\boldsymbol{x}_t^i), y_t^i) \tag{2.53}$$

where $\boldsymbol{w}^d$ is the $d$-th row of the matrix $\boldsymbol{W} \triangleq [\boldsymbol{w}_1, \cdots, \boldsymbol{w}_t]$. In Problem (2.53), the $L_p - L_1$-norm group-lasso type regularizer encourages sparsity on the vector $[\|\boldsymbol{w}^1\|_p, \cdots, \|\boldsymbol{w}^D\|_p]$, thus is able to generate a certain amount of all-zero rows for $\boldsymbol{W}$. In other words, the common feature representation amongst all tasks learned by Problem (2.53) is characterized by the rows whose elements are not all zero.

One advantage of the general MTL formulation (2.51) is that, the regularizer and/or loss term are usually flexible enough to be designed to capture different forms of task relationships, with the above two models as examples. This flexibility has lead to a fruitful of MTL models, which we refer the readers to Chapter 3 for a literature review. Also, similar to MKL, when the hinge loss function is used, the optimization of the corresponding MTL models are usually able to be split into sub-problems, some of which can take advantage of existing efficient SVM solvers. Therefore, the efficiency of the optimization algorithm can be significantly improved. This is also true when considering certain types of regression problems, such as ridge regression-based MTL models, where the closed-form solution to the ridge regression problem may be used as part of the algorithm.

*Multi-Task Multiple Kernel Learning*

The MKL method has been proved to be successful in alleviating the burden of kernel selection and providing richer space of candidate kernel functions, which is able to generate better generalization performance for kernel machines. Such a success inspired researchers to combine the advantage of MKL with MTL, which is therefore called MT-MKL. Clearly, such a combination brings all advantages to the MT-MKL framework, which is able to boost the model performance by capturing the relationship between tasks, while at the same time automatically learns a good kernel function from a rich candidate space.

To formulate a MT-MKL model, a straightforward way is to directly adapt the linear Regularization-Loss framework, and let each $\boldsymbol{w}_t \triangleq (\sqrt{\theta_1}\boldsymbol{w}_t^1, \cdots, \sqrt{\theta_M}\boldsymbol{w}_t^M) \in \mathcal{H}_{\boldsymbol{\theta}}$, where, similar to the single task MKL, $\boldsymbol{w}_t^m \in \mathcal{H}_m$, $\mathcal{H}_{\boldsymbol{\theta}} \triangleq \mathcal{H}_1 \times \cdots \times \mathcal{H}_M$ is the RKHS that is associated with kernel function $k = \sum_{m=1}^M \theta_m k_m$. Given training set $\{\boldsymbol{x}_t^i, y_t^i\} \in \mathcal{X} \times \mathcal{Y}, i = 1, \cdots, N_t, t = 1, \cdots, T$ from $T$ tasks, and $M$ pre-selected kernel functions $k_m, m = 1, \cdots, M$, a simple MT-MKL framework can be formulated as follows:

$$\begin{aligned} \min_{\boldsymbol{w},\boldsymbol{\theta}} \; & R(\boldsymbol{w}) + C \sum_{t=1}^{T} \sum_{i=1}^{N_t} L(\langle \boldsymbol{w}_t, \phi(\boldsymbol{x}_t^i) \rangle_{\mathcal{H}_{\boldsymbol{\theta}}}, y_t^i) \\ \text{s.t. } & \boldsymbol{w}_t \in \mathcal{H}_{\boldsymbol{\theta}}, t = 1, \cdots, T \\ & \boldsymbol{\theta} \in \Omega(\boldsymbol{\theta}) \end{aligned} \tag{2.54}$$

where, same as before, $\boldsymbol{w} \triangleq (\boldsymbol{w}_1, \cdots, \boldsymbol{w}_T)$. Clearly, the feature mapping $\phi : \mathcal{X} \to \mathcal{H}_{\boldsymbol{\theta}}$ is also associated with the kernel function $k$. It is worth pointing out that, although quite a different formulation to Problem (2.53), the above MT-MKL framework is also a method that encourages all tasks to share the same feature space $\mathcal{H}$. This is because the same kernel function $k$ is employed

across tasks, therefore data from all tasks are pre-processed by the same feature mapping $\phi$. This is analogous to the neural networks-based MTL model, which is introduced in Figure 2.2, where data from all tasks are pre-processed by the same matrix $W$, which is given by the inter-connection from the input layer to the shared hidden layer.

To give a more concrete MT-MKL model, a typical SVM-based MT-MKL can be formulated as follows:

$$\begin{aligned} \min_{\boldsymbol{w},\boldsymbol{b},\boldsymbol{\theta}} \ & \sum_{t=1}^{T} \frac{\|\boldsymbol{w}_t\|_{\mathcal{H}_\theta}^2}{2} + C \sum_{t=1}^{T} \sum_{i=1}^{N_t} L_{hinge}(y_t^i(\langle \boldsymbol{w}_t, \phi(\boldsymbol{x}_t^i)\rangle_{\mathcal{H}_\theta} + b_t)) \\ \text{s.t. } & \boldsymbol{w}_t \in \mathcal{H}_{\boldsymbol{\theta}}, t = 1, \cdots, T \\ & \boldsymbol{\theta} \in \Omega(\boldsymbol{\theta}) \end{aligned} \tag{2.55}$$

Similar to the single task MKL scenario, in the above problem,

$$\|\boldsymbol{w}_t\|_{\mathcal{H}_{\boldsymbol{\theta}}}^2 \triangleq \sum_{m=1}^{M} \theta_m \|\boldsymbol{w}_t^m\|_{\mathcal{H}_m}^2 \tag{2.56}$$

$$\langle \boldsymbol{w}_t, \phi(\boldsymbol{x}_t^i)\rangle_{\mathcal{H}_{\boldsymbol{\theta}}} = \sum_{m=1}^{M} \theta_m \langle \boldsymbol{w}_t^m, \phi_m(\boldsymbol{x}_t^i)\rangle_{\mathcal{H}_m} \tag{2.57}$$

both of which are not jointly convex with respect to $\boldsymbol{w}$ and $\boldsymbol{\theta}$. Again, we can equivalently transfer Problem (2.55) to a convex problem by applying variable change $\boldsymbol{w}_t^m \leftarrow \boldsymbol{w}_t^m \theta_m$:

$$\begin{aligned} \min_{\boldsymbol{w},\boldsymbol{b},\boldsymbol{\theta}} \ & \sum_{t=1}^{T} \sum_{m=1}^{M} \frac{\|\boldsymbol{w}_t^m\|_{\mathcal{H}_m}^2}{2\theta_m} + C \sum_{t=1}^{T} \sum_{i=1}^{N_t} L_{hinge}(y_t^i(\sum_{m=1}^{M} \langle \boldsymbol{w}_t^m, \phi_m(\boldsymbol{x}_t^i)\rangle_{\mathcal{H}_m} + b_t)) \\ \text{s.t. } & \boldsymbol{\theta} \in \Omega(\boldsymbol{\theta}) \end{aligned} \tag{2.58}$$

whose dual formulation is given as

$$\begin{aligned} \min_{\boldsymbol{\theta}} \max_{\boldsymbol{\alpha}} \ & \sum_{t=1}^{T} (\boldsymbol{\alpha}_t{}' \mathbf{1} - \frac{1}{2} \boldsymbol{\alpha}_t{}' \boldsymbol{Y}_t (\sum_{m=1}^{M} \theta_m \boldsymbol{K}_t^m) \boldsymbol{Y}_t \boldsymbol{\alpha}_t) \\ & \text{s.t. } \mathbf{0} \preceq \boldsymbol{\alpha}_t \preceq C\mathbf{1}, \ \boldsymbol{\alpha}_t{}' \boldsymbol{y}_t = 0, t = 1, \cdots, T \\ & \boldsymbol{\theta} \in \Omega(\boldsymbol{\theta}) \end{aligned} \tag{2.59}$$

where the $\boldsymbol{K}_m^t$ is the kernel matrix calculated by the kernel function $k_m$ on the training samples of the $t$-th task, *i.e.*, the $(i, j)$-th element of $\boldsymbol{K}_m^t$ is computed as $k_m(\boldsymbol{x}_i^t, \boldsymbol{x}_j^t)$. It can be more clearly observed from the above dual problem that this SVM-based MT-MKL formulation forces all task to share the same kernel function $k = \sum_{m=1}^{M} \theta_m k_m$, which is equivalent to sharing the same feature space.

Although a common kernel function for all tasks is a straightforward method for information sharing, many other MT-MKL models have been proposed differently. For example, in [79], a $L_1 - L_q$ Group-lasso type regularizer with $q \geq 1$ is employed:

$$\min_{\boldsymbol{w}, \boldsymbol{b}} \sum_{m=1}^{M} (\sum_{t=1}^{T} \|\boldsymbol{w}_t^m\|_{\mathcal{H}_m}^q)^{1/q} + C \sum_{t=1}^{T} \sum_{i=1}^{N_t} L_{hinge}(y_t^i (\sum_{m=1}^{M} \langle \boldsymbol{w}_t^m, \phi_m(\boldsymbol{x}_t^i) \rangle_{\mathcal{H}_m} + b_t)) \tag{2.60}$$

This regularizer is able to bring sparsity at the kernel level, and different levels of task-wise sparsity can be achieved when $q$ is assigned to different values. Its corresponding dual formulation is shown

as follows:

$$\begin{aligned} \min_{\boldsymbol{\gamma}} \max_{\boldsymbol{\alpha}} \ & \sum_{t=1}^{T} (\boldsymbol{\alpha}_t{}' \mathbf{1} - \frac{1}{2} \boldsymbol{\alpha}_t{}' \boldsymbol{Y}_t (\sum_{m=1}^{M} \gamma_t^m \boldsymbol{K}_t^m) \boldsymbol{Y}_t \boldsymbol{\alpha}_t) \\ & \text{s.t. } \mathbf{0} \preceq \boldsymbol{\alpha}_t \preceq C\mathbf{1}, \ \boldsymbol{\alpha}_t{}' \boldsymbol{y}_t = 0, t = 1, \cdots, T \\ & \boldsymbol{\gamma} \succeq \mathbf{0}, \ \sum_{m=1}^{M} [\sum_{t=1}^{T} (\gamma_t^m)^{1/s}]^s \leq 1 \end{aligned} \tag{2.61}$$

where $s \triangleq \frac{2-q}{q}$. Admittedly, the above dual formulation reveals more insight about this model. It can be seen that, compared to (2.59), instead of using a common kernel function across tasks, the above model uses a task-independent kernel function, *i.e.*, $k_t \triangleq \sum_{m=1}^{M} \gamma_t^m k_t^m$. However, the mixed-norm constraint on $\gamma_t^m$'s prevents the kernel functions of all tasks to become completely independent, thus the relation between tasks is built via the kernel functions.

In the previous section, we emphasized the benefits of the min-max MKL dual formulations. Similarly, MT-MKL models also enjoy the advantages of their corresponding min-max dual counterparts, which can be clearly seen from the above example: In Problem (2.61), we easily observe that this model utilizes task-independent kernel functions, which is not an obvious fact in the primal formulation (2.60). Additionally and arguably more importantly, in the dual domain, it is easy to modify the model for different needs, *e.g.*, allow tasks to partially share the same kernel by letting $k_t \triangleq \sum_{m=1}^{M} (\zeta_m + \gamma_t^m) k_t^m$, which is not obvious how to formulate in the primal domain.

On the other hand, similar to MKL, an obstacle that prevents people from using advanced MT-MKL models, which are usually formulated as min-max problems, is that it is usually difficult to derive appropriate algorithm to solve these min-max problems. It is common that each MT-MKL model is featured with an algorithm that is tailored for the corresponding model, and most of these algorithms are not only complicated, but also hard to understand and implement. Therefore, it leaves

the users almost impossible to compare several different models, due to the algorithmic complexity. Therefore, as stated in the previous section, such difficulties inspired us to develop a general purpose, easy-to-implement algorithm to solve most MKL, kernel-based MTL and MT-MKL problems in the dual domain. This algorithm is introduced in Chapter 5.

*Generalization Bound*

As stated at the beginning of this section, MTL has been shown to be advantageous in many real world application problems by providing better generalization performance. At the same time, a large number of MTL models have been developed with different prior assumptions about task relationships. However, while the efficacy of the MTL learning paradigm has been demonstrated, it is crucial to understand the reason of its good performance. As introduced in the previous sections, since the generalization bound serves as an important tool in revealing the capability (in terms of generalization performance) of machine learning models, many MTL research works have been focused on this direction. In the following, we introduce a general result, which not only explains the efficacy of MTL paradigm, but also serves as a foundation of other research on MTL generalization bound.

**Theorem 7.** *Let* $\{\boldsymbol{x}_t^i, y_t^i\} \in \mathcal{X} \times \mathcal{Y}, i = 1, \cdots, N, t = 1, \cdots, T$ *be a set of training data, and the data of the* $t$*-th task are drawn identically and independently from the probability distribution* $P_t(X, Y)$*. Let* $\mathcal{F}$ *be the HS of* $\mathbb{R}^T$*-valued functions* $\boldsymbol{f} \triangleq [f_1, \cdots f_T]'$*, with* $f_t : \mathcal{X} \mapsto \mathbb{R}$*. Suppose that* $L : \mathbb{R} \mapsto [0, 1]$ *be a fixed Lipschitz continuous loss function with Lipschitz constant* $\gamma$ *and dominates the* $0/1$ *loss function* $1_{(-\infty,0]}(\cdot)$*. Then, for any* $\delta > 0$*, with probability at least* $1 - \delta$*, each of the following holds for all* $\boldsymbol{f} \in \mathcal{F}$

$$er(\boldsymbol{f}) \leq \hat{er}_L(\boldsymbol{f}) + \frac{1}{\gamma}\hat{R}(\mathcal{F}) + \sqrt{\frac{9 \log \frac{2}{\delta}}{2TN}} \tag{2.62}$$

$$er(\boldsymbol{f}) \leq \hat{er}_L(\boldsymbol{f}) + \frac{1}{\gamma} R(\mathcal{F}) + \sqrt{\frac{\log \frac{2}{\delta}}{2TN}} \tag{2.63}$$

*The MTL expected error is defined as*

$$er(\boldsymbol{f}) \triangleq \frac{1}{T} \sum_{t=1}^{T} E_{(X_t,Y_t)} \{ \mathbf{1}_{(-\infty,0]}(Y_t f_t(X_t)) \} \tag{2.64}$$

*where* $(X_t, Y_t) \sim P_t(X, Y)$*, and the corresponding MTL empirical loss is defined as*

$$\hat{er}_L(\boldsymbol{f}) \triangleq \frac{1}{TN} \sum_{t=1}^{T} \sum_{i=1}^{N} L(y_t^i f_t(\boldsymbol{x}_t^i)) \tag{2.65}$$

*Finally,* $\hat{R}(\mathcal{F})$ *and* $R(\mathcal{F})$ *are the ERC and RC for MTL problems correspondingly, defined as*

$$\hat{R}(\mathcal{F}) \triangleq E_\sigma \{ \sup_{\boldsymbol{f} \in \mathcal{F}} \frac{2}{TN} \sum_{t=1}^{T} \sum_{i=1}^{N} \sigma_t^i f_t(\boldsymbol{x}_t^i) \} \tag{2.66}$$

$$R(\mathcal{F}) \triangleq E_{\boldsymbol{x}} E_\sigma \{ \sup_{\boldsymbol{f} \in \mathcal{F}} \frac{2}{TN} \sum_{t=1}^{T} \sum_{i=1}^{N} \sigma_t^i f_t(\boldsymbol{x}_t^i) \} \tag{2.67}$$

*where the* $\sigma_t^i$*'s are i.i.d. Bernoulli* $\left(\frac{1}{2}\right)$ *random variables with sample space* $\{-1, +1\}$*.*

The above theorem, which can be simply proved based on Theorem 16 and 17 of [65], is an extension of Theorem 2 to the MTL setting. Note that it is a slightly more general than Theorem 18 of [65], since the latter one only considered the margin loss instead of a general loss function $L$. It is worth to mention that, in this theorem, we assumed equal number of training samples for

all tasks. This assumption is made only to keep the notations uncluttered, without preventing it from being generalized to unequal number of training samples across tasks. For similar reason, the range of the loss function $L$ is assumed to be upper bounded by 1. Finally, we only focus on the ERC-based bound, since the RC is distribution-dependent, and is difficult to calculate or estimate. In this research, we stick with these assumptions without further reiteration.

The above theorem reveals the following insights: The difference between the generalization error and the empirical loss is upper bounded by the last two terms in (2.62). Importantly, the last term is of order $O(\sqrt{\frac{1}{TN}})$. Therefore, as the number of tasks increases, the number of training samples needed from each task can be significantly decreased. If similar property is held in the ERC term, then we are able to get even lower generalization bound value by increasing the number of tasks. Therefore, it not only explains why MTL models usually achieve better performance than single-task approaches, but also indicates that we need to carefully choose the HS $\mathcal{F}$, so that the ERC term decreases fast when $T$ increases.

Admittedly, in some cases where the tasks are "irrelevant" to each other, training them simultaneously with information being shared across tasks may cause large the empirical loss $\hat{er}_L(\boldsymbol{f})$ and deteriorates the generalization performance of each task, a phenomenon that is referred to as "*negative transfer*". In this case, MTL may be less beneficial. Therefore, it is necessary to either carefully choose tasks to avoid negative transfer, or develop powerful MTL models that are able to recognize irrelevant tasks and train them independently.

During the past decade, many researchers have devoted to calculating or upper bounding the ERC term of different HSs. These theoretical research works not only delivered useful insights about the corresponding MTL HSs, but also inspired the invention of better HSs with lower generalization bound values and MTL models with better generalization performance, including the first part of works in this dissertation, which are introduced in Chapter 4.

In the next chapter, we give a literature review to MTL, including research works on generalization bound, models and algorithms.

# CHAPTER 3: LITERATURE REVIEW

## Generalization Bound

The first work that systematically discussed the generalization performance of MTL paradigm appeared in [9]. Although the generalization bounds derived in this paper are based on the concept of "covering number", thus are of combinatorial nature and potentially loose, it revealed the fact that it is beneficial to conduct MTL paradigm in terms of sample complexity. On the other hand, modern MTL theoretical research is based on the Rademacher complexity, majorly because of its tightness. Given a MTL model

$$\min_{\boldsymbol{f}} \; R(\boldsymbol{f}) + C \sum_{t=1}^{T} \sum_{i=1}^{N} L(f_t(\boldsymbol{x}_t^i), y_t^i) \tag{3.1}$$

the corresponding HS is

$$\mathcal{F} \triangleq \{\boldsymbol{f} = [f_1, \cdots, f_T]' : R(\boldsymbol{f}) \leq r\} \tag{3.2}$$

For different the regularizers, various Rademacher complexity-based generalization bounds have been studied.

In [66], the authors considered a simple linear model, where $f_t(\boldsymbol{x}) = \langle \boldsymbol{w}_t, \boldsymbol{x} \rangle$, $\boldsymbol{w}_t \in \mathbb{R}^m$, and two different HSs, namely

$$\mathcal{F} \triangleq \{\boldsymbol{x} \to [\langle \boldsymbol{w}_1, \boldsymbol{x} \rangle, \cdots, \langle \boldsymbol{w}_T, \boldsymbol{x} \rangle]' : \frac{\|\boldsymbol{A}^{\frac{1}{2}} \boldsymbol{W}\|_2}{\sqrt{m}} \leq B\} \tag{3.3}$$

and

$$\mathcal{F} \triangleq \{\boldsymbol{x} \to [\langle \boldsymbol{w}_1, \boldsymbol{x} \rangle, \cdots, \langle \boldsymbol{w}_T, \boldsymbol{x} \rangle]' : \frac{\|\boldsymbol{W}\|_q}{\sqrt{m}} \leq B\} \tag{3.4}$$

where $\boldsymbol{W} = [\boldsymbol{w}_1, \cdots, \boldsymbol{w}_T]$, $\boldsymbol{A}$ is any positive definite operator in $\mathbb{R}^m$, $1 \leq q \leq \frac{4}{3}$, and $B$ is set to control the capacity of the HSs. In the first HS, if $\boldsymbol{A} = \boldsymbol{I}$, the identity matrix, then it degrades to independent learning. Therefore, in order to conduct MTL, meaningful configuration of $\boldsymbol{A}$ is required. For example, as the authors pointed out, when $\boldsymbol{A} \triangleq \boldsymbol{L} + \eta \boldsymbol{I}$, where $\boldsymbol{L}$ is the Laplacian on a graph of $m$ vertices and $\eta > 0$, the HS corresponds to the graph regularization model [30]. On the other hand, in (3.4), $\boldsymbol{W}_q \triangleq (\sum_{l=1}^{M} \|\boldsymbol{w}^l\|^q)^{\frac{1}{q}}$, where $\boldsymbol{w}^l$ is the $l$-th row of $\boldsymbol{W}$. In this case, since $q$ is close to 1, it encourages sparse rows for the matrix $\boldsymbol{W}$, which forces tasks to share the same subspace, characterized by the non-zero rows of $\boldsymbol{W}$.

In [47], a more general class of functions is studied. Similar to the previous work, linear functions parametrized by the weights $\boldsymbol{w}_t$'s are assumed. Then, the authors considered the following HS

$$\mathcal{F} \triangleq \{\boldsymbol{x} \to [\langle \boldsymbol{w}_1, \boldsymbol{x} \rangle, \cdots, \langle \boldsymbol{w}_T, \boldsymbol{x} \rangle]' : F(\boldsymbol{W}) \leq F_{max}\} \tag{3.5}$$

where the function $F(\cdot)$ is $\beta$-*strongly convex* with respect to a norm $\|\cdot\|$, a condition that is satisfied in many situations, including when $F(\boldsymbol{W}) = \|\boldsymbol{W}\|_{1,1}$, $\|\boldsymbol{W}\|_{1,2}$, $\|\boldsymbol{W}\|_{2,1}$ and $\|\boldsymbol{W}\|_{2,2}$. In these different cases, sparsity of $\boldsymbol{W}$ in the row level and (or) the column level may be achieved, depending which one of the four functions is used.

Different to [47], in [68], the authors defined the following norm

$$\|\boldsymbol{w}\|_{\mathcal{M}} \triangleq \inf\{\sum_{M\in\mathcal{M}} \|\boldsymbol{v}_M\| : \boldsymbol{v}_M \in \mathcal{H}, \sum_{M\in\mathcal{M}} M\boldsymbol{v}_M = \boldsymbol{w}\} \tag{3.6}$$

where $\boldsymbol{w}$ is the collection of $\boldsymbol{w}_1, \cdots, \boldsymbol{w}_T$, $\mathcal{M}$ is an almost countable set of symmetric bounded linear operators on $\mathcal{H}$. Then, the authors discussed the generalization bound based on the HS

$$\mathcal{F} \triangleq \{\boldsymbol{x} \to [\langle \boldsymbol{w}_1, \boldsymbol{x}\rangle, \cdots, \langle \boldsymbol{w}_T, \boldsymbol{x}\rangle]' : \|\boldsymbol{w}\|_{\mathcal{M}} \leq 1\} \tag{3.7}$$

Since the above defined norm covers several widely used regularizers, such as Lasso, Group-Lasso, weighted Group-Lasso, etc., the generalization bound that is derived by the authors is also applicable to many Regularization-Loss methods.

Finally, in [75], the authors specifically considered the trace norm regularization, due to its capability in minimizing the rank of the weight matrix, and the existence of efficient algorithms [41] [46] [74] [29]. The HS is given as

$$\mathcal{F} \triangleq \{\boldsymbol{x} \to [\langle \boldsymbol{w}_1, \boldsymbol{x}\rangle, \cdots, \langle \boldsymbol{w}_T, \boldsymbol{x}\rangle]' : tr((\boldsymbol{W}'\boldsymbol{W})^{\frac{1}{2}} \leq B\sqrt{T}\} \tag{3.8}$$

For the theoretical works that are reviewed above, the task relationships are captured via the regularizer $R(\boldsymbol{f})$. In this way, information learned by each task is shared to the other ones.

On the other hand, another angle of applying information sharing amongst tasks is to pre-process the data from all tasks by a common processor, and subsequently, a linear model is learned based on the processed data. One typical model that follows this approach is the neural networks-based

method that is introduced in [16] and Figure 2.2; some other examples include the ones proposed in [4, 30, 6, 21, 53, 40].

One application of this learning framework is subspace learning, where data of each task are projected to a common subspace by an operator $A$, and then the $\boldsymbol{w}_t$'s are learned in that subspace. Another particularly straightforward and useful adaptation of this framework is kernel-based MTL. In this situation, the role of the operator $A$ is assumed by the non-linear feature mapping $\phi$ associated with the kernel function that is chosen by the user. In this case, all data are pre-processed by a *common* kernel function, which is pre-selected or learned during the training phase, while the $\boldsymbol{w}_t$'s are then learned in the corresponding RKHS.

In [4], the authors assumed that each task $t$ is featured with the linear mapping $f_t(\boldsymbol{x}) = \langle \boldsymbol{u}_t, \Phi(\boldsymbol{x}) \rangle + \langle \boldsymbol{v}_t, \boldsymbol{\Theta}\Psi(\boldsymbol{x}) \rangle$, where $\boldsymbol{u}_t \in \mathbb{R}^m$ and $\boldsymbol{v}_t \in \mathbb{R}^n$ are the weights associated to the linear model of $t$-th task, and $\Phi : \mathcal{X} \mapsto \mathbb{R}^m$, $\Psi : \mathcal{X} \mapsto \mathbb{R}^p$ are two pre-selected feature mappings. Finally, the matrix $\boldsymbol{\Theta} \in \mathbb{R}^{n \times p}$ is learned during the training stage. Then, the author considered the following HS

$$\begin{aligned} \mathcal{F} \triangleq \{ \boldsymbol{x} \to \{ \langle \boldsymbol{u}_t, \Phi(\boldsymbol{x}) \rangle + \langle \boldsymbol{v}_t, \boldsymbol{\Theta}\Psi(\boldsymbol{x}) \rangle \}_{t=1}^T : \\ \|\boldsymbol{u}_t\|_2 \leq \frac{A}{\sup_{\boldsymbol{x}} \|\Phi(\boldsymbol{x})\|_2}, \|\boldsymbol{v}_t\|_2 \leq \frac{B}{\sup_{\boldsymbol{x}} \|\Psi(\boldsymbol{x})\|_2}, t = 1, \cdots, T, \boldsymbol{\Theta}\boldsymbol{\Theta}' = \boldsymbol{I} \} \end{aligned} \tag{3.9}$$

where the constants $A$ and $B$ are used to control the capacity of the HS. It is not difficult to see that, in this HS, data are simultaneously pre-processed by the two feature mappings, $\Phi$ and $\Psi$. Besides, tasks are connected via the common matrix $\boldsymbol{\Theta}$, without which all tasks will be independent to each other.

Another work that discussed the generalization bound of this category of MTL approaches is given in [65]. For a set $\mathcal{A}$ of bounded self-adjoint linear operators on $\mathcal{X}$ and $T$ linear functions with

weights $\boldsymbol{w}_t$'s, the HS is given as

$$\mathcal{F} = \{\boldsymbol{x} \mapsto [\langle \boldsymbol{w}_1, A\boldsymbol{x}\rangle, \cdots, \langle \boldsymbol{w}_T, A\boldsymbol{x}\rangle]' : \|\boldsymbol{w}_t\|^2 \leq R, t = 1, \cdots, T, A \in \mathcal{A}\} \tag{3.10}$$

Clearly, in this HS, data are pre-processed by the operator $A$ to a common space, as explained above. By either cleverly choosing $A$ beforehand or by learning $A \in \mathcal{A}$, it is expected that a tighter generalization bound can be attained compared to learning each task independently. It is straightforward to see that pre-selecting $A$ beforehand is a special case of learning $A \in \mathcal{A}$, *i.e.*, pre-selecting $A$ is equivalent to $\mathcal{A} = \{A\}$.

However, as introduced in Chapter 1, the research of generalization bound on this type of MTL methods only covers very limited situations, primarily due to two drawbacks of the work in [65] and [4]: First, neither of these two papers include the MTL models where a common kernel function is learned for all tasks, *e.g.*, the MT-MKL models, since the feature mappings that considered in these two papers are all assumed to be linear. Second, both of these two papers unnecessarily set all the task weights to be constrained in norm balls, whose radii are the same. However, letting each task have its own radius for the corresponding norm-ball constraint yields a more flexible HS, and, in practice, may lead to an improved generalization bound and performance.

Such limitations inspired us to generalize the HS $\mathcal{F}$, particularly for kernel-based multi-task classification problems, by considering the common operator $\phi$ (which is associated with a kernel function) for all tasks and by imposing norm-ball constraints with different radii for each task that are learned during the training process, instead of being chosen prior to training, as discussed in Chapter 4.

## Models

Although started as a neural network-based learning algorithm, MTL has been widely studied based on many machine learning frameworks, such as Gaussian process [13] [12] [96] [102], probabilistic model [99] [106] [103] [118], metric learning [73], etc. Most of the MTL models are formulated based as the minimization of Regularization-Loss problems (2.51), given $T$ linear functions $f_t$ with weights $\boldsymbol{w}_t, t = 1, \cdots, T$, as discussed in Chapter 2. In this section, we give a literature review to MTL models.

### *Exploring Task Relationship*

A fundamental assumption for MTL is that there exist some underlying relations between all (or some) tasks. Therefore, one crucial objective that MTL models should achieve is to capture such relations. For most previous research works, this is accomplished by making a priori assumptions for task relationships, and implement these assumptions in the objective function, by specifying the corresponding regularizers and loss functions. On the other hand, some other paper developed MTL models that are able to learn task relationships automatically. This sub-section reviews both of the two types of MTL models.

In [31], the authors assumed that all task weights $\boldsymbol{w}_t$'s to be close to their mean vector $\bar{\boldsymbol{w}} \triangleq \frac{1}{T}\sum_{t=1}^{T} \boldsymbol{w}_t$, by letting the regularizer to be

$$R(\boldsymbol{f}) \triangleq \sum_{t=1}^{T} \|\boldsymbol{w}_t - \bar{\boldsymbol{w}}\|^2 \tag{3.11}$$

This idea is generalized in [115], which assumed a clustered structure amongst tasks, and each task weight is close to the cluster center, instead of the global average of all task weights. In their

model, the regularizer is specified as

$$R(\boldsymbol{f}) \triangleq \sum_{j=1}^{J} \sum_{t \in G_j} \|\boldsymbol{w}_t - \bar{\boldsymbol{w}}_j\|^2 \tag{3.12}$$

where $G_j$ is the set of indices of tasks that belong to the $j$-th cluster, and $\bar{\boldsymbol{w}}_j$ is the $j$-th cluster mean. A similar task clustering strategy is also studied in [114], which assumes the clustered structure appears on the feature level across tasks. In this paper, function weight for each task is decomposed into two components $\boldsymbol{w}_t \triangleq \boldsymbol{u}_t + \boldsymbol{v}_t$, and the feature level clustering is achieve via a specially defined regularizer on the matrix $\boldsymbol{U} \triangleq [\boldsymbol{u}_1, \cdots, \boldsymbol{u}_T]$:

$$\|\boldsymbol{U}\|_{clus} \triangleq \sum_{d=1}^{D} \sum_{t_1 < t_2} |u_{t_1}^d - u_{t_2}^d| \tag{3.13}$$

where $\boldsymbol{U} \triangleq \{u_t^d\}, d = 1, \cdots, D, t = 1, \cdots, T$. For each feature $d$, the above norm encourages close distance between $u_{t_1}^d$ and $u_{t_2}^d$, which leads to clustered structure for each feature.

The above weight decomposition trick has been adapted in several other papers. In [113], it is assumed that each task weight is composed with a vector $\boldsymbol{w}_0$, which is common across tasks, and a task specific element $\boldsymbol{v}_t$, *i.e.*, $\boldsymbol{w}_t \triangleq \boldsymbol{w}_0 + \boldsymbol{v}_t$. With the following regularizer

$$R(\boldsymbol{f}) \triangleq \frac{\lambda_1}{T} \sum_{t=1}^{T} \|\boldsymbol{v}_t\|^2 + \lambda_2 \|\boldsymbol{w}_0\|^2 \tag{3.14}$$

the shared structure amongst tasks is able to be captured by $\boldsymbol{w}_0$.

In [36], the task weight $\boldsymbol{w}_t$ is decomposed in the same way as [114]. However, such a decomposition is utilized for detecting the "outlier tasks", *i.e.*, tasks that are considered not dissimilar to the

other tasks. This outlier task detection is implemented by the regularizer

$$R(\boldsymbol{f}) \triangleq \lambda_1 \|\boldsymbol{U}\|_{2,1} + \lambda_2 \|\boldsymbol{V}'\|_{2,1} \tag{3.15}$$

where the above $L_2 - L_1$-norm regularizer is defined as $\|\boldsymbol{U}\|_{2,1} \triangleq \sum_{d=1}^{D} \|\boldsymbol{u}^d\|_2$, and $\boldsymbol{u}^d$ is the $d$-th row of the matrix $\boldsymbol{U}$. Obviously, the first term $\|\boldsymbol{U}\|_{2,1}$ encourages shared feature representation among tasks, given by the non-zero rows of $\boldsymbol{U}$. Also, the second term $\|\boldsymbol{V}'\|_{2,1}$ encourages sparsity in the column level of $\boldsymbol{V}$, thus the tasks that correspond to all-zero columns of $\boldsymbol{V}$ share the common feature representation given by $\boldsymbol{U}$, while the other tasks are identified as outliers and are characterized by their own feature representation, which is given by the combination of $\boldsymbol{U}$ and $\boldsymbol{V}$. This task outlier identification method by utilizing the $\|\boldsymbol{V}'\|_{2,1}$ regularizer is also employed in [22]. However, instead of $\|\boldsymbol{U}\|_{2,1}$, the authors in [22] proposed the using of trace norm (or nuclear norm) on $\boldsymbol{U}$, *i.e.*, $\|\boldsymbol{U}\|_* \triangleq \mathit{trace}(\sqrt{\boldsymbol{U}'\boldsymbol{U}})$, in order to encourage a low-rank matrix, thus a low dimensional feature space.

In [4], such a decomposition is not performed on the task weight level, instead, it is the task function decomposed into two parts. Given a training sample $\boldsymbol{x}$, the $t$-th task function value is evaluated as

$$f_t(\boldsymbol{x}) \triangleq \langle \boldsymbol{u}_t, \Phi(\boldsymbol{x}) \rangle + \langle \boldsymbol{v}_t, \boldsymbol{\Theta}\Psi(\boldsymbol{x}) \rangle \tag{3.16}$$

where, $\Phi$ and $\Psi$ are two feature mappings, and $\boldsymbol{\Theta}$ is shared across tasks, thus the tasks are assumed to be related via this common matrix. The MTL model in this paper is further studied in [21] by providing a convex relaxation of this approach.

Similar to [22], the authors in [53] considered learning multiple tasks with outlier tasks being

aware: It is assumed the existence of $K$ *latent* basis tasks, parametrized by $\boldsymbol{l}_k$, and each *observed* task is represented as linear combination of the latent tasks, *i.e.*, $\boldsymbol{w}_t \triangleq \sum_{k=1}^{K} s_k \boldsymbol{l}_k$, or, $\boldsymbol{W} = \boldsymbol{L}\boldsymbol{S}$, where $\boldsymbol{L} \in \mathbb{R}^{d\times K}$, $\boldsymbol{S} \in \mathbb{R}^{K\times T}$. Then, the authors introduced $\|\boldsymbol{S}\|_{1,1}$ regularizer for outlier task detection, similar to (3.15). On the other hand, the $\|\boldsymbol{L}\|_*$ regularizer is introduced to promote a low-dimensional feature space. This method is further studied theoretically and extended to the context of transfer learning in [67].

In [116], $T$ regression tasks that are assumed to be similar to each other are considered. The following regularizer is introduced in order to preserve the smoothness between tasks:

$$R(\boldsymbol{f}) \triangleq \sum_{t=1}^{T-1} \|\boldsymbol{w}_t - \boldsymbol{w}_{t+1}\|_2^2 \tag{3.17}$$

As another similar work, in [49], the authors assumed the availability of a task network, which represents the relatedness between tasks. Then, the distance between weights of the similar tasks are minimized, by proposing the following constraints on the weights

$$\|\boldsymbol{w}_{i_k} - \boldsymbol{w}_{j_k}\| \leq \rho, \forall k = 1, \cdots, K \tag{3.18}$$

where $K$ is the number of edges in the network, and $i_k$ and $j_k$ are the indices of the two tasks connected by the edge.

Unlike the previous approach, which assumes the task relation is known a priori, in [32], the authors proposed a model to extract the task relation during the training stage, instead of set prior

to training. The following regularizer is invented to learn the task relation

$$R(\boldsymbol{f}) \triangleq \lambda_1 \|\boldsymbol{W}\|_{1,1} + \frac{\lambda_2}{2} trace(\boldsymbol{W}\boldsymbol{\Omega}^{-1}\boldsymbol{W}') \tag{3.19}$$

where the matrix $\boldsymbol{\Omega} \in \mathbb{R}^{T\times T}$, which is learned during the training phase, represents the relationship between tasks. This approach is also studied in [111] and [112], and is incorporated with the boosting algorithm in [110]. As a similar approach, the authors of [81] proposed an online algorithm to solve the MTL problem with the task relationship matrix $\boldsymbol{A}$. This is achieved via the regularizer $\boldsymbol{w}'\boldsymbol{A}_{\otimes}\boldsymbol{w}$, where $\boldsymbol{w}$ is the concatenation of all task weights $\boldsymbol{w}_t$'s, $\boldsymbol{A}_{\otimes} \triangleq \boldsymbol{A} \bigotimes \boldsymbol{I}$, and $\boldsymbol{A} \in \mathbb{R}^{T\times T}$.

On the other hand, the method proposed in [7] is designed to capture the structures between the features, which are shared across tasks, by introducing a matrix $\boldsymbol{D} \in \mathbb{R}^{T\times T}$ and employing the following regularizer

$$R(\boldsymbol{f}) \triangleq trace(F(\boldsymbol{D})\boldsymbol{W}\boldsymbol{W}') \tag{3.20}$$

where $\boldsymbol{D} = \boldsymbol{U}\boldsymbol{\Lambda}\boldsymbol{U}'$ is the eigen-value decomposition of matrix $\boldsymbol{D}$, $\boldsymbol{\Lambda} \triangleq diag(\lambda_1, \cdots, \lambda_D)$ is the diagonal matrix with the eigen-values of $\boldsymbol{D}$ being the diagonal elements, $F(\boldsymbol{D}) \triangleq \boldsymbol{U}F(\boldsymbol{\Lambda})\boldsymbol{U}'$, and $F(\boldsymbol{\Lambda}) \triangleq diag(f(\lambda_1), \cdots, f(\lambda_D))$.

Finally, task relationship is learned in a different manner in [48]. In this work, $G$ task groups are assumed, and each task is considered belong to one of the $G$ groups. For each group $g$, a diagonal matrix $\boldsymbol{Q}_g \in \mathbb{R}^{T\times T}$ is learned during the training stage, indicating the group assignment of each

task. Specifically, if task $t$ belongs to group $g$, then $\boldsymbol{Q}_g(t,t) = 1$, otherwise 0. With the regularizer

$$R(\boldsymbol{f}) \triangleq \sum_{g=1}^{G} \|\boldsymbol{W}\boldsymbol{Q}_g\|_* \tag{3.21}$$

and constraint $\sum_{g=1}^{G} \boldsymbol{Q}_g = \boldsymbol{I}$, the weight matrix $\boldsymbol{W}$ and group assignment matrices $\boldsymbol{Q}_g$'s are learned simultaneously during training.

*Feature Learning*

Information sharing is critical for learning multiple tasks simultaneously, otherwise it is equivalent to learning each task independently. One most straightforward and heavily studied method for information sharing is to let tasks (partially) share a common feature representation. In this subsection, we review the MTL models that fall into this category.

The most commonly utilized feature learning method is to employ the $L_2 - L_1$-norm regularizer on the weight matrix $\boldsymbol{W}$. As a typical example, in [5], the $L_2 - L_1$-norm regularizer is utilized:

$$R(\boldsymbol{f}) \triangleq \|\boldsymbol{W}\|_{2,1} \triangleq \sum_{d=1}^{D} \|\boldsymbol{w}^d\|_2 \tag{3.22}$$

where $\boldsymbol{w}^d$ is the $d$-th row of the matrix $\boldsymbol{W}$, and $D$ is the number of features. By conducting the $L_1$-norm on the row level of $\boldsymbol{W}$, it is expected that most rows to be all-zero vectors, and therefore, the non-zero rows construct a low-dimensional feature space that is shared for all tasks. Due to the simplicity and efficacy of this regularizer, it has been studied in many other works, including [64] [62] [72] [104] [101].

In [42], the authors proposed a "dirty" MTL model, *i.e.*, a combination of two regularizers is used, instead of employing only one regularizer, which is referred to "clean" model by the authors. In this paper, instead of using the $L_2 - L_1$-norm regularizer, the authors switched to the $L_\infty - L_1$-norm to encourage sparsity in the row level of $\boldsymbol{W}$ for feature learning. Besides, to further promote sparse feature representation, a $L_1 - L_1$-norm regularizer is added for element-wise sparsity in $\boldsymbol{W}$, resulting in the following regularizer

$$R(\boldsymbol{f}) \triangleq \lambda_1 \|\boldsymbol{W}\|_{1,1} + \lambda_2 \|\boldsymbol{W}\|_{\infty,1} \tag{3.23}$$

Besides [42], the learning with $L_\infty - L_1$-norm is also studied in [59] [23].

Similar to [42], a combination of $L_1 - L_1$-norm and $L_2 - L_1$-norm regularizer is considered in [100] for simultaneous row level and element-wise sparsity on $\boldsymbol{W}$, and an online algorithm is derived to solve the optimization problem. A similar problem that also uses $L_1 - L_1$-norm and $L_2 - L_1$-norm regularizer for regression problems is studied in [76], with an Iteratively Reweighted Least Square algorithm proposed to solve the optimization problem.

In [37], the $L_2 - L_1$-norm regularizer is combined with the Frobenious norm, in order to control the capacity of the matrix $\boldsymbol{W}$. The proposed MTL model is used to solve the multi-task Calibrated Multivariate Regression (CMR) problem, which, unlike regular regression problems that uses $L_2$ loss function, a $L_1$ loss is utilized. Similarly, the addition of the $L_2 - L_1$-norm and the trace norm regularizer is studied in [22], in order to learn a low-rank weight matrix $\boldsymbol{W}$, or, in other words, a low dimensional feature space that is shared across tasks. The learning with trace norm regularization is also discussed in [46] [74] [29].

Although the $L_1 - L_1$-norm regularizer is powerful in inducing sparsity on $\boldsymbol{W}$, other attempts for better sparsity property have been studied. In [35], the so-called *capped*-$L_1 - L_1$ regularization is

thoroughly studied, which is defined as

$$R(\boldsymbol{f}) \triangleq \sum_{d=1}^{D} \min(\|\boldsymbol{w}^d\|_1, \theta) \tag{3.24}$$

This regularizer prevents the $L_1$-norm of each row of $\boldsymbol{W}$ from being larger than the pre-defined parameter $\theta$, thus potentially leads to sparser structure on the row level of $\boldsymbol{W}$. This regularizer is also systematically studied in [108] and [109].

As another attempt for sparsity inducing feature learning MTL method, the authors in [20] decomposed the weight matrix $\boldsymbol{W}$ to $\boldsymbol{U}$ and $\boldsymbol{V}$, and conducted $L_0$-norm regularizer, *i.e.*, the total number of non-zero elements, on $\boldsymbol{V}$, for sparsity promotion. Besides, the constraint $\textit{rank}(\boldsymbol{U}) \leq \lambda$ leads to a low dimensional feature space.

In [117], the authors argued that in some cases, such as when tasks are not all relevant to each other, it may not be appropriate to let tasks to share the same feature representation. To promote partial feature sharing, the authors proposed the so-called "exclusive lasso" regularizer

$$R(\boldsymbol{f}) \triangleq \sum_{d=1}^{D} (\sum_{t=1}^{T} |w_t^d|)^2 \tag{3.25}$$

which, in essence, is a $L_1 - L_2$-norm regularizer, on the weight matrix $\boldsymbol{W}$. Such a regularizer encourages sparsity in the task level for each feature.

Finally, as a different strategy, the authors in [80] tries to improve the performance of the MTL model by adding $S$ unrelated auxiliary tasks, whose weights are represented as $\boldsymbol{v}_s, s = 1, \cdots, S$. Then the authors designed a sophisticated regularizer for $\boldsymbol{W}$ and $\boldsymbol{V}$, so that $\boldsymbol{W}$ and $\boldsymbol{V}$ don't share feature representation. By validating with a series of experimentals, it is shown in this paper that

such a MTL setting, with the help of the auxiliary tasks, is beneficial in terms of generalization performance.

*Kernel-based MTL and MT-MKL*

Theoretically, most MTL approaches can be extended to the corresponding non-linear models by applying the representer theorem in a RKHS. However, in many cases, simply adapting the representer theorem to kernelize the MTL models may lead to optimization problems that are difficult to solve. Therefore, many research works have focused on deriving kernel-based methods tailored for MTL, instead of simply adapting the representer theorem.

In [70], the learning of $T$ linear functions is modeled as the problem of learning the vector-valued function $\boldsymbol{f} : \mathcal{X} \rightarrow \mathbb{R}^T$, while the matrix-valued kernel function is utilized to kernelize the MTL models. Similarly, such kernel function is adapted in [28], by specifying the kernel function to be a product between scalar-valued kernel function (called input kernel) and a positive semi-definite matrix (referred to as the output kernel), which is learned during the training stage. Therefore, the function $\boldsymbol{f}$ is learned via the learning of the output kernel. This model is not only kernelized by the matrix-valued kernel function, but also able to capture the task relationship by learning the output kernel. The theory of using matrix-valued kernel function in kernelization of MTL methods is thoroughly studied in [15].

In [60], the authors constructed the "multi-task finite rank kernels" based on the operator-valued kernel function, a general extension of matrix-valued kernel function. Several basic properties of the multi-task finite rank kernels are established, and, the usefulness is demonstrated. Besides, the authors specifically discussed polynomial kernel, which has been shown to be very powerful, and proved that any continuous kernel function can be uniformly approximated by the polynomial kernel function, which is a very interesting result.

On the other hand, in [30], the authors defined a type of kernel function specifically for MTL problems. Given task $t_1$, $t_2$ and training data $\boldsymbol{x}_1$, $\boldsymbol{x}_2$, a kernel function is defined as $k((\boldsymbol{x}_1, t_1), (\boldsymbol{x}_2, t_2)) \triangleq \boldsymbol{x}_1' \boldsymbol{B}_{t_1}' \boldsymbol{B}_{t_2} \boldsymbol{x}_2$, where $\boldsymbol{B}_{t_i}$ is the feature mapping matrix for the $t_i$'s task. Based on this definition of kernel function, a Regularization-Loss MTL framework is studied. This kernelization strategy is further studied by [89], where a general MTL framework is proposed for regression problems.

Instead of defining kernel functions specifically for MTL problems, as the methods described above, in many scenarios, regular real-valued kernel function is utilized. In [90], the proposed MTL model tries to capture the commonality between tasks and the task specific characteristics. Based on the representer theorem, this is achieved by defining the function $f_t$ in the RKHS as

$$f_t(\boldsymbol{x}) \triangleq \boldsymbol{\alpha}' \boldsymbol{K}_{\boldsymbol{x}C} + \boldsymbol{\beta}_t' \boldsymbol{K}_{\boldsymbol{x}J_t} \tag{3.26}$$

where $\boldsymbol{\alpha} \in \mathbb{R}^{NT}$ and $\boldsymbol{\beta}_t \in \mathbb{R}^N, t = 1, \cdots, T$ are variables to be learned during the training stage, and $\boldsymbol{K}_{\boldsymbol{x}C} \triangleq [k(\boldsymbol{x}, \boldsymbol{x}_1^1), \cdots, k(\boldsymbol{x}, \boldsymbol{x}_T^N)]' \in \mathbb{R}^{NT}$, $\boldsymbol{K}_{\boldsymbol{x}J_t} \triangleq [k(\boldsymbol{x}, \boldsymbol{x}_t^1), \cdots, k(\boldsymbol{x}, \boldsymbol{x}_t^N)]' \in \mathbb{R}^N$. Clearly, the commonality between tasks is captured by the first term, while the task specific characteristics is extracted via the second term.

On the other hand, as a more commonly utilized kernelization method, the training samples are pre-mapped to a RKHS $\mathcal{H}$ via a feature mapping $\phi$, which is associated to a kernel function $k$, and a linear model is learned in the feature space $\mathcal{H}$:

$$\min_{\boldsymbol{f}} \; R(\boldsymbol{f}) + C \sum_{t=1}^{T} \sum_{i=1}^{N} L(f_t(\phi(\boldsymbol{x}_t^i)), y_t^i) \tag{3.27}$$

Such kernelization method is followed in [104] [2] [31] [49].

More importantly, the above mentioned method has been widely used in the research of MT-MKL. One simple example, which is the MT-MKL model based on the SVM, is introduced in (2.55). Besides, several previous MT-MKL works have been proposed following this direction.

In [1], similar to the non-kernelized methods that are introduced before, a Group-lasso type regularzer is utilized in a MT-MKL context:

$$R(\boldsymbol{f}) \triangleq (\sum_{t}^{T}(\sum_{m=1}^{M} \|\boldsymbol{w}_t^m\|_2)^{2q})^{1/q} \tag{3.28}$$

However, the difference between this work and the other ones is that, in this case, the weights $\boldsymbol{w}_t^m$'s are members of the RKHS $\mathcal{H}_m$, which is associated with a kernel function $k_m$. A similar but more sophisticated work is given in [43]. Also, as introduced in (2.60), the work in [79] utilized an $L_1 - L_q$ regularizer with $q \geq 1$, which brings sparsity in the kernel level, such that only a subset of the $M$ kernels are active, which are multiplied by a non-zero coefficient.

While these models are built in the primal domain, on the other hand, some other MT-MKL works directly formulate the model in the dual domain. Similar to the $L_2$-norm MKL method [51], in [82], the authors proposed a MT-MKL model, where all tasks share a common kernel function, which is learned in a $L_2$-norm MKL manner. However, in some cases, forcing all tasks to share the same kernel function may not be appropriate, especially when outlier tasks exist. Therefore, in [91], the authors proposed a model, which allows partial sharing of kernel function, by letting $k_t \triangleq \sum_{m=1}^{M}(\xi_m + \gamma_t^m)k_t^m$. Besides, a MT-MKL model that allows both feature and kernel selection is proposed in [44] and then extended in [45], based on the maximum entropy principle.

Finally, most previous MTL works minimize an average of the $T$ task objectives. However the authors in [57] argued that such an averaged objective function only leads to a specific point on the Pareto-Front of the corresponding Multi-Objective Optimization (MOO) problem. Clearly, such a

Pareto point does not necessarily lead to the best generalization performance for each task. Therefore, an MT-MKL model that minimizes a $L_p$-norm of the $T$ task objectives, named "Pareto-Path MTL", is proposed. Such a model is able to traverse a path on the Pareto-Front of the corresponding MOO problem, and has been shown to be advantages in most scenarios, compared to the averaged objective. This paper was further extended to a "Conic MTL" in [56], where the conic combination of the task objectives are optimized. The generalization bound for the Conic MTL model was derived. It was proved that, in most cases, the tightest generalization bound was not achieved by the averaged MTL model, thus justified the benefits of considering Conic MTL against averaged MTL.

### Algorithms

One serious difficulty that prevents people from using MTL methods, especially kernel-based MTL and MT-MKL is that, it is not usually easy to develop practical algorithm to solve the corresponding optimization problems. In previous research, while some papers ignore the algorithm implementation of their proposed methods, several other papers tried to develop optimization algorithms specifically to their models. This section reviews these algorithms without describing their details.

The authors in [104] proposed a kernel-based MTL model in the primal domain, where an Accelerated Proximal Gradient method is utlized to solve their problem. The authors of [91] formulated a framework, where individual tasks utilize their own task-specific space in conjunction with a shared space component, while the balance between these two components is controlled via weights. An exchange method algorithm is proposed to solve the resulting min-max problem. In [1], a Group-Lasso regularizer on SVM weights is considered that yields coefficient sparsity within a group of tasks and non-sparse regularization across groups. The resulting problem is solved in the dual domain via a mirror descent-based algorithm. A variation of this algorithm is used in solving the

formulation that is proposed in [43]. In [79], the Group-Lasso regularizer is generalized to $L_p - L_q$ regularization and two problems are addressed, one being convex and the other one non-convex. For each of the two formulations, a specialized algorithm is proposed. Finally, maximum entropy discrimination is employed in [44] and extended in [45] to construct a MT-MKL framework, while a sequential quadratic programming algorithm is proposed specifically to solve their model.

For example, the early work in [55] suggests a MKL formulation with trace constraints over the linearly combined kernels, which is further transformed into a solvable Semi-Definite Programming (SDP) problem. Simple-MKL [78] proposes a MKL formulation with $L_1$-norm constrained coefficients. In each iteration of the algorithm, it solves an SVM problem by taking advantage of an existing efficient SVM solver and the coefficients are updated based on a gradient-based scheme. Similar algorithm is also applied in [97]. In [51], an $L_2$-norm constraint is applied to the linear combination coefficients and the proposed min-max formulation is transformed into a Semi-Infinite Programming (SIP) problem, which is then solved via a cutting plane algorithm. Moreover, two algorithms are proposed in [50] to solve the $L_p$-MKL formulation, where the coefficient constraint is generalized to an $L_p$-norm constraint. The relationship between the latter $L_p$-MKL formulation and one that entails a Group-Lasso regularizer is pointed out in [98], which utilizes a block coordinate descent algorithm to solve the problem in the primal domain. Besides MKL-based SVM models, Cortes *et al.* [24] proposed a MKL formulation for KRR[83]. Cast as a min-max problem, it is solved via an interpolated iterative algorithm, which takes advantage of the closed-form solution of KRR and the kernel coefficients.

The algorithmic difficulty of both single-task and multi-task MKL problems not only brought a large amount of algorithms, most of which are tailored for specific problems, but also lead to model selection issues. For people who want to compare different models, a large amount of sophisticated algorithms may need to be implemented. Also, for researchers who proposed new models, it is usually an additional burden to develop algorithms for solving the corresponding models. Ob-

serving these issues, in Chapter 5, we solve these problems by proposing a general framework, which covers most single-task and multi-task MKL problems. More importantly, we propose an easy-to-implement algorithm to solve this general framework, thus solves all formulations that are covered by the proposed framework.

# CHAPTER 4: NEW HYPOTHESIS SPACES, GENERALIZATION BOUNDS AND MODELS

The research on generalization bound for MTL HSs (thus models) provides theoretical justification of learning multiple tasks simultaneously with shared information, as introduced in Chapter 2. Also, by studying the generalization bounds for specific HS, useful insights of the performance of the corresponding models can be drawn from the bound. In a better scenario, studying generalization bound of HSs may inspire the invention of new HSs, whose generalization bounds are tighter, and the performance of corresponding models are potentially better.

In order to conduct inductive transfer by training multiple tasks simultaneously, as reviewed in Chapter 3, there are two widely used techniques for information sharing amongst tasks: One is to apply specifically designed regularizers, which usually capture the prior assumptions of the underlying task relationships; another one is to pre-map data from all tasks to a (partially) common feature space by a feature mapping, which is pre-selected or learned during the training stage, followed by the learning of all tasks in the (partially) common feature space.

Although the research of generalization bound for the first information sharing strategy has received a fruitful of accomplishments, the second one has largely been omitted. As discussed in Chapter 3, the two papers follow this line of research only considered very limited cases: First, none of the two works considered the case of a non-linear feature mapping, which may be learned during training, an important scenario for kernel-based MTL. Second, the two papers constrained the weight $\boldsymbol{w}_t$ of each task in a norm-ball with the same radius, *i.e.*$\|\boldsymbol{w}_t\|^2 \leq R, \forall t = 1, \cdots, T$. This setting unnecessarily bereaves the flexibility of the HS, and, as we will show in this chapter, lead to a much looser generalization bound, which potentially deteriorate the performance of MTL models that based on this HS.

Inspired by the above observation, in this chapter, we propose two new HSs, one for single kernel MTL and another one for MT-MKL, which are generalization of previous HS (3.10). More importantly, we derive the generalization bounds for the two HSs, followed by an analysis of each bound. We show that, indeed, by introducing the flexibility of the norm-ball constraint, tighter generalization bound can be achieved, which indicates that the corresponding model may enjoy better generalization performance. Such an observation directly inspired us to propose a MTL model, which is also introduced in this chapter, based on the proposed HSs.

## Hypothesis Spaces

Given $T$ tasks, where each task features a linear function in the RKHS $\mathcal{H}$, *i.e.*, $f_t(\boldsymbol{x}) \triangleq \langle \boldsymbol{w}_t, \phi(\boldsymbol{x}) \rangle_{\mathcal{H}}$, we propose the following HS for kernel-based MTL

$$\mathcal{F}_s \triangleq \{\boldsymbol{x} \mapsto [\langle \boldsymbol{w}_1, \phi(\boldsymbol{x}) \rangle_{\mathcal{H}}, \cdots, \langle \boldsymbol{w}_T, \phi(\boldsymbol{x}) \rangle_{\mathcal{H}}]' : \|\boldsymbol{w}_t\|_{\mathcal{H}}^2 \leq (\lambda_t)^2 R, \boldsymbol{\lambda} \in \Omega_s(\boldsymbol{\lambda})\} \tag{4.1}$$

where the radius parameters $\lambda_t$'s are selected in the feasible region $\Omega_s(\boldsymbol{\lambda}) \triangleq \{\boldsymbol{\lambda} \succeq \mathbf{0}, \|\boldsymbol{\lambda}\|_s \leq 1, s \geq 1\}$, $\phi : \mathcal{X} \to \mathcal{H}$ is the feature mapping associated to a kernel function $k : \mathcal{X} \times \mathcal{X} \to \mathbb{R}$, and $\mathcal{H}$ is the corresponding RKHS. Also, we propose the following HS for MT-MKL

$$\begin{aligned} \mathcal{F}_{s,r} \triangleq \{\boldsymbol{x} \mapsto [\langle \boldsymbol{w}_1, \phi(\boldsymbol{x}) \rangle_{\mathcal{H}_{\boldsymbol{\theta}}}, \cdots, \langle \boldsymbol{w}_T, \phi(\boldsymbol{x}) \rangle_{\mathcal{H}_{\boldsymbol{\theta}}}]' : \\ \|\boldsymbol{w}_t\|_{\mathcal{H}_{\boldsymbol{\theta}}}^2 \leq (\lambda_t)^2 R, \boldsymbol{\lambda} \in \Omega_s(\boldsymbol{\lambda}), \phi \in \Omega_r(\phi)\} \end{aligned} \tag{4.2}$$

where, instead of letting $\phi$ begin fixed before training, it is selected from the feasible region $\Omega_r(\phi) = \{\phi : \phi = (\sqrt{\theta_1}\phi_1, \cdots, \sqrt{\theta_M}\phi_M), \boldsymbol{\theta} \succeq \mathbf{0}, \|\boldsymbol{\theta}\|_r \leq 1, r \geq 1\}$, $\phi_m : \mathcal{X} \to \mathcal{H}_m$, $\phi(\boldsymbol{x}) \in \mathcal{H}_{\boldsymbol{\theta}}$, $\mathcal{H}_{\boldsymbol{\theta}} = \mathcal{H}_1 \times \cdots \times \mathcal{H}_M$ is the RKHS that corresponds to the kernel function $k \triangleq \sum_{m=1}^{M} \theta_m k_m$,

$\phi_m : \mathcal{X} \to \mathcal{H}_m$ is the $m$-th feature mapping is associated to the manually selected kernel function $k_m : \mathcal{X} \times \mathcal{X} \to \mathbb{R}$.

It is not difficult to see that, in the first HS, $\mathcal{F}_s$ possesses a fixed, pre-selected feature mapping $\phi$, while in the second HS, the feature mapping $\phi$ is learned via the MKL approach, as introduced in Equation (2.30). Besides, in both of the two HSs, each task weight $\boldsymbol{w}_t$ is constrained in the norm-ball, whose size is parametrized by $\lambda_t$, instead of forcing the same radius for all norm-ball constraints, which is the case in [4] and [65]. Obviously, by setting $\lambda_t = 1, \forall t = 1, \cdots, T$, $\mathcal{F}_s$ degrades to the equal-radius HS, which is the case of [65].

In the following, we derive the ERC-based generalization bound for $\mathcal{F}_s$, which is introduced in Theorem 7. Then, we demonstrate that the ERC is monotonically increasing with $s$, which implies that the lowest ERC value is achieved, when $s = 1$. We then provide an upper bound for the ERC of $\mathcal{F}_s$, which also monotonically increases with respect to $s$. Besides, we illustrate that, the equal-radius HS, which is introduced in [65], is a special case of our proposed HS $\mathcal{F}_s$, when $s \to +\infty$. Therefore, due to the monotonicity of the generalization bound of $\mathcal{F}_s$, we conclude that the HS with equal radius for all norm-ball constraints is the worst, in terms of the tightness of generalization bound, which indicates potential worse performance for models that are based on this HS. In fact, in the optimal case ($s = 1$), we achieve a generalization bound of order $O(\frac{\sqrt{\log T}}{T})$, which decreases quickly with respect to $T$. On the other hand, when $s \to +\infty$, the bound does not decrease with $T$, thus, does not take advantage of the learning with multiple tasks and is less preferred.

We then derive the generalization bound for the HS $\mathcal{F}_{s,r}$, which still features a bound of order $O(\frac{\sqrt{\log T}}{T})$ when $s = 1$, as in the single-kernel setting. Additionally, if $M$ kernel functions are involved, the bound is of order $O(\sqrt{\log M})$, which has been proven to be the best bound that can be obtained in single-task multi-kernel classification [26]. Therefore, this optimal the bound is also preserved in the MT-MKL case.

Fixed Feature Mapping

Let $\{\boldsymbol{x}_t^i, y_t^i\} \in \mathcal{X} \times \{-1, 1\}, i = 1, \cdots, N$ be the training sample drawn identically and independently from the probability distribution $P_t(X, Y), t = 1, \cdots, T$. Let $\mathcal{H}$ be a RKHS associated with the kernel function $k(\cdot, \cdot) : \mathcal{X} \times \mathcal{X} \mapsto \mathbb{R}$, whose corresponding feature mapping is denoted by $\phi : \mathcal{X} \mapsto \mathcal{H}$. In what follows we give the theoretical analysis of our HS $\mathcal{F}_s$.

*Theoretical Results*

Before digging into our theoretical results, we first give the following definition for duality mapping, which will be used later:

$$(\cdot)^* : a \mapsto a^* \triangleq \begin{cases} \frac{a}{a-1}, & \forall a \neq 1 \\ +\infty, & a = 1 \end{cases} \tag{4.3}$$

Based on the definition of ERC for MTL, it is straightforward to see that

$$\hat{R}(\mathcal{F}_s) = E_\sigma\{\sup_{\boldsymbol{f} \in \mathcal{F}_s} \frac{2}{TN} \sum_{t=1}^{T} \sum_{i=1}^{N} \sigma_t^i f_t(\boldsymbol{x}_t^i)\} \tag{4.4}$$

According to Theorem 7, in order to derive the generalization bound for $\mathcal{F}_s$, it is needed to calculate, or, at least upper bound, the above ERC term. The next lemma provides an equivalent form of Equation (4.4), which not only facilitates the proof of the upper bound of $\hat{R}(\mathcal{F}_s)$, but also helps the proof of other important properties of $\mathcal{F}_s$ that are shown in Theorem 8 and Theorem 9.

**Lemma 2.** *Let $\boldsymbol{\sigma}_t \triangleq [\sigma_t^1, \cdots, \sigma_t^N]'$, $u_t \triangleq \sqrt{\boldsymbol{\sigma}_t' \boldsymbol{K}_t \boldsymbol{\sigma}_t}$, where $\boldsymbol{K}_t$ is the kernel matrix that consists*

*of elements* $k(\boldsymbol{x}_t^i, \boldsymbol{x}_t^j)$, $\boldsymbol{u} \triangleq [u_1, \cdots, u_T]'$. *Then* $\forall s \geq 1$

$$\hat{R}(\mathcal{F}_s) = \frac{2}{TN}\sqrt{R}E_\sigma\{\|\boldsymbol{u}\|_{s^*}\} \tag{4.5}$$

Given Equation (4.5), the following theorem illustrates that $\hat{R}(\mathcal{F}_s)$ is monotonically increasing with respect to $s$.

**Theorem 8.** $\hat{R}(\mathcal{F}_s)$ *is monotonically increasing with respect to* $s$.

The above theorem clearly unveils the fact that, in order to get a potentially tight generalization bound, a small value of $s$ is desired, and the tightest bound is likely to be achieved when $s = 1$. As a tight generalization bound potentially leads to better generalization performance, the above theorem serves as a guide for the selection of the parameter $s$.

Define $\tilde{\mathcal{F}} \triangleq \{\boldsymbol{x} \mapsto [\langle \boldsymbol{w}_1, \phi(\boldsymbol{x})\rangle_{\mathcal{H}}, \cdots, \langle \boldsymbol{w}_T, \phi(\boldsymbol{x})\rangle_{\mathcal{H}}]' : \|\boldsymbol{w}_t\|_{\mathcal{H}}^2 \leq R\}$, which is the HS that is given in [65] under kernelized MTL setting, then $\tilde{\mathcal{F}}$ is the HS with equal radius for each $\|\boldsymbol{w}_t\|^2$. This is the special case of $\mathcal{F}_s$ with all $\lambda_t$'s equal to 1.

**Theorem 9.** $\hat{R}(\tilde{\mathcal{F}}) = \hat{R}(\mathcal{F}_{+\infty})$.

The above result, in conjunction with Theorem 8, implies that the HS $\tilde{\mathcal{F}}$, which sets equal radii for all $\boldsymbol{w}_t$'s, is the least preferred, since it has the largest ERC value. This fact can be also observed from another angle: Since $0 \leq \lambda_t \leq 1, \forall t = 1, \cdots, T$, the volume of $\mathcal{F}_s$ is smaller than that of $\tilde{\mathcal{F}}$. Therefore, it is destined that the ERC of $\tilde{\mathcal{F}}$ is always larger than that of $\mathcal{F}_s$. In other words, our proposed HS $\mathcal{F}_s$ receives a lower value of ERC by shrinking its volume. As described in Chapter 2, decreasing the capacity of the HS may lead to increased empirical error $\hat{er}(\boldsymbol{f})$, which can potentially deteriorates the generalization performance. However, we will show via experiments that this is rarely encountered.

Next, as described above, in order to derive the generalization bound for $\mathcal{F}_s$, we need to compute, or, at least find an upper bound for $\hat{R}(\mathcal{F}_s)$. The following theorem addresses this requisite.

**Theorem 10.** *Let $\mathcal{F}_s$ be as defined in Equation (4.1), and let $\rho \triangleq 2\ln T$. Assume that $\forall \boldsymbol{x} \in \mathcal{X}$, $k(\boldsymbol{x}, \boldsymbol{x}) = \langle \phi(\boldsymbol{x}), \phi(\boldsymbol{x}) \rangle \leq 1$. Then the ERC is bounded as follows:*

$$\hat{R}(\mathcal{F}_s) \leq \frac{2}{T\sqrt{N}} \sqrt{\tau R T^{\frac{2}{s^*}}} \tag{4.6}$$

*where $\tau \triangleq (\max\{s, \rho^*\})^*$.*

*Analysis*

Based on Theorem 10, in this sub-section, we point out several useful insights that are captured by the ERC bound.

First of all, as Theorem 8 indicates that the ERC value of $\mathcal{F}_s$ is monotonically increasing with $s$, for a potentially tight ERC bound, it is expected that the bound is also monotonically increasing with $s$. It is not difficult to see that this property is preserved in our ERC bound.

As $s \to +\infty$, $\mathcal{F}_s$ degrades to $\tilde{\mathcal{F}}$. In this case, our ERC bound becomes $\hat{R}(\mathcal{F}_{+\infty}) \leq 2\sqrt{\frac{R}{N}}$. One immediate observation is that the bound is of order $O(\sqrt{\frac{1}{N}})$. If it was not a surprise that $\tilde{\mathcal{F}}$ maintains the largest ERC value because of its largest volume, it is a useful insight that its ERC bound does not decrease with respect to $T$. This fact implies that its real ERC value may decrease with $T$ very slowly, or, if the bound is tight enough, the real ERC value may not decrease with $T$ either. In other words, such MTL HS is not able to take advantage of multiple tasks; it is similar to learning each task independently.

Besides, it is worth to mention that the ERC bound $2\sqrt{\frac{R}{N}}$ matches the bound that is given in [65]. This is because of the following relation between $\tilde{\mathcal{F}}$ and the HS of [65], $\mathcal{F}$, that is introduced in Equation (3.10): First, let the operator $A$ in $\mathcal{F}$ be the identity operator, and then let $\boldsymbol{x}$ in $\mathcal{F}$ be an element of $\mathcal{H}$, *i.e.*let $\boldsymbol{x}$ in $\mathcal{F}$ be $\phi(\boldsymbol{x})$ in $\tilde{\mathcal{F}}$. Then $\mathcal{F}$ becomes $\tilde{\mathcal{F}}$. In other words, $\mathcal{F}$ is a special case of $\tilde{\mathcal{F}}$.

When $s$ decreases, a better generalization bound can be achieved, compared to when $s \to +\infty$. For finite $s$, the bound for $\mathcal{F}_s$ is of order $O(\frac{1}{T^{\frac{1}{s}}}\sqrt{\frac{1}{N}})$, which is obviously more preferred over the aforementioned $O(\frac{1}{\sqrt{N}})$ bound, as it asymptotically decreases with increasing number of tasks. Furthermore, when $s = \rho^*$, $\hat{R}(\mathcal{F}_{\rho^*}) \leq \frac{2}{T\sqrt{N}}\sqrt{2eR\log T}$. Here we achieve a bound of order $O(\frac{\sqrt{\log T}}{T})$, which decreases faster with increasing $T$ compared to the bound, when $s > \rho^*$. Finally, when $s = 1$, $\hat{R}(\mathcal{F}_1) \leq \frac{2}{T\sqrt{N}}\sqrt{2R\log T}$. While also being of order $O(\frac{\sqrt{\log T}}{T})$, in this case, it features a smaller constant compared to the bound of $\hat{R}(\mathcal{F}_{\rho^*})$. In fact, due to the monotonicity of the bound that is given in (4.6), the tightest bound is obtained when $s = 1$.

Based on the above analysis, we conclude that, theoretically, when $s = 1$, we can expect to achieve the best generalization performance for the corresponding MTL model, while the performance potentially deteriorates as $s$ increases its value. Therefore, the generalization bound not only reveals underlying insights for the HS (thus the MTL models that adapt $\mathcal{F}_s$ as their HS), but also proposed a suggestion for model selection: The smaller $s$ it is, the better model performance may be achieved.

### Learning the Feature Mapping

In this section, we consider the MTL HS $\mathcal{F}_{s,r}$, where the feature mapping $\phi$ is selected during the training stage via an MKL approach. In particular, we will assume that $\phi = (\sqrt{\theta_1}\phi_1, \cdots, \sqrt{\theta_M}\phi_M)$

$\in \mathcal{H}_{\boldsymbol{\theta}} = \mathcal{H}_1 \times \cdots \times \mathcal{H}_M$, where each $\phi_m : \mathcal{X} \mapsto \mathcal{H}_m$ is selected before training, and $\phi \in \Omega_r(\phi)$, $\Omega_r(\phi) = \{\phi : \phi = (\sqrt{\theta_1}\phi_1, \cdots, \sqrt{\theta_M}\phi_M), \boldsymbol{\theta} \succeq \mathbf{0}, \|\boldsymbol{\theta}\|_r \leq 1, r \geq 1\}$. In the next sub-section, we give theoretical analysis of the HS $\mathcal{F}_{s,r}$.

*Theoretical Results*

Based on the definition of ERC for MTL, it is straightforward to see that

$$\hat{R}(\mathcal{F}_{s,r}) = E_\sigma\{\sup_{\boldsymbol{f} \in \mathcal{F}_{s,r}} \frac{2}{TN} \sum_{t=1}^{T} \sum_{i=1}^{N} \sigma_t^i f_t(\boldsymbol{x}_t^i)\} \tag{4.7}$$

Similar to the previous section, it is needed to upper bound the above ERC term. The next lemma provides two equivalent forms of Equation (4.7), which not only facilitates the proof of the upper bound of $\hat{R}(\mathcal{F}_{s,r})$, but also help the proof other important properties of $\mathcal{F}_{s,r}$ that are shown in Theorem 11 and Theorem 12.

**Lemma 3.** *Let* $\boldsymbol{\sigma}_t \triangleq [\sigma_t^1, \cdots, \sigma_t^N]'$, $u_t^m \triangleq \boldsymbol{\sigma}_t' \boldsymbol{K}_t^m \boldsymbol{\sigma}_t$, $\boldsymbol{u}_t \triangleq [u_t^1, \cdots, u_t^M]'$, $v_m \triangleq \sum_{t=1}^{T} \lambda_t \boldsymbol{\sigma}_t' \boldsymbol{K}_t^m \boldsymbol{\alpha}_t$, $\boldsymbol{v} \triangleq [v_1, \cdots, v_M]'$, *where* $\boldsymbol{K}_t^m$ *is the kernel matrix that contains elements* $k_m(\boldsymbol{x}_t^i, \boldsymbol{x}_t^j)$. *Then* $\forall s \geq 1$ *and* $r \geq 1$,

$$\hat{R}(\mathcal{F}_{s,r}) = \frac{2}{TN}\sqrt{R} E_\sigma\{\sup_{\boldsymbol{\theta} \in \Omega_r(\boldsymbol{\theta})} \sum_{t=1}^{T} (\boldsymbol{\theta}' \boldsymbol{u}_t)^{\frac{s^*}{2}}\}^{\frac{1}{s^*}} = \frac{2}{TN} E_\sigma\{\sup_{\boldsymbol{\lambda} \in \Omega_s(\boldsymbol{\lambda}), \boldsymbol{\alpha} \in \Omega(\boldsymbol{\alpha})} \|\boldsymbol{v}\|_{r^*}\} \tag{4.8}$$

*where* $\Omega_r(\boldsymbol{\theta}) \triangleq \{\boldsymbol{\theta} : \boldsymbol{\theta} \succeq \mathbf{0}, \|\boldsymbol{\theta}\|_r \leq 1\}$, *and* $\Omega(\boldsymbol{\alpha}) \triangleq \{\boldsymbol{\alpha}_t : \boldsymbol{\sigma}_t{}' \boldsymbol{K}_t^m \boldsymbol{\alpha}_t \leq R, \forall t\}$.

Leveraging from Equation (4.8), the following theorem illustrates that $\hat{R}(\mathcal{F}_{s,r})$ is monotonically increasing with respect to $s$.

**Theorem 11.** $\hat{R}(\mathcal{F}_{s,r})$ *is monotonically increasing with respect to* $s$.

Again, the preference to small $s$ value is clearly indicated by the above theorem, and when $s = 1$, the smallest ERC value can be achieved. It potentially leads to the tightest generalization bound, and the best generalization performance.

Define $\tilde{\mathcal{F}}_r \triangleq \{\boldsymbol{x} \mapsto (\langle \boldsymbol{w}_1, \phi(\boldsymbol{x})\rangle_{\mathcal{H}_{\boldsymbol{\theta}}}, \cdots, \langle \boldsymbol{w}_T, \phi(\boldsymbol{x})\rangle_{\mathcal{H}_{\boldsymbol{\theta}}})' : \|\boldsymbol{w}_t\|^2_{\mathcal{H}_{\boldsymbol{\theta}}} \leq R, \phi \in \Omega_r(\phi)\}$, then the HS $\tilde{\mathcal{F}}$ is extended to a MT-MKL setting, in order to compare the proposed HS and the one features norm-ball constraints with equal radius.

**Theorem 12.** $\hat{R}(\tilde{\mathcal{F}}_r) = \hat{R}(\mathcal{F}_{+\infty,r})$.

This, again, indicates that the tightest generalization bound is obtained, when $s = 1$, as is the case for single kernel MKL. Also, the HS that constraints the $\boldsymbol{w}_t$'s within equal radius norm balls is therefore the least preferred. Similar to the previous discussion, in order to estimate the expected classification error, one has to calculate the quantity of the ERC, or at least upper bound it. Due to the difficulty of calculating the precise value, in the following, we provide an upper bound on $\hat{R}(\mathcal{F}_{s,r})$.

**Theorem 13.** *Let $\mathcal{F}_{s,r}$ be as defined in Equation (4.2). Assume that $\forall \boldsymbol{x} \in \mathcal{X}$, $m = 1, \cdots, M$, $k_m(\boldsymbol{x}, \boldsymbol{x}) = \langle \phi_m(\boldsymbol{x}), \phi_m(\boldsymbol{x})\rangle_{\mathcal{H}_m} \leq 1$. The ERC can be bounded as follows:*

$$\hat{R}(\mathcal{F}_{s,r}) \leq \frac{2}{T\sqrt{N}} \sqrt{R s^* T^{\frac{2}{s^*}} M^{\max\{\frac{1}{r^*}, \frac{2}{s^*}\}}} \tag{4.9}$$

The above theorem is valid for any $r \geq 1$ and $s \geq 1$. However, due to its generality, it is not tight enough in some cases. In the following corollary, we provide refinement of the above bound under two specific situations:

**Corollary 1.** *Under the conditions that are given in Theorem 13, we have*

$$\begin{aligned} \hat{R}(\mathcal{F}_{s,r}) &\leq \frac{2}{T\sqrt{N}}\sqrt{\tau R T^{\frac{2}{s^*}} M^{\frac{1}{r^*}}} \qquad \textit{if } r^* \leq \ln T \\ \hat{R}(\mathcal{F}_{s,r}) &\leq \frac{2}{T\sqrt{N}}\sqrt{\tau R T^{\frac{2}{s^*}} M^{\frac{2}{\tau}}} \qquad \textit{if } r^* \geq \ln MT \end{aligned} \tag{4.10}$$

$\forall s \geq 1$, *where* $\tau \triangleq (\max\{s, \rho^*\})^*$, *and*

$$\rho \triangleq \begin{cases} 2\ln T, & r^* \leq \ln T \\ 2\ln MT, & r^* \geq \ln MT \end{cases} \tag{4.11}$$

*Analysis*

Once again, to unveil the insights of the MT-MKL HS, it is needed to analyze the ERC bounds that are given in Theorem 13 and Corollary 1.

First of all, we analyze the bound that is given in (4.9), which is a general result $\forall s \geq 1, r \geq 1$. Clearly, for fixed $r$, (4.9) gives a bound of order $O(\frac{1}{T^{\frac{1}{s}}})$. Therefore, it is not difficult to see that $s \to +\infty$ is the least preferred, since it is the same as independent learning and does not take advantage of the appearance of multiple tasks. This conclusion is the same as the single kernel scenario, as discussed in the previous section. Moreover, based on (4.9), $\forall r \geq 1$, $\hat{R}(\mathcal{F}_{1,r})$'s bound is of order $O(\sqrt{M^{\frac{1}{r^*}}})$. Compared to the $O(\sqrt{M^{\frac{1}{r^*}}\min(\lceil \ln M \rceil, \lceil r^* \rceil)})$ bound, which is given in [52], our bound for MT-MKL is tighter.

When $r^* \geq \ln MT$, the monotonicity with respect to $s$ is preserved in the ERC bound that is given in (4.10). When $s = \rho^*$, $\hat{R}(\mathcal{F}_{\rho^*,r}) \leq \frac{2}{T\sqrt{N}}\sqrt{2eR\ln MT}$. This gives a $O(\frac{\sqrt{\ln MT}}{T})$ bound. Since it has been proved that the best obtainable bound in a single-task multiple kernel classification

setting is of order $O(\sqrt{\ln M})$ [26], this logarithmic bound is preserved in our proposed HS in the MT-MKL context. When $s = 1$, $\hat{R}(\mathcal{F}_{1,r}) \leq \frac{2}{T\sqrt{N}}\sqrt{2RM^{\frac{1}{\ln MT}} \ln MT}$. Since it is true that $M^{\frac{1}{\ln MT}} \leq e, \forall M \geq 1, T \geq 1$, this bound is even tighter than the one obtained, when $s = \rho^*$.

When $r^* \leq \ln T$, similar to the previous case, the monotonicity is preserved. Differently, in this bound, when $s = \rho^*$, $\hat{R}(\mathcal{F}_{\rho^*,r}) \leq \frac{2}{T\sqrt{N}}\sqrt{2eRM^{\frac{1}{r^*}} \ln T}$. This gives a $O(\frac{\sqrt{M^{\frac{1}{r^*}} \ln T}}{T})$ bound. When $s = 1$, $\hat{R}(\mathcal{F}_{1,r}) \leq \frac{2}{T\sqrt{N}}\sqrt{2RM^{\frac{1}{r^*}} \ln T}$. Clearly, it further decreases the bound by a constant factor $e$.

To compare the above analyzed optimum bounds, when $r^* \geq \ln MT$, a better bound with respect to $M$ is achieved, *i.e.*, $O(\sqrt{\ln M})$ versus $O(\sqrt{M^{\frac{1}{r^*}}})$. On the other hand, with regards to $T$, even though a $O(\frac{\sqrt{\ln T}}{T})$ bound is reached in both cases, when $r^* \leq \ln T$, a lower constant is obtained. Therefore, there is a trade-off exist on the tightness of bound, induced by the number of tasks and number of kernels. In practice, we usually have a much larger number of tasks $T$, compared to the number of kernel functions $M$. Therefore, it is more important to set the parameter value $r$ such that the generalization bound can be benefited the most from the large number of tasks, where the increasing of the bound value caused by $M$ can therefore be compensated.

To finally point out the benefit of the proposed HS for MT-MKL, we note that the corresponding generalization bound not only preserves the optimal $O(\sqrt{\ln M})$ bound encountered in single-task MKL, but also preserves the optimal $O(\frac{\sqrt{\ln T}}{T})$ bound encountered in the single-kernel MTL case, which was given in the previous section. This fact indicates a potentially good design of $\mathcal{F}_{s,r}$.

## New Models for Kernel-based MTL and MT-MKL

The previous sections demonstrated the benefits of our proposed HSs, in terms of tighter generalization bound. It is of interest to verify that the advantageous HS is indeed beneficial in prac-

tical application, by testing the generalization performance of the corresponding MTL model. In this section, a new kernel-based MTL model is proposed, by adapting $\mathcal{F}_s$ as its HS. Besides, a MT-MKL model is also developed, with $\mathcal{F}_{s,r}$ being assigned as its HS.

Clearly, due to the structure of $\mathcal{F}_s$, the corresponding MTL model can be easily formulated as follows:

$$\begin{aligned} \min_{\boldsymbol{w},\boldsymbol{\lambda}} \ & \sum_{t=1}^{T}\sum_{i=1}^{N} L(\langle \boldsymbol{w}_t, \phi(\boldsymbol{x}_t^i)\rangle_{\mathcal{H}}, y_t^i) \\ \text{s.t. } & \|\boldsymbol{w}_t\|_{\mathcal{H}}^2 \leq \lambda_t^2 R, t = 1, \cdots, T \\ & \boldsymbol{\lambda} \in \Omega_s(\boldsymbol{\lambda}) \end{aligned} \tag{4.12}$$

and the corresponding MT-MKL model can be formulated similarly.

However, in practice, it is not quite straightforward to figure out how to solve the above optimization problem, and it will not be a surprise that the corresponding MT-MKL model is even more difficult to optimize.

In order to tackle this problem,in the next theorem, we show the relation between our HSs and the ones that are given by the Group-Lasso type regularizer. We conclude that an equivalence exists between our HSs and the ones that utilize Group-lasso type regularizer, which is a key property in facilitating the proposal of easy-to-solve MTL models, based on our HSs.

**Theorem 14.** *The HS $\mathcal{F}_s$ is equivalent to*

$$\mathcal{F}_s^{GL} \triangleq \{\boldsymbol{x} \mapsto [\langle \boldsymbol{w}_1, \phi(\boldsymbol{x})\rangle_{\mathcal{H}}, \cdots, \langle \boldsymbol{w}_T, \phi(\boldsymbol{x})\rangle_{\mathcal{H}}]' : (\sum_{t=1}^{T} \|\boldsymbol{w}_t\|_{\mathcal{H}}^s)^{\frac{2}{s}} \leq R\} \tag{4.13}$$

*Similarly, $\mathcal{F}_{s,r}$ is equivalent to*

$$\mathcal{F}_{s,r}^{GL} \triangleq \{\boldsymbol{x} \mapsto [\langle \boldsymbol{w}_1, \phi(\boldsymbol{x})\rangle_{\mathcal{H}_{\boldsymbol{\theta}}}, \cdots, \langle \boldsymbol{w}_T, \phi(\boldsymbol{x})\rangle_{\mathcal{H}_{\boldsymbol{\theta}}}]' : (\sum_{t=1}^{T} \|\boldsymbol{w}_t\|_{\mathcal{H}_{\boldsymbol{\theta}}}^{s})^{\frac{2}{s}} \leq R, \boldsymbol{\theta} \in \Omega_r(\boldsymbol{\theta})\} \quad (4.14)$$

Based on the above theorem and Proposition 1, we immediately reach the following kernel-based MTL formulation, whose HS is given as $\mathcal{F}_s$:

$$\min_{\boldsymbol{w}} (\sum_{t=1}^{T} (\frac{\|\boldsymbol{w}_t\|_{\mathcal{H}}^2}{2})^{\frac{s}{2}})^{\frac{2}{s}} + C \sum_{t=1}^{T} \sum_{i=1}^{N} L(\langle \boldsymbol{w}_t, \phi(\boldsymbol{x}_t^i)\rangle_{\mathcal{H}}, y_t^i) \quad (4.15)$$

Similarly, the MT-MKL formulation can be written as follows:

$$\min_{\boldsymbol{w}, \boldsymbol{\theta} \in \Omega_r(\boldsymbol{\theta})} (\sum_{t=1}^{T} (\frac{\|\boldsymbol{w}_t\|_{\mathcal{H}_{\boldsymbol{\theta}}}^2}{2})^{\frac{s}{2}})^{\frac{2}{s}} + C \sum_{t=1}^{T} \sum_{i=1}^{N} L(\langle \boldsymbol{w}_t, \phi(\boldsymbol{x}_t^i)\rangle_{\mathcal{H}_{\boldsymbol{\theta}}}, y_t^i) \quad (4.16)$$

In the following sub-sections, we discuss in detail the models on classification problems, where the hinge loss is utilized, and thus leads to SVM-based formulations.

### *MTL Formulation*

Before digging into the proposed MTL formulation, we first introduce the following theorem, which is provided in [57].

**Theorem 15.** *Let $p \geq 1$, $\boldsymbol{\lambda}, \boldsymbol{g} \in \mathbb{R}^T$ such that $\boldsymbol{g} \succeq \mathbf{0}$, but $\boldsymbol{g} \neq \mathbf{0}$. Also, let $q \triangleq \frac{p}{p-1}$. Then,*

$$\max_{\boldsymbol{\lambda} \in \bar{B}_{\boldsymbol{\lambda},q}} \boldsymbol{\lambda}' \boldsymbol{g} = \nu(\boldsymbol{g})_p = \|\boldsymbol{g}\|_p \tag{4.17}$$

*Furthermore, a solution to the previously stated maximization problem is given as*

$$\boldsymbol{\lambda}^* = \begin{cases} (\frac{\boldsymbol{g}}{\|\boldsymbol{g}\|_p})^{p-1} & \text{if } p > 1 \\ \mathbf{1} & \text{if } p = 1 \end{cases} \tag{4.18}$$

Now, given a set of training samples $\{\boldsymbol{x}_t^i, y_t^i\} \in \mathcal{X} \times \{-1, 1\}, i = 1, \cdots, N, t = 1, \cdots, T$ and fixed feature mapping $\phi : \mathcal{X} \mapsto \mathcal{H}$, our model in a SVM-based context for classification problems is given as follows:

$$\min_{\boldsymbol{w},\boldsymbol{b}} (\sum_{t=1}^{T} (\frac{\|\boldsymbol{w}_t\|_{\mathcal{H}}^2}{2})^{\frac{s}{2}})^{\frac{2}{s}} + C \sum_{t=1}^{T} \sum_{i=1}^{N} L_{hinge}(y_t^i(\langle \boldsymbol{w}_t, \phi(\boldsymbol{x}_t^i) \rangle_{\mathcal{H}} + b_t)) \tag{4.19}$$

Obviously, based on the discussions above, we are aware that $\mathcal{F}_s$ is the HS of (4.19). This minimization problem is generally difficult to solve. However, with a simple transform, it can be easily optimized.

First, note that, based on Theorem 5, when $1 \leq s \leq 2$, Problem (4.19) is equivalent to

$$\begin{aligned} \min_{\boldsymbol{w},\boldsymbol{b},\boldsymbol{\lambda}} \; & \sum_{t=1}^{T} \frac{\|\boldsymbol{w}_t\|_{\mathcal{H}}^2}{2\lambda_t} + C \sum_{t=1}^{T} \sum_{i=1}^{N} L_{hinge}(y_t^i(\langle \boldsymbol{w}_t, \phi(\boldsymbol{x}_t^i) \rangle_{\mathcal{H}} + b_t)) \\ \text{s.t. } & \boldsymbol{\lambda} \succeq \mathbf{0}, \|\boldsymbol{\lambda}\|_{\frac{s}{2-s}} \leq 1 \end{aligned} \tag{4.20}$$

which can be easily solved via block coordinate descent method, with $\{\boldsymbol{w}, \boldsymbol{b}\}$ as a group and $\boldsymbol{\lambda}$ as another. Specifically, when $\boldsymbol{\lambda}$ is fixed, minimizing with respect to $\{\boldsymbol{w}, \boldsymbol{b}\}$ is simply solving $T$ independent SVM problems, where efficient solver can be applied, such as `LIBSVM` [18]. On the other hand, when optimizing with respect to $\boldsymbol{\lambda}$ with $\{\boldsymbol{w}, \boldsymbol{b}\}$ fixed, the closed-form solution is already provided in Theorem 5.

When $s > 2$, according to Theorem 15, (4.19) is equivalent to

$$\begin{aligned} \min_{\boldsymbol{w},\boldsymbol{b}} \max_{\boldsymbol{\lambda}} \ & \sum_{t=1}^{T} \frac{\lambda_t \|\boldsymbol{w}_t\|^2}{2} + C \sum_{t=1}^{T} \sum_{i=1}^{N} L_{hinge}(y_t^i(\langle \boldsymbol{w}_t, \phi(\boldsymbol{x}_t^i)\rangle_{\mathcal{H}} + b_t)) \\ & \text{s.t. } \boldsymbol{\lambda} \succeq \mathbf{0}, \|\boldsymbol{\lambda}\|_{\frac{s}{s-2}} \leq 1 \end{aligned} \tag{4.21}$$

The order of the min and max operation can be changed in the above min-max problem. This is because that the above problem is convex-concave, and the feasible region is compact [88]. Therefore, we have

$$\begin{aligned} \max_{\boldsymbol{\lambda}} \min_{\boldsymbol{w},\boldsymbol{b}} \ & \sum_{t=1}^{T} \frac{\lambda_t \|\boldsymbol{w}_t\|^2}{2} + C \sum_{t=1}^{T} \sum_{i=1}^{N} L_{hinge}(y_t^i(\langle \boldsymbol{w}_t, \phi(\boldsymbol{x}_t^i)\rangle_{\mathcal{H}} + b_t)) \\ & \text{s.t. } \boldsymbol{\lambda} \succeq \mathbf{0}, \|\boldsymbol{\lambda}\|_{\frac{s}{s-2}} \leq 1 \end{aligned} \tag{4.22}$$

The above problem can be further transfered to the following equivalent maximization problem, by calculating the dual form of the inner SVM problem:

$$\begin{aligned} \max_{\boldsymbol{\alpha},\boldsymbol{\lambda}} \ & \sum_{t=1}^{T} \lambda_t (\boldsymbol{\alpha}_t{}' \mathbf{1} - \frac{1}{2} \boldsymbol{\alpha}_t{}' \boldsymbol{Y}_t \boldsymbol{K}_t \boldsymbol{Y}_t \boldsymbol{\alpha}_t) \\ & \text{s.t. } \mathbf{0} \preceq \boldsymbol{\alpha}_t \preceq \frac{C}{\lambda_t} \mathbf{1}, \ \boldsymbol{\alpha}_t{}' \boldsymbol{y}_t = 0, \ \forall t = 1, \cdots, T \\ & \quad \boldsymbol{\lambda} \succeq \mathbf{0}, \|\boldsymbol{\lambda}\|_{\frac{s}{s-2}} \leq 1 \end{aligned} \tag{4.23}$$

where $\boldsymbol{Y}_t \triangleq \mathrm{diag}(y_t^1, \cdots, y_t^N)$ and $\boldsymbol{K}_t$ is the kernel matrix calculated based on the training data from the $t$-th task. Similar to the previous case, here, group coordinate descent can be utilized to solve (4.23), with $\boldsymbol{\lambda}$ as a group, which has closed-form solution based on Theorem 15, and $\boldsymbol{\alpha}$ as another group, leading to solving $T$ SVM problems.

*MT-MKL Formulation*

In order to propose a MT-MKL model in a SVM-based classification context, Problem (4.16) is customized to the following model by utilizing the hinge loss:

$$\begin{aligned} \min_{\boldsymbol{w},\boldsymbol{b},\boldsymbol{\theta}\in\Omega_r(\boldsymbol{\theta})} & (\sum_{t=1}^{T}(\frac{\|\boldsymbol{w}_t\|_{\mathcal{H}_{\boldsymbol{\theta}}}^2}{2})^{\frac{s}{2}})^{\frac{2}{s}} + C\sum_{t=1}^{T}\sum_{i=1}^{N} L_{hinge}(y_t^i(\langle \boldsymbol{w}_t, \phi(\boldsymbol{x}_t^i)\rangle_{\mathcal{H}_{\boldsymbol{\theta}}} + b_t)) \\ & \text{s.t. } \boldsymbol{\theta} \succeq \mathbf{0}, \|\boldsymbol{\theta}\|_r \leq 1 \end{aligned} \tag{4.24}$$

Similar to the previous argument, $\mathcal{F}_{s,r}$ serves as the HS of the above formulation. To solve the above problem, when $1 \leq s \leq 2$, a similar block coordinate descent can be utilized. To be concrete, in this case, the above problem is equivalent to

$$\begin{aligned} \min_{\boldsymbol{w},\boldsymbol{b},\boldsymbol{\lambda},\boldsymbol{\theta}} & \sum_{t=1}^{T}\sum_{m=1}^{M}\frac{\|\boldsymbol{w}_t^m\|_{\mathcal{H}_m}^2}{2\lambda_t\theta_m} + C\sum_{t=1}^{T}\sum_{i=1}^{N} L_{hinge}(y_t^i(\sum_{m=1}^{M}\langle \boldsymbol{w}_t^m, \phi_m(\boldsymbol{x}_t^i)\rangle_{\mathcal{H}_m} + b_t)) \\ & \text{s.t. } \boldsymbol{\theta} \succeq \mathbf{0}, \|\boldsymbol{\theta}\|_r \leq 1 \\ & \quad \boldsymbol{\lambda} \succeq \mathbf{0}, \|\boldsymbol{\lambda}\|_{\frac{s}{2-s}} \leq 1 \end{aligned} \tag{4.25}$$

When $\boldsymbol{\lambda}$ and $\boldsymbol{\theta}$ are fixed, optimizing with respect to $\{\boldsymbol{w}, \boldsymbol{b}\}$ involves solving $T$ SVM problems, whose kernel function is given by $k = \sum_{m=1}^{M}\theta_m k_m$. On the other hand, optimizing with respect to both $\boldsymbol{\lambda}$ and $\boldsymbol{\theta}$ enjoys closed-form solution, again, as given in Theorem 5.

In the case of $s > 2$, with the help of Theorem 15, Problem (4.24) is equivalent to the following min-max problem:

$$\begin{aligned}\min_{\boldsymbol{w},\boldsymbol{b},\boldsymbol{\theta}} \max_{\boldsymbol{\lambda}} \ & \sum_{t=1}^{T}\sum_{m=1}^{M} \frac{\lambda_t \|\boldsymbol{w}_t^m\|_{\mathcal{H}_m}^2}{2\theta_m} + C\sum_{t=1}^{T}\sum_{i=1}^{N} L_{hinge}(y_t^i(\sum_{m=1}^{M}\langle \boldsymbol{w}_t^m, \phi_m(\boldsymbol{x}_t^i)\rangle_{\mathcal{H}_m} + b_t)) \\ & \text{s.t. } \boldsymbol{\theta} \succeq \mathbf{0}, \|\boldsymbol{\theta}\|_r \leq 1 \\ & \boldsymbol{\lambda} \succeq \mathbf{0}, \|\boldsymbol{\lambda}\|_{\frac{s}{s-2}} \leq 1\end{aligned} \tag{4.26}$$

which, similar to the single kernel case, can be equivalently transformed to the dual domain, yielding

$$\begin{aligned}\min_{\boldsymbol{\theta}} \max_{\boldsymbol{\alpha},\boldsymbol{\lambda}} \ & \sum_{t=1}^{T}(\boldsymbol{\alpha}_t{}'\mathbf{1} - \frac{1}{2\lambda_t}\boldsymbol{\alpha}_t{}'\boldsymbol{Y}_t(\sum_{m=1}^{M}\theta_m \boldsymbol{K}_t^m)\boldsymbol{Y}_t\boldsymbol{\alpha}_t) \\ & \text{s.t. } \mathbf{0} \preceq \boldsymbol{\alpha}_t \preceq C\mathbf{1},\ \boldsymbol{\alpha}_t{}'\boldsymbol{y}_t = 0,\ \forall t \\ & \boldsymbol{\lambda} \succeq \mathbf{0}, \|\boldsymbol{\lambda}\|_{\frac{s}{s-2}} \leq 1 \\ & \boldsymbol{\theta} \succeq \mathbf{0}, \|\boldsymbol{\theta}\|_r \leq 1\end{aligned} \tag{4.27}$$

However, different to the single kernel case, which is formulated as a maximization problem and enjoys a simple group coordinate descent algorithm, the above min-max problem cannot be similarly solved by group gradient descent method. In fact, many MKL and MT-MKL models have been formulated similarly as min-max problems, which are not easy to solve. As discussed in Chapter 2 and Chapter 3, such a difficulty severely hindered the research and application of MT-MKL, since in most cases new algorithms have to be invented for new models, and practitioners need to interpret and implement several sophisticated algorithms in order to compare various models. This fact inspired us to propose a general framework, which encompasses many MKL and

MT-MKL dual formulations. Then, we will propose an algorithm for solving the framework, thus is able to solve most MKL and MT-MKL formulations, without need to develop and implement one algorithm for each model. The framework and algorithm is introduced in the next chapter.

It is worth to note that, the single kernel MTL model (4.19) is a special case of the above MT-MKL model (4.24): Problem (4.24) degrades to Problem (4.19), when $M = 1$. The trade-off between these two methods is obvious: For the single kernel model, it features a very simple and straightforward algorithm, as described before. However, in this case, there is a potential need for kernel selection via cross-validation, which requires an additional validation data set and computational effort. Clearly, in the case that the optimal kernel function is known a priori, or, the candidate set of the kernel function is small, this single kernel formulation is preferred. On the other hand, for the MT-MKL model (4.24), the kernel function is automatically generated via the MKL scheme, therefore saves the computation effort for kernel selection, as in the single kernel scenario. Besides, potentially better kernel function can be generated via the MKL scheme, due to the richer space for kernel selection. This brings higher classification accuracy to the MT-MKL model. However, the MT-MKL model requires the investigation of a new algorithm, which is a little more complicated than that of Problem (4.19). Therefore, this model is preferred, when it is not sure which kernel function to use, and when a potentially better generalization performance is highly desired, even though studying and implementing the new algorithm is required.

### Other related works

Prior to delving into the next major contribution of our research, which is a general purpose model framework and algorithm for both single-task and multi-task MKL problems, in this section, we discuss the relation between our proposed contents and four closely related papers.

First, we discuss [1] and [79]. Both of these two research works consider the Group-Lasso type regularizer to achieve different level of sparsity. Specifically, [1] considered the following regularizer

$$(\sum_{j=1}^{n}(\sum_{k=1}^{n_j}\|\boldsymbol{w}_{jk}\|)^s)^{\frac{2}{s}},\ s\geq 2 \tag{4.28}$$

where the weights of each of the $n$ groups is processed via a $L_1$-norm (*i.e.*the inner summation), and the $L_s$ norm regularizer is used for group level regularization. This regularizer can be customized to the scenario of MT-MKL, by equating $\boldsymbol{w}_{jk}$ to $\boldsymbol{w}_t^m$, yielding

$$(\sum_{t=1}^{T}(\sum_{m=1}^{M}\|\boldsymbol{w}_t^m\|)^s)^{\frac{2}{s}},\ s\geq 2 \tag{4.29}$$

Compare the above regularizer with the one that defined in $\mathcal{F}_{s,r}^{GL}$, we see the following difference: In $\mathcal{F}_{s,r}^{GL}$, instead of applying an $L_1$-norm, we used $\sum_{m=1}^{M}\frac{\|\boldsymbol{w}_t^m\|^2}{\theta_m}$, where $\theta_m$ is learned during the training from the feasible region $\Omega_r(\boldsymbol{\theta})$. Therefore, our regularizer subsumes MT-MKL with a common kernel function, a feature that is not allowed in (4.29).

In [79], the following regularizer is considered by the authors:

$$\sum_{m=1}^{M}(\sum_{t=1}^{T}\|\boldsymbol{w}_t^m\|^q)^{\frac{p}{q}},\ 0\leq p\leq 1, q\geq 1 \tag{4.30}$$

There exist two major differences between (4.30) and the regularizer that is defined in $\mathcal{F}_{s,r}^{GL}$. First, in (4.30), the $\boldsymbol{w}_t^m$'s that corresponds to the same kernel function is considered as a group, while $\mathcal{F}_{s,r}^{GL}$ treats each task as a group. Therefore, the order of the double summation is different between these two regularizers. Second, again, the MT-MKL with common kernel function setting is not

included in (4.30), due to the same reason that is stated above.

In the following, we discuss the difference between our work and the two theoretical works [68] and [47]. We first state following the regularizer that is considered in [68], which is introduced in Chapter 3:

$$\|\boldsymbol{w}\|_{\mathcal{M}} \triangleq \inf\{\sum_{M\in\mathcal{M}} \|\boldsymbol{v}_M\| : \boldsymbol{v}_M \in \mathcal{H}, \sum_{M\in\mathcal{M}} M\boldsymbol{v}_M = \boldsymbol{w}\} \tag{4.31}$$

where $\mathcal{M}$ is an almost countable set of symmetric bounded linear operators on $\mathcal{H}$. It is worth to note that, a specific regularizer needs to be either summation of several norms, or the infimum of such a summation over a feasible region, in order for it to be covered by this general expression (4.31). For our regularizer $(\sum_{t=1}^{T} \|\boldsymbol{w}_t\|^s)^{\frac{2}{s}}$, if attention is paid on the power outside the summation, it is not difficult to observe that it is not summation of norms. Also, it is not clear if it can be represented by an infimum expression. Therefore, (4.31) neither subsumes the regularizer that is given in $\mathcal{F}_s^{GL}$, nor the one in $\mathcal{F}_{s,r}^{GL}$. Besides, one advantage of our ERC bounds is that, it shows a clear relation between the bound and both the number of tasks $T$ and number of kernels $M$. Besides, based on our analysis, it is clearly to see that the logarithmic bound can be achieved. These are not the results that can be easily drawn from the bound that is derived in (4.31), even though our regularizers can be case as special cases of (4.31).

In [47], the authors considered the following hypothesis space

$$\mathcal{F} \triangleq \{\boldsymbol{x} \to [\langle \boldsymbol{w}_1, \boldsymbol{x}\rangle, \cdots, \langle \boldsymbol{w}_T, \boldsymbol{x}\rangle]' : F(\boldsymbol{W}) \leq F_{max}\} \tag{4.32}$$

where the function $F$ is $\beta$-*strongly convex* with respect to a norm $\|\cdot\|$, a condition that is satisfied in many situations, including when $F(\boldsymbol{W}) = \|\boldsymbol{W}\|_{1,1}$, $\|\boldsymbol{W}\|_{1,2}$, $\|\boldsymbol{W}\|_{2,1}$ and $\|\boldsymbol{W}\|_{2,2}$. However,

their work is very limited, since it is assume $\boldsymbol{W} \in \mathbb{R}^{m \times n}$. On the other hand, we assume our $\boldsymbol{w}^t$'s be an element of a RKHS, which can be potentially infinite-dimensional. Also, their regularizer cannot be applied to the MT-MKL scenario (see the one that introduced in (4.24)), where kernel combination coefficient $\boldsymbol{\theta}$ is learned during training, which is emphasized by our work.

# CHAPTER 5: A GENERAL PURPOSE FRAMEWORK AND ALGORITHM

Solving MKL problems has been an algorithmic difficulty for a long time. Usually, for models that are formulated in the primal domain, a block coordinate descent algorithm is the most frequently used. However, as introduced in Chapter 2, not all MKL formulations can be easily modeled in the primal domain; moreover, creating models directly in the dual domain is usually more straightforward: For example, it is easy to formulate the optimal-neighbor MKL problem, as given in Problem (2.43), in the dual domain, whose primal formulation is not trivially easy to create. This is even more true, when MT-MKL problems are considered. As an example, models with kernels that are partially shared across tasks can be easily formulated in the dual domain, which is not very straightforward to model in the primal domain.

However, dual formulations for single-task MKL and MT-MKL problems are, in the most cases, in the form of min-max optimization problem, which is by itself a difficult problem to solve. This problem is also encountered in our proposed MT-MKL model (4.27). We believe that such an algorithmic difficulty hinders both research progress and practical usage of single-task MKL and MT-MKL. In the research perspective, it is usually a pain for researchers to develop new algorithms for their models, especially when the research is originally focused on model formulation, instead of algorithms. In the practical perspective, it is a nightmare for users, who want to use single-task MKL and MT-MKL models, to understand complicated algorithms. This is even worse, when people want to compare different models, in which case several algorithms have to be interpreted and implemented.

Observing these issues, in this chapter, we propose a general framework for single-task MKL and MT-MKL problems in the dual domain, which covers most existing (and potentially many

future) models and our proposed MT-MKL model (4.27). Then, we develop an easy-to-implement algorithm to solve this framework, thus solves all formulations that are covered by our framework, including Problem (4.27). In this chapter, we discuss the framework and algorithm in detail.

## Framework

Given a set of training data $\{\boldsymbol{x}_i, y_i\} \in \mathcal{X} \times \mathcal{Y}, i = 1, \cdots, N$, consider a kernelized supervised learning problem, parametrized by $\boldsymbol{\alpha}$, which is given in the following form

$$\max_{\boldsymbol{\alpha} \in \boldsymbol{\Omega}(\boldsymbol{\alpha})} \bar{g}(\boldsymbol{\alpha}, \boldsymbol{K}) \tag{5.1}$$

where $\boldsymbol{K}$ is the kernel matrix, calculated by the kernel function $k$ on the training set. Also, we assume that the function $\bar{g}$ has a finite maximum, a finite number of local maxima with respect to $\boldsymbol{\alpha}$ in the feasible set $\boldsymbol{\Omega}(\boldsymbol{\alpha}) \subset \mathbb{R}^n$, and is affine with respect to the individual entries of $\boldsymbol{K}$.

It is not difficult to see that, several widely encountered machine learning models satisfy the above listed requirements, including the SVM, KRR, SVDD [92] and One-Class SVM [85], to mention a few major ones. In the following, we list their objective functions and feasible regions, in order to demonstrate how these models fit into the general optimization problem (5.1):

$$\bar{g}_{SVM}(\boldsymbol{\alpha}, \boldsymbol{K}) \triangleq \boldsymbol{\alpha}'\mathbf{1} - \frac{1}{2}\boldsymbol{\alpha}'\boldsymbol{Y}\boldsymbol{K}\boldsymbol{Y}\boldsymbol{\alpha}, \tag{5.2}$$

$$\text{with } \boldsymbol{\Omega}(\boldsymbol{\alpha}) \triangleq \{\boldsymbol{\alpha} \in \mathbb{R}^n : \mathbf{0} \preceq \boldsymbol{\alpha} \preceq C\mathbf{1}, \boldsymbol{\alpha}'\boldsymbol{y} = 0\}$$

$$\bar{g}_{KRR}(\boldsymbol{\alpha}, \boldsymbol{K}) \triangleq 2\boldsymbol{\alpha}'\boldsymbol{y} - \boldsymbol{\alpha}'(\lambda \boldsymbol{I} + \boldsymbol{K})\boldsymbol{\alpha}, \tag{5.3}$$

with $\boldsymbol{\Omega}(\boldsymbol{\alpha}) \triangleq \mathbb{R}^n$

$$\bar{g}_{SVDD}(\boldsymbol{\alpha}, \boldsymbol{K}) \triangleq \boldsymbol{\alpha}'\boldsymbol{k} - \boldsymbol{\alpha}'\boldsymbol{K}\boldsymbol{\alpha}, \tag{5.4}$$

with $\boldsymbol{\Omega}(\boldsymbol{\alpha}) \triangleq \{\boldsymbol{\alpha} \in \mathbb{R}^n : \mathbf{0} \preceq \boldsymbol{\alpha} \preceq C\mathbf{1}, \boldsymbol{\alpha}'\mathbf{1} = 0\}$

$$\bar{g}_{OCSVM}(\boldsymbol{\alpha}, \boldsymbol{K}) \triangleq -\boldsymbol{\alpha}'\boldsymbol{K}\boldsymbol{\alpha}, \tag{5.5}$$

with $\boldsymbol{\Omega}(\boldsymbol{\alpha}) \triangleq \{\boldsymbol{\alpha} \in \mathbb{R}^n : \mathbf{0} \preceq \boldsymbol{\alpha} \preceq \frac{1}{\nu l}\mathbf{1}, \boldsymbol{\alpha}'\mathbf{1} = 0\}$

In the above examples, $\boldsymbol{y} \triangleq [y_1, \cdots, y_n]'$ is the target vector, $\boldsymbol{k} \triangleq [k(\boldsymbol{x}_1, \boldsymbol{x}_1), \cdots, k(\boldsymbol{x}_n, \boldsymbol{x}_n)]'$, $\boldsymbol{Y} \triangleq \mathrm{diag}(y_1, \cdots, y_n)$, and $\mathbf{1}$ is the all-ones vector with appropriate dimension.

To extend Problem (5.1) to a MTL setting, we assume that we have $T$ such tasks. Then, the dual domain MT-MKL framework can be formulated as follows:

$$\min_{\boldsymbol{\theta} \in \boldsymbol{\Psi}(\boldsymbol{\theta})} \max_{\boldsymbol{a} \in \boldsymbol{\Omega}(\boldsymbol{a})} \sum_{t=1}^{T} \bar{g}(\boldsymbol{\alpha}_t, \sum_{m=1}^{M} \theta_t^m \boldsymbol{K}_t^m) \tag{5.6}$$

where $\boldsymbol{\Psi}(\boldsymbol{\theta})$ denotes the feasible region of $\boldsymbol{\theta}$.

Due to the generality of the above framework, it covers several existing single-task MKL model and MT-MKL models. To give an example, in the single-task MKL scenario, *i.e.*, when $T = 1$, it

covers the popular $L_p$-norm MKL model [50] by letting

$$\bar{g}(\boldsymbol{\alpha}, \sum_{m=1}^{M} \theta_m \boldsymbol{K}_m) \triangleq \boldsymbol{\alpha}'\mathbf{1} - \frac{1}{2}\boldsymbol{\alpha}'\boldsymbol{Y}(\sum_{m=1}^{M} \theta_m \boldsymbol{K}_m)\boldsymbol{Y}\boldsymbol{\alpha} \tag{5.7}$$

with $\boldsymbol{\Omega}(\boldsymbol{\alpha}) \triangleq \{\boldsymbol{\alpha} \in \mathbb{R}^n : \mathbf{0} \preceq \boldsymbol{\alpha} \preceq C\mathbf{1}, \boldsymbol{\alpha}'\boldsymbol{y} = 0\}$ and $\boldsymbol{\Psi}(\boldsymbol{\theta}) = \{\boldsymbol{\theta} : \|\boldsymbol{\theta}\|_p \leq 1, \boldsymbol{\theta} \succeq \mathbf{0}\}$.

Clearly, in the above setting, with $p = 2$, the $L_2$-norm MKL method [51] is also covered by the above framework.

In the scenario of MT-MKL, the model that allows partially shared kernel function, which is introduced in [91], can be specified by our framework as follows:

$$\bar{g}(\boldsymbol{\alpha}_t, \sum_{m=1}^{M} \theta_t^m \boldsymbol{K}_t^m) \triangleq \boldsymbol{\alpha}_t'\mathbf{1} - \frac{1}{2}\boldsymbol{\alpha}_t'\boldsymbol{Y}_t(\sum_{m=1}^{M} \theta_t^m \boldsymbol{K}_t^m)\boldsymbol{Y}_t\boldsymbol{\alpha}_t \tag{5.8}$$

with $\boldsymbol{\Omega}(\boldsymbol{\alpha}) \triangleq \{\boldsymbol{\alpha}_t \in \mathbb{R}^n : \mathbf{0} \preceq \boldsymbol{\alpha}_t \preceq C\mathbf{1}, \boldsymbol{\alpha}_t'\boldsymbol{y} = 0, \forall t = 1, \cdots, T\}$ and $\boldsymbol{\Psi}(\boldsymbol{\theta}) = \{\boldsymbol{\theta} : \theta_t^m = \zeta_m + \gamma_t^m, \zeta_m \geq 0, \gamma_t^m \geq 0, \sum_{m=1}^{M} \theta_t^m = 1, \sum_{m=1}^{M}\sum_{t=1}^{T} \gamma_t^m \leq \beta\}$.

Therefore, the two special models of the above setting, namely the Common Space (CS) and Independent Space (IS) MT-MKL models, are also covered by our framework. To be more concrete, the CS model can be achieved by letting $\theta_t^m = \zeta_m, \forall t = 1, \cdots, T, m = 1, \cdots, M$, so that all tasks share a common kernel function, which is parametrized by the $\zeta_m$'s. On the other hand, the IS model is simply obtained by letting $\boldsymbol{\theta}_t = [\theta_t^1, \cdots, \theta_t^M]'$, where each task occupies its own kernel function. Of course, in both cases, appropriate feasible region should be added to the kernel coefficients.

As mentioned before, one objective of developing the above framework is to let it cover our pre-

viously proposed MT-MKL model Problem (4.27), which features the HS $\mathcal{F}_{s,r}$. Then, by developing a simple algorithm that solve the framework, the originally complicated Problem (4.27) can therefore be easily solved. To see how Problem (4.27) is a special case of Problem (5.6), first, a simple variable change for Problem (4.27) is needed to prevent conflicted notation: $\boldsymbol{\beta}_t \leftarrow \boldsymbol{\alpha}_t, \forall t = 1, \cdots, T$, *i.e.*, Problem (4.27) now becomes

$$\begin{aligned} \min_{\boldsymbol{\theta}} \max_{\boldsymbol{\beta},\boldsymbol{\lambda}} \; & \sum_{t=1}^{T} (\boldsymbol{\beta}_t{}'\mathbf{1} - \frac{1}{2\lambda_t}\boldsymbol{\beta}_t{}'\boldsymbol{Y}_t(\sum_{m=1}^{M}\theta_m \boldsymbol{K}_t^m)\boldsymbol{Y}_t\boldsymbol{\beta}_t) \\ \text{s.t.} \; & \mathbf{0} \preceq \boldsymbol{\beta}_t \preceq C\mathbf{1}, \; \boldsymbol{\beta}_t{}'\boldsymbol{y}_t = 0, \; \forall t \\ & \boldsymbol{\lambda} \succeq \mathbf{0}, \|\boldsymbol{\lambda}\|_{\frac{s}{s-2}} \leq 1 \\ & \boldsymbol{\theta} \succeq \mathbf{0}, \|\boldsymbol{\theta}\|_r \leq 1 \end{aligned} \tag{5.9}$$

Then, define $\boldsymbol{\alpha}_t \triangleq [\boldsymbol{\beta}_t', \lambda_t]'$, we easily see the above problem is a special case of Problem (5.6) with the following specification:

$$\bar{g}(\boldsymbol{\alpha}_t, \sum_{m=1}^{M}\theta_t^m \boldsymbol{K}_t^m) \triangleq \boldsymbol{\beta}_t{}'\mathbf{1} - \frac{1}{2\lambda_t}\boldsymbol{\beta}_t{}'\boldsymbol{Y}_t(\sum_{m=1}^{M}\theta_m \boldsymbol{K}_t^m)\boldsymbol{Y}_t\boldsymbol{\beta}_t \tag{5.10}$$

with $\boldsymbol{\Omega}(\boldsymbol{\alpha}) \triangleq \{\boldsymbol{\beta}_t \in \mathbb{R}^n : \mathbf{0} \preceq \boldsymbol{\beta}_t \preceq C\mathbf{1}, \boldsymbol{\beta}_t'\boldsymbol{y} = 0, \forall t = 1, \cdots, T, \boldsymbol{\lambda} \succeq \mathbf{0}, \|\boldsymbol{\lambda}\|_{\frac{s}{s-2}} \leq 1\}$ and $\boldsymbol{\Psi}(\boldsymbol{\theta}) = \{\boldsymbol{\theta} : \boldsymbol{\theta} \succeq \mathbf{0}, \|\boldsymbol{\theta}\|_r \leq 1\}$.

Clearly, due to the generality of the proposed framework, it is highly beneficial to develop an algorithm to solve it. In this case, it will be able to solve all the models that are covered by the framework. This algorithm is introduced in the following sections. In the next section, we first convert Problem (5.6) to an equivalent SIP problem, followed by showing the equivalence between general SIP problems and Exact Penalty Function (EPF)-based problems. The latter result will allow us to cast Problem (5.6) as an EPF-based problem, which can be easily solved.

Exact Penalty Function Method

The proposed framework, which is given as a min-max problem (5.6), is not generally easy to solve. To conquer this problem, we first equivalently transform Problem (5.6) to the following epigraph problem:

$$\begin{aligned} \min_{\omega\in\mathbb{R},\boldsymbol{\theta}\in\boldsymbol{\Psi}(\boldsymbol{\theta})} \quad & \omega \\ \text{s.t.} \quad & \sum_{t=1}^{T} \bar{g}(\boldsymbol{\alpha}_t, \sum_{m=1}^{M} \theta_t^m \boldsymbol{K}_t^m) \leq \omega, \ \forall \boldsymbol{a} \in \boldsymbol{\Omega}(\boldsymbol{a}). \end{aligned} \tag{5.11}$$

At the first glance, such an equivalent transformation is not very helpful, since the above SIP problem is still not easy to solve. However, as will be shown next, a general SIP problem can be equivalently solved by solving the corresponding EPF-based problem [95]. Then, we will propose a simple and easy-to-implement algorithm for solving the EPF-based problem, based on our MT-MKL framework. As a result, this algorithm will be able to solve the proposed framework (5.6).

Next, before delving into the algorithm, we introduce the relation between the general SIP problem and the corresponding EPF-based problem.

**General SIP problem**

Assume $f$, $g$, $l_u$'s and $r_v$'s are continuously differentiable functions. Consider the general SIP

problem

$$\begin{aligned} &\min_{\boldsymbol{x}\in\mathbb{R}^n} f(\boldsymbol{x}) \\ &\text{s.t. } g(\boldsymbol{a},\boldsymbol{x}) \leq 0,\ \forall \boldsymbol{a} \in \boldsymbol{\Omega}(\boldsymbol{a}); \\ &\quad l_u(\boldsymbol{x}) = 0, u = 1, \cdots, U; \\ &\quad r_v(\boldsymbol{x}) \leq 0, v = 1, \cdots, V. \end{aligned} \tag{5.12}$$

Clearly, Problem (5.11) is a general SIP problem, since it is a a special case of Problem (5.12). This can be observed by defining $\boldsymbol{x} \triangleq [\omega, \boldsymbol{\theta}']'$, $\boldsymbol{a} \triangleq [\boldsymbol{\alpha}_1', \cdots, \boldsymbol{\alpha}_T']'$, $f(\boldsymbol{x}) \triangleq \omega$, $g(\boldsymbol{a},\boldsymbol{x}) \triangleq \sum_{t=1}^{T} \bar{g}(\boldsymbol{\alpha}_t, \sum_{m=1}^{M} \theta_t^m \boldsymbol{K}_t^m) - \omega$, and letting the constraints $l_u$'s and $r_v$'s define $\boldsymbol{\Psi}(\boldsymbol{\theta})$.

**EPF-based problem**

Given a fixed $\boldsymbol{x}$ and $\eta > 0$, assume there are $N(\boldsymbol{x})$ local maxima of $g(\boldsymbol{a},\boldsymbol{x})$, denoted as $\boldsymbol{a}_i^* \in \boldsymbol{\Omega}(\boldsymbol{a}), i = 1, \cdots, N(\boldsymbol{x})$. Assume that the $\boldsymbol{a}_i^*$'s satisfy $g(\boldsymbol{a}_i^*,\boldsymbol{x}) \geq -\eta$, and $g(\boldsymbol{a}_i^*,\boldsymbol{x}) \neq g(\boldsymbol{a}_j^*,\boldsymbol{x})$, $\forall \boldsymbol{a}_i^*, \boldsymbol{a}_j^* \in \Omega(\boldsymbol{a})$. Let this set of local maxima be denoted as $E(\boldsymbol{x}) \triangleq \{\boldsymbol{a}_i^*\}_{i=1}^{N(\boldsymbol{x})}$, and $I(\boldsymbol{x}) \triangleq \{1, 2, \ldots, N(\boldsymbol{x})\}$ be the index set. Obviously, for different $\boldsymbol{x}$, the local maxima will be different. Therefore, each $\boldsymbol{a}_i^*$ is implicitly depend on $\boldsymbol{x}$. If we assume that the Implicit Function Theorem conditions hold, then there is a function $\boldsymbol{a}_i$ of $\boldsymbol{x}$, such that $\boldsymbol{a}_i^* = \boldsymbol{a}_i(\boldsymbol{x})$. Then, by defining $h_i(\boldsymbol{x}) \triangleq g(\boldsymbol{a}_i(\boldsymbol{x}),\boldsymbol{x})$, the EPF $P(\boldsymbol{x})$ that is introduced in [95] is defined in a *neighborhood* of $\boldsymbol{x}$:

$$P(\boldsymbol{x}) = f(\boldsymbol{x}) + \nu \sum_{i\in I(\boldsymbol{x})} h_i(\boldsymbol{x})_+ \tag{5.13}$$

where $\nu > 0$ and $h_i(\boldsymbol{x})_+ = \max\{0, h_i(\boldsymbol{x})\}$.

Based on the EPF $P(\boldsymbol{x})$, the EPF-based optimization problem can be developed:

$$\begin{aligned} &\min_{\boldsymbol{x}\in\mathbb{R}^n} \; P(\boldsymbol{x}) \\ &\text{s.t. } l_u(\boldsymbol{x}) = 0, \forall u; \; r_v(\boldsymbol{x}) \leq 0, \forall v. \end{aligned} \tag{5.14}$$

The next theorem shows the relation between Problem (5.12) and Problem (5.14): Problem (5.12) can be solved by solving Problem (5.14).

**Theorem 16.** *Let $f$, $g$, $l_u$'s and $r_v$'s in Problem (5.12) and Problem (5.14) be continuously differentiable, and let the EPF function $P(\boldsymbol{x})$ be defined as $Equation\ (5.13)$. Suppose for fixed $\boldsymbol{x}$, there are finite number of $\boldsymbol{a}_s \in \boldsymbol{\Omega}(\boldsymbol{a})$, $s = 1, \cdots, S$, such that $g(\boldsymbol{a}_s, \boldsymbol{x}) = 0$. If $\hat{\boldsymbol{x}}$ is in the feasible region of Problem (5.12) and solves Problem (5.14), then $\hat{\boldsymbol{x}}$ is a Karush-Kuhn-Tucker (KKT) point of Problem (5.12).*

The above theorem illustrated that, although the general SIP problem (5.12) is complicated due to its infinite constraint $g(\boldsymbol{a}, \boldsymbol{x}) \leq 0, \forall \boldsymbol{a} \in \boldsymbol{\Omega}(\boldsymbol{a})$, it can be equivalently solved by solving the EPF-based problem (5.14), which is easier to optimize, since the absence of the infinite constraint. As our proposed MT-MKL framework (5.11) is a general SIP problem, the above theorem immediately provides a way to simplify our framework, by transforming it to the corresponding EPF-based problem, which turns out having a simple and easy-to-implement algorithm. This algorithm is introduced in the next section.

## Algorithm

Before introducing the algorithm, we re-emphasize that the general SIP problem can be specialized to our proposed framework 5.11, by letting $\boldsymbol{x} \triangleq [\omega, \boldsymbol{\theta}']'$, $\boldsymbol{a} \triangleq [\boldsymbol{\alpha}_1', \cdots, \boldsymbol{\alpha}_T']'$, $f(\boldsymbol{x}) \triangleq \omega$, $g(\boldsymbol{a}, \boldsymbol{x}) \triangleq$

$\sum_{t=1}^{T} \bar{g}(\boldsymbol{\alpha}_t, \sum_{m=1}^{M} \theta_t^m \boldsymbol{K}_t^m) - \omega$, and letting the constraints $l_u$'s and $r_v$'s define $\boldsymbol{\Psi}(\boldsymbol{\theta})$. In this section, we solve the corresponding EPF-based problem in this MT-MKL context.

The EPF-based Problem (5.14) is generally solved via a descent method, which is introduced in [95]. Specifically, given $\boldsymbol{x}^{(k)}$ in the $k$-th iteration, $\boldsymbol{x}^{(k+1)} = \boldsymbol{x}^{(k)} + \epsilon^{(k)} \boldsymbol{d}^{(k)}$, given the descent direction $\boldsymbol{d}^{(k)}$. The convergence of such an algorithm for solving the EPF-based problem is provided in [95]: the sequence $\{\boldsymbol{x}^{(k)}\}$ converges to the KKT point of Problem (5.12), as long as the following conditions are satisfied: First, $\nu$ is large enough. Second, the step length $\epsilon$ is small enough. Third, $f$ and $g$ are doubly-differentiable functions with bounded second-order derivatives. These three conditions are not difficult to be satisifed. First, we will prove that, for our proposed framework, the first condition is satisfied as long as $\nu > 1$. Second, it is not difficult to choose the step length $\epsilon^{(k)}$ in the $k$-th iteration, as will be described later. Finally, the last condition is satisfied in many frequently encountered models, such as the ones that we listed from (5.2) to (5.5).

According to the above description, it is clear that, in the $k$-th iteration, given $\boldsymbol{x}^{(k)}$, which is $\boldsymbol{\theta}^{(k)}$ and $\omega^{(k)}$ in our problem, two crucial components should be calculated, namely the set of local maxima $E(\boldsymbol{x}^{(k)})$, and the descent direction $\boldsymbol{d}^{(k)}$. It is straightforward to calculate $E(\boldsymbol{x}^{(k)})$, which involves solving the maximization problem

$$\max_{\boldsymbol{a} \in \boldsymbol{\Omega}(\boldsymbol{a})} \sum_t \bar{g}(\boldsymbol{\alpha}_t, \sum_m \theta_t^{m,(k)} \boldsymbol{K}_t^m) \tag{5.15}$$

When the above maximization problem is concave, there is only one local maxima, which is therefore easy to find. On the other hand, if it is not concave, it will be more involved to find all local maxima. Fortunately, for many frequently used models, the above problem is concave, such as the ones that are listed from (5.2) to (5.5).

On the other hand, the next theorem shows how to find the descent direction $\boldsymbol{d}^{(k)}$, which is the key

step that makes our algorithm simple and easy to implement.

**Theorem 17.** *Let $\bar{g}(\boldsymbol{\alpha}, \boldsymbol{K})$ be affine with respect to the individual entries of $\boldsymbol{K}$. Suppose it has a finite maximum and a finite number of local maxima with respect to $\boldsymbol{\alpha}$ in the feasible set $\boldsymbol{\Omega}(\boldsymbol{\alpha})$. Let $l_u$'s and $r_v$'s be continuous and differentiable. Let $E(\boldsymbol{x})$ and $I(\boldsymbol{x})$ as defined in Section 5, and $\boldsymbol{a}_i^{(k)} \triangleq [\boldsymbol{\alpha}_{1,i}^{(k)'}, \cdots, \boldsymbol{\alpha}_{T,i}^{(k)'}]' \in E(\boldsymbol{x}^{(k)})$, $i \in I(\boldsymbol{x}^{(k)}) = \{1, \cdots, N(\boldsymbol{x}^{(k)})\}$. Consider the following $N(\boldsymbol{x}^{(k)})$ problems, $\forall\ i \in I(\boldsymbol{x}^{(k)})$:*

$$\begin{aligned} &\min_{\boldsymbol{\theta}} \sum_{t=1}^{T} \bar{g}(\boldsymbol{\alpha}_{t,i}^{(k)}, \sum_{m=1}^{M} \theta_t^m \boldsymbol{K}_t^m) \\ &\textit{s.t. } l_u(\boldsymbol{x}) = 0, \forall u;\ r_v(\boldsymbol{x}) \leq 0, \forall v. \end{aligned} \tag{5.16}$$

*Let $\hat{\boldsymbol{\theta}}_i$, which consists of all elements $\hat{\theta}_{t,i}^m$'s, be the solution to the $i$-th problem. Also, let $i_0 = \arg\min_{i \in I(\boldsymbol{x}^{(k)})} \max_{j \in I(\boldsymbol{x}^{(k)})} \sum_{t=1}^{T} \bar{g}(\boldsymbol{\alpha}_{t,j}^{(k)}, \sum_{m=1}^{M} \hat{\theta}_{t,i}^m \boldsymbol{K}_t^m)$ and $\hat{\omega} = \min_{i \in I(\boldsymbol{x}^{(k)})} \max_{j \in I(\boldsymbol{x}^{(k)})} \sum_{t=1}^{T} \bar{g}(\boldsymbol{\alpha}_{t,j}^{(k)}, \sum_{m=1}^{M} \hat{\theta}_{t,i}^m \boldsymbol{K}_t^m)$. Finally, let $\hat{\boldsymbol{x}} = [\hat{\omega}, \hat{\boldsymbol{\theta}}_{i_0}']'$. Then, $\boldsymbol{d}^{(k)} = \hat{\boldsymbol{x}} - \boldsymbol{x}^{(k)}$ is a descent direction for the EPF-based Problem (5.14) when $\nu > 1$.*

In the case of $\bar{g}$ being concave with respect to $\boldsymbol{\alpha}$, as stated above, only one maxima exists, *i.e.*, $N(\boldsymbol{x}^{(k)}) = 1$. This leads to an even simpler way to find the descent direction $\boldsymbol{d}^{(k)}$, which is introduced in the next corollary.

**Corollary 2.** *Let $\bar{g}(\boldsymbol{\alpha}, \boldsymbol{K})$ be affine with respect to the individual entries of $\boldsymbol{K}$ and is concave with respect to $\boldsymbol{\alpha}$. Let $l_u$'s and $r_v$'s be continuous and differentiable. Let $E(\boldsymbol{x})$ and $I(\boldsymbol{x})$ as defined in Section 5, and $\boldsymbol{a}^{(k)} \triangleq [\boldsymbol{\alpha}_1^{(k)'}, \cdots, \boldsymbol{\alpha}_T^{(k)'}]' \in E(\boldsymbol{x}^{(k)})$. Consider the problem*

$$\begin{aligned} &\min_{\boldsymbol{\theta}} \sum_{t=1}^{T} \bar{g}(\boldsymbol{\alpha}_t^{(k)}, \sum_{m=1}^{M} \theta_t^m \boldsymbol{K}_t^m) \\ &\textit{s.t. } l_u(\boldsymbol{x}) = 0, \forall u;\ r_v(\boldsymbol{x}) \leq 0, \forall v. \end{aligned} \tag{5.17}$$

*Denote* $\hat{\boldsymbol{\theta}}$ *be the solution to the above problem and* $\hat{\omega} \triangleq \sum_{t=1}^{T} \bar{g}(\boldsymbol{\alpha}_t^{(k)}, \sum_{m=1}^{M} \hat{\theta}_t^m \boldsymbol{K}_t^m)$. *Let* $\hat{\boldsymbol{x}} \triangleq [\hat{\omega}, \hat{\boldsymbol{\theta}}']'$. *Then,* $\boldsymbol{d}^{(k)} = \hat{\boldsymbol{x}} - \boldsymbol{x}^{(k)}$ *is a descent direction for the EPF-based Problem (5.14) when* $\nu > 1$.

The above discussion is summarized in Algorithm 1. Note that it only focuses on the case when $\bar{g}$ is concave with respect to $\boldsymbol{\alpha}$, since it is the most commonly encountered setting in single-task MKL and MT-MKL.

**Algorithm 1** Algorithm for solving Problem (5.14)

Choose $M$ kernel functions and calculate the kernel matrices $\boldsymbol{K}_t^m$. Randomly initialize $\boldsymbol{\theta}_0 \succeq \mathbf{0}$ and $\boldsymbol{\theta} \in \Psi(\boldsymbol{\theta})$. Initialize $\eta$ and $\epsilon_0$ to small positive values. $k \leftarrow 0$.
**while** not converged **do**
  $\boldsymbol{a}^{(k)} \leftarrow \arg\max_{\boldsymbol{a} \in \boldsymbol{\Omega}(\boldsymbol{a})} \sum_t \bar{g}(\boldsymbol{\alpha}_t, \sum_m \theta_t^{m,(k)} \boldsymbol{K}_t^m)$
  $\hat{\boldsymbol{\theta}}^{(k)} \leftarrow$ solve Problem (5.17) given $\boldsymbol{a}^{(k)}$
  $\hat{\omega}^{(k)} \leftarrow \sum_t \bar{g}(\boldsymbol{\alpha}_t^{(k)}, \sum_m \hat{\theta}_t^{m,(k)} \boldsymbol{K}_t^m)$
  $\boldsymbol{\theta}^{(k+1)} \leftarrow \boldsymbol{\theta}^{(k)} + \epsilon^{(k)}(\hat{\boldsymbol{\theta}}^{(k)} - \boldsymbol{\theta}^{(k)})$
  $\omega^{(k+1)} \leftarrow \omega^{(k)} + \epsilon^{(k)}(\hat{\omega}^{(k)} - \omega^{(k)})$
  $k \leftarrow k + 1$
**end while**

To choose the step length $\epsilon^{(k)}$ in the $k$-th iteration, as suggested in [95], one method is to select $\epsilon^{(k)}$ be the largest element in the set $\{1, \beta, \beta^2, \cdots\}$, for some $0 < \beta < 1$, such that

$$[P(\boldsymbol{x}^{(k)} + \epsilon^{(k)} \boldsymbol{d}^{(k)}) - P(\boldsymbol{x}^{(k)})]/\epsilon^{(k)} G^{(k)} \geq \sigma \tag{5.18}$$

where $P$ is the EPF, which is defined in Problem (5.14), $\sigma$ is some constant that satisfies $0 < \sigma < 1$, and $G^{(k)}$ is the directional derivative of $P$ with respect to $\boldsymbol{x}$ in the $k$-th step. It is not difficult to

show that

$$\begin{aligned} G^{(k)} = & f(\hat{\boldsymbol{x}}^{(k)}) + \nu\ g(\boldsymbol{a}^{(k)}, \hat{\boldsymbol{x}}^{(k)})_+ \\ & - f(\boldsymbol{x}^{(k)}) - \nu\ g(\boldsymbol{a}^{(k)}, \boldsymbol{x}^{(k)})_+. \end{aligned} \tag{5.19}$$

where $\hat{\boldsymbol{x}}^{(k)} = [\hat{\omega}^{(k)}, \hat{\boldsymbol{\theta}}^{(k)'}]'$ and $\boldsymbol{x}^{(k)} = [\omega^{(k)}, \boldsymbol{\theta}^{(k)'}]'$.

For the case of $\bar{g}$ begin non-concave with respect to $\boldsymbol{\alpha}$, the calculation of $G^{(k)}$ is slightly more complicated. Since this is a situation that is rarely encountered in practice, the discussion of calculating $G^{(k)}$ is therefore introduced in the proof of Theorem 17 in the Appendix.

*Analysis*

In this sub-section, we emphasize several advantages of the above proposed algorithm.

First, one benefit of the proposed framework (5.6) is that, it is very general, and therefore covers many single-task MKL and MT-MKL models, and any other optimization problems that are special cases of the framework. This is majorly due to the relatively mild constraints on the function $\bar{g}$, which are introduced at the beginning of this chapter. It is worth to note that there is even no need for $\bar{g}$ to be concave with respect to $\boldsymbol{\alpha}$. As a result, the proposed algorithm is able to solve a wide range of models, and largely solves the algorithmic difficulty of current single-task MKL and MT-MKL research, as discussed in Chapter 3.

Secondly, in the first step of each iteration of the algorithm, it is needed to solve a maximization problem. Such a problem is usually very easy to solve. In a commonly encountered situation, where the feasible region of each $\boldsymbol{\alpha}_t$ is independent with each other, this problem can be separated into $T$ independent maximization problems. As a concrete example, for the MT-MKL model that

is proposed in [91] and discussed in (5.8), $T$ independent SVM problems should be solved in this step. This can be done efficiently, due to the availability of efficient SVM solvers, such as `LIBSVM` [18]. On the other hand, in a more complicated scenario, for the MT-MKL model that we developed in Chapter 4 and discussed in (5.10), the maximization problem is also not difficult to solve; it can be optimized with a block-coordinate descent algorithm with $\boldsymbol{\beta}$ as a group and $\boldsymbol{\lambda}$ as another: When $\boldsymbol{\lambda}$ is fixed and optimize with respect to $\boldsymbol{\beta}$, again, $T$ SVM problems are involved; when $\boldsymbol{\beta}$ is fixed, optimizing with respect to $\boldsymbol{\lambda}$ has a closed-form solution, based on Theorem 5. It is easy to verify the simplicity of this step by adapting existing kernel machines with our framework.

Thirdly, in the second step of each iteration of our algorithm, a minimization problem is involved. Compared to the maximization step that is discussed above, such a minimization problem is even easier to optimize, in many common encountered scenarios. Since $\bar{g}$ is affine with respect to the individual entries of the kernel matrix, in essence, this step is solving the following problem:

$$\begin{aligned} &\min_{\boldsymbol{\theta}} \ \boldsymbol{c}'\boldsymbol{\theta} \\ &\text{s.t. } l_u(\boldsymbol{x}) = 0, \forall u; \ r_v(\boldsymbol{x}) \leq 0, \forall v. \end{aligned} \tag{5.20}$$

where $\boldsymbol{c}$ is the coefficient vector. For many existing models, the constraints of the above problem are set so that a closed-form solution exists. One obvious example is our proposed model (4.27), which, when solved via our proposed algorithm, has closed-form solution in this step, due to Theorem 15. With the same reason, the closed-form solution also exists for the $L_p$-norm MKL model, if solved by our algorithm. As a more complicated example, the model that is proposed in

[91] involves the following minimization problem in this step:

$$\begin{aligned} &\min_{\boldsymbol{\theta}} \ \boldsymbol{c}'\boldsymbol{\theta} \\ &\text{s.t. } \theta_t^m = \zeta_m + \gamma_t^m, \zeta_m \geq 0, \ \gamma_t^m \geq 0, \forall m, t; \\ &\quad \sum_{m=1}^{M} \theta_t^m = 1, \ \forall t; \ \sum_{m=1}^{M} \sum_{t=1}^{T} \gamma_t^m \leq \beta. \end{aligned} \tag{5.21}$$

At the first glance, the above problem is difficult to solve. However, if a block coordinate descent algorithm is utilized with $\boldsymbol{\zeta} = [\zeta_1, \cdots, \zeta_M]$ as a group and $\boldsymbol{\gamma} = [\boldsymbol{\gamma}_1{}', \cdots, \boldsymbol{\gamma}_T{}']'$ as another, in each step, closed-form solution exist.

The following proposition provides closed-form solutions for a variety of constraints on $\boldsymbol{\theta}$, which covers most existing models.

**Proposition 2.** *Let $\boldsymbol{c} \triangleq [c_1, \cdots, c_n]' \in \mathbb{R}^n$ be the concatenation of $T$ vectors $\boldsymbol{c}_1 \in \mathbb{R}^{n_1}, \cdots, \boldsymbol{c}_T \in \mathbb{R}^{n_T}$ with $\sum_{t=1}^{T} n_t = n$. Suppose for $t = 1, \cdots, T$, each $\boldsymbol{c}^t$ has at least one negative element. Similarly, let $\boldsymbol{\theta} \triangleq [\theta_1, \cdots, \theta_n] \in \mathbb{R}^n$ be the concatenation of $\boldsymbol{\theta}_1 \in \mathbb{R}^{n_1}, \cdots, \boldsymbol{\theta}_T \in \mathbb{R}^{n_T}$. The optimization problem*

$$\begin{aligned} &\min_{\boldsymbol{\theta} \in \mathbb{R}^n} \ \boldsymbol{c}'\boldsymbol{\theta} \\ &\quad \textit{s.t. } \boldsymbol{\theta} \succeq \mathbf{0}, \ (\sum_{t=1}^{T} \|\boldsymbol{\theta}_t\|_p^q)^{1/q} \leq a \end{aligned} \tag{5.22}$$

*has closed-form solution*

$$\begin{cases} \hat{\boldsymbol{\theta}}_t = \frac{\sigma_t(\tilde{\boldsymbol{c}}_t)^r}{\|\tilde{\boldsymbol{c}}_t\|_{r+1}^r},\ \forall t, & p>1, q>1 \\ \hat{\boldsymbol{\theta}}_t = \frac{a(\tilde{\boldsymbol{c}}_t)^r}{\|\tilde{\boldsymbol{c}}_t\|_{r+1}^r}[t=t_0],\ \forall t, & p>1, q=1 \\ \hat{\boldsymbol{\theta}} = \frac{a(\boldsymbol{c}^*)^s}{\|\boldsymbol{c}^*\|_{s+1}^s}, & p=1, q>1 \\ \hat{\boldsymbol{\theta}} = a\boldsymbol{e}_j, & p=1, q=1 \end{cases} \tag{5.23}$$

*where* $\sigma_t = \frac{a\|\tilde{\boldsymbol{c}}_t\|_{r+1}^s}{(\sum_{t=1}^T \|\tilde{\boldsymbol{c}}_t\|_{r+1}^{s+1})^{1/q}}$, $[t=t_0]=1$ *if* $t=t_0$ *and* $0$ *otherwise,* $t_0 = \arg\min_t \{\|\tilde{\boldsymbol{c}}_t\|_{r+1}\}$, $r \triangleq \frac{1}{p-1}$, $s \triangleq \frac{1}{q-1}$, $\tilde{\boldsymbol{c}}_t$ *is a vector with elements* $[\max\{-c_t^m, 0\}]_{m=1}^M$ *and* $(\tilde{\boldsymbol{c}}_t)^r$ *denotes element-wise exponentiation of vectors. In the third branch of (5.23),* $\boldsymbol{c}^*$ *is concatenation of* $\boldsymbol{c}_t^* \triangleq \boldsymbol{c}_t \circ \boldsymbol{e}_t^{i_t}, t = 1, \cdots, T$*, where* $\circ$ *is element-wise multiplication of vectors. Here* $\boldsymbol{e}_t^{i_t} \in \mathbb{R}^{n_t}$ *is a vector, whose* $i_t$*-th entry is* $1$*, while the remaining are* $0$*, and* $i_t = \arg\min_i\{c_t^i\}$*. Similarly, in the last branch of (5.23),* $\boldsymbol{e}_j$ *is an all-*$0$ *vector except the* $j$*-th entry is* $1$*, and* $j = \arg\min_i\{c_i\}$*. Finally, if* $\exists\ t$ *such that* $\boldsymbol{c}_t \succeq \boldsymbol{0}$*, then* $\hat{\boldsymbol{\theta}}_t = \boldsymbol{0}$*.*

When $\bar{g}$ is not concave with respect to $\boldsymbol{\alpha}$, as long as its number of local maxima is finite, *i.e.*, $N(\boldsymbol{x}^{(k)}) < +\infty$, in this step, we only need to solve $N(\boldsymbol{x}^{(k)})$ problems that are in the form of Problem (5.17). This is not difficult, since Problem (5.17) is, in most cases, easy to optimize, as analyzed above.

# CHAPTER 6: EXPERIMENTS

In Chapter 4, we proposed a pair of new HSs, one for the single kernel MTL problems, and another one for the MT-MKL scenario. We in Chapter 4 concluded that, the ERC of each HS is monotonically increasing with respect to $s$, and so is the corresponding ERC bound. This important observation provides a useful guidance for model selection: a small value of $s$ leads to lower ERC value, and potentially gives better generalization performance. In order to verify the monotonicity of the ERCs and their bounds, and to investigate the tightness of the bound, in this chapter, we adapt real world data sets, and calculate the ERC values (via a Monte Carlo simulation) and the corresponding bounds based on the data sets. It is expected that both the ERC values and the bounds to increase, when $s$ increases. Besides, we evaluate the performance of our proposed models (4.19) and (4.24), in order to see if the small value of $s$ is actually able to generate better generalization performance. Finally, to verify the quality of the proposed model (therefore the HS), we employ a series of data sets, and compare our model with several other widely used MT-MKL formulations.

## Evaluation of Monotonicity

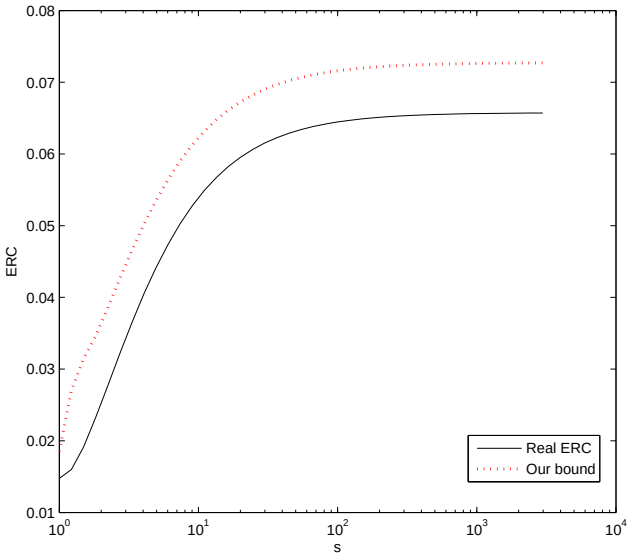
In this section, we conduct the first part of the experiments, which is aimed to verify the monotonicity of the ERC values of the proposed HSs, the corresponding ERC bounds, and the generalization performance of the proposed models. Besides, by observing the difference between the ERCs and the bounds, we can investigate the tightness of the bounds.

As analyzed in Chapter 4, the actual ERC values of $\mathcal{F}_s$ and $\mathcal{F}_{s,r}$ cannot be calculated, primarily due to the difficulty of computing the expectation over the Rademacher random variable $\boldsymbol{\sigma}$ (see

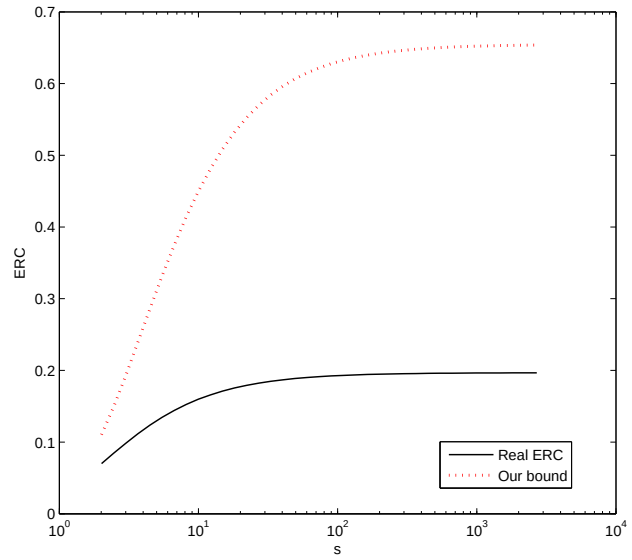
Equation (4.5) and Equation (4.8)). Instead, we approximate the expectation term by conducting Monte Carlo simulation, by drawing $D = 10^4$ samples of the $\boldsymbol{\sigma}$'s independently and identically from a uniform distribution on the hyper-cube $\{-1, 1\}^N$. Given a sample of $\boldsymbol{\sigma}$ and a pre-selected kernel function, $\hat{R}(\mathcal{F}_s)$ is calculated via Equation (4.5). On the other hand, the computation of $\hat{R}(\mathcal{F}_{s,r})$ based on the first equation of (4.8), is a little bit more sophisticated. This is because that, given a sample of $\boldsymbol{\sigma}$, a maximization problem is involved, which is even non-concave when $1 \leq s < 2$. Since there is no efficient algorithm that guarantees global maximum convergence for non-conave maximization problem, we omitted the case of $1 \leq s < 2$. Instead, for $\hat{R}(\mathcal{F}_{s,r})$, we only consider the case when $s \geq 2$, due to the pleasant concavity of the maximization problem, which is solved via CVX [38] [39].

To perform the experiment, a real-world data set is employed. This *Letter* data set is formulated with a collection of handwritten letters, where each letter is represented by $8 \times 16$ pixels. Therefore, each sample is formulated by a $128$ dimensional feature vector. Eight tasks are involved in this data set, where each task is a binary classification problem: 'C' versus 'E', 'G' versus 'Y', 'M' versus 'N', 'A' versus 'G', 'I' versus 'J', 'A' versus 'O', 'F' versus 'T' and 'H' versus 'N'. For each class, $100$ samples are used in this experiment. Gaussian kernel with spread parameter equals $2^7$, since this kernel gives the best classification performance on the corresponding model (4.19). For the MT-MKL HS, nine kernel functions are involved, including the Gaussian kernel function with spreads $\{2^{-7}, 2^{-5}, 2^{-3}, 2^{-1}, 2^0, 2^1, 2^3, 2^5, 2^7\}$. Finally, $R$ is set to $1$.

The evaluation of the ERC value and the corresponding ERC bound, in both single kernel MKL and MT-MKL cases, are shown in Figure 6.1. Several observations can be drawn from the figures. First, it is clear that, for both of the two HSs, the ERC and the corresponding ERC bound are monotonically increasing with respect to $s$. This result verified our theoretical analysis, which is given in Chapter 4. Second, for $\mathcal{F}_s$, it can be seen that the ERC bound, which is derived in Chapter 4, is very tight, since it matches the real ERC value with only a small gap for all $s \geq 1$.




(a) HS: $\mathcal{F}_s$



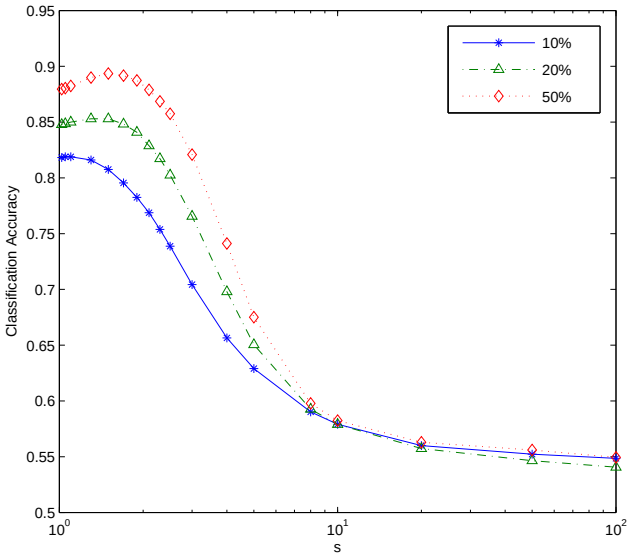

(b) HS: $\mathcal{F}_{s,r}$

Figure 6.1: Comparison between Monte Carlo-estimated ERCs and our derived bounds using the Letter data set. $10^4$ $\boldsymbol{\sigma}_t$ samples were used for Monte Carlo estimation. We sampled $100$ data for each letter and used $9$ kernel functions in multiple kernel scenario.

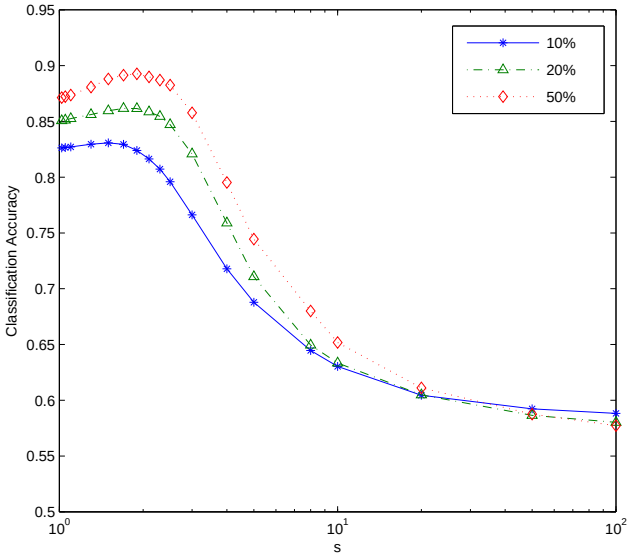
On the other hand, although the ERC bound for $\mathcal{F}_{s,r}$ is relatively loose, for most $s$, it is still very tight, when $s$ is small, a more preferred scenario. Therefore, this set of experiments demonstrated the good quality of our theoretical analysis in Chapter 4.

Next, we adapt the same data set to train the models that are proposed in Chapter 4, namely Problem (4.19) and Problem (4.24), in order to see the relation between the generalization performance and the value of $s$. Kernel selection for the single kernel model is performed via cross-validation. On the other hand, as a convention of MKL models, for the MT-MKL formulation, besides the nine Gaussian kernel functions, we added two other kernel functions, namely the linear kernel and the $2^{nd}$-order polynomial kernel. For the regularization parameter $C$, it is also chosen via cross-validation. Besides, the training set is randomly drawn from the original data set, with three different sizes, *i.e.*, $10\%$, $20\%$ and $50\%$ of the original data set are used for training, where the original data set contains $400$ samples for each class of each task. When $s > 2$, the MT-MKL problem is solved via the general purpose algorithm that we developed in Chapter 5, while the MT-MKL model with $1 \leq s \leq 2$ and the single kernel MTL can be easily solved by the block coordinate descent algorithm, as discussed in Chapter 4. Finally, for each value of $s$, we report the averaged classification accuracy, with $20$ runs on randomly drawn training set. The experimental results are reported in Figure 6.2.

It can be seen from Figure 6.2 that, clearly, the averaged classification accuracy generally monotonically decreases with respect to $s$, in both the single kernel MTL and the MT-MKL scenario. This is expected, because the ERC value in both cases increases with $s$, which potentially leads to an increasing generalization bound. Therefore, it is not a surprise that the deteriorated performance is achieved, when $s$ is large. On the other hand, it is of interest to see that, in many experiments, the best performance is not obtained when $s = 1$, but slightly larger than $s$. This phenomenon is explained as follows:




(a) Single kernel MTL model




(b) MT-MKL model

Figure 6.2: Experimental results on the Letter data set with different sizes of the training set. The training set consists with 10%, 20% and 50% of the original data set. Averaged classification accuracy over 20 runs on randomly drawn training samples are reported.

When $s = 1$, although the lowest ERC value is obtained, a much larger empirical error is achieved, compared to the empirical error that corresponds to the optimal $s$. As explained in Chapter 2, this is due to the small volume of the HS. As a result, the generalization bound for $s = 1$ has a larger value, which explains the slightly lower classification accuracy. This can be simply verified. For example, when $50\%$ of the data are used for training, the averaged empirical error over $20$ runs for the single kernel model is $0.0195$, when $s = 1$, which is much larger than the case of $s = 2$, where the empirical error is as small as $0.0025$.

### Comparison with Other Methods

In this section, we experimentally examine the performance of the proposed MT-MKL model (4.24), by comparing it with $6$ other popular MT-MKL models on a few real world application data sets. We omit the single-kernel model, since it is simply a special case of the MT-MKL formulation.

The first model that we compare with is the *Conic MTL* method [56].

$$\min_{\boldsymbol{w}\in\Omega(\boldsymbol{w})} \sum_{t=1}^{T} \lambda_t f(\boldsymbol{w}_t, \boldsymbol{x}_t, \boldsymbol{y}_t) \tag{6.1}$$

with $\lambda_t > 0, \forall t = 1, \cdots, T$. It features a conic combination of the $T$ task objective functions, instead of using the most commonly applied *Average MTL*, which averages the objective functions. The authors derived the generalization bound of the *Conic MTL* framework, and proposed the

following formulation, which selects the coefficients that minimizes the generalization bound:

$$\begin{aligned}\min_{\boldsymbol{w},\boldsymbol{b},\boldsymbol{\lambda},\boldsymbol{\theta}} \ & \sum_{t=1}^{T}\lambda_t\Big(\sum_{m=1}^{M}\frac{\|\boldsymbol{w}_t^m\|_{\mathcal{H}_m}^2}{2\theta_m}+C\sum_{i=1}^{N}\sum_{m=1}^{M}L_{hinge}(y_t^i(\langle\boldsymbol{w}_t^m,\phi_m(\boldsymbol{x}_t^i)\rangle_{\mathcal{H}_m}+b_t))\Big)\\ \text{s.t. } & \boldsymbol{\theta}\in\Omega(\boldsymbol{\theta}),\ \sum_{t=1}^{T}\frac{1}{\lambda_t}\leq a, \mathbf{1}\prec\boldsymbol{\lambda}\prec r_\lambda\mathbf{1}.\end{aligned} \tag{6.2}$$

In this experiment, we specify the feasible region of $\boldsymbol{\theta}$ to feature a $L_p$-norm MKL, *i.e.*, $\Omega(\boldsymbol{\theta}) = \{\boldsymbol{\theta}:\boldsymbol{\theta}\succeq\mathbf{0}, \|\boldsymbol{\theta}\|_p\leq 1\}$.

Another similar formulation, called *Pareto-Path MTL* [57], is also compared with our model. The spirit of Pareto-Path MTL is the same as Conic MTL, which is to minimize the non-averaged objective function of the $T$ tasks. Therefore, the model is designed to minimize the $L_p$-(pseudo) norm of the $T$ objectives:

$$\min_{\boldsymbol{w}\in\Omega(\boldsymbol{w})}\ (\sum_{t=1}^{T}f(\boldsymbol{w}_t,\boldsymbol{x}_t,\boldsymbol{y}_t)^p)^{\frac{1}{p}} \tag{6.3}$$

with $p>0$. It is proved that, by varying $p$, the solution of the above optimization problem features a series of points on the Pareto-Front of the corresponding Multi-Objective Optimization problem. In our experiment, similar to Problem (6.2), we specify $f$ to be the SVM objective with a $L_p$-norm MKL setting.

The third model that we compare with is the one that is proposed in [91]. Different to many commonly encountered MT-MKL models, this formulation does not force all tasks to share the same kernel function. Instead, the kernel function is comprised with both a common kernel across tasks and a task specific kernel. With such a setting, it is able to reduce the effect of "negative transfer", when outlier tasks exist, by letting them to be trained on the feature space that is not

shared with the other tasks. This model is formulated as follows:

$$\begin{aligned}
&\min_{\boldsymbol{\theta}} \max_{\boldsymbol{\alpha}} \sum_{t=1}^{T} (\boldsymbol{\alpha}_t{}'\mathbf{1} - \frac{1}{2}\boldsymbol{\alpha}_t{}'\boldsymbol{Y}_t(\sum_{m=1}^{M}\theta_t^m \boldsymbol{K}_t^m)\boldsymbol{Y}_t\boldsymbol{\alpha}_t) \\
&\quad \text{s.t. } \mathbf{0} \preceq \boldsymbol{\alpha}_t \preceq C\mathbf{1},\ \boldsymbol{\alpha}_t{}'\boldsymbol{y}_t = 0, t = 1, \cdots, T \\
&\qquad \theta_t^m = \xi_m + \gamma_t^m, \xi_m \geq 0, \gamma_t^m \geq 0, m = 1, \cdots, M, t = 1, \cdots, T \\
&\qquad \sum_{m=1}^{M} \theta_t^m = 1, \forall t = 1, \cdots, T, \sum_{m=1}^{M}\sum_{t=1}^{T} \gamma_t^m \leq \beta
\end{aligned} \tag{6.4}$$

The fourth method that we compare with is the so-called *Sparse MTL*, which is originally proposed in [79]:

$$\min_{\boldsymbol{w}} \sum_{m=1}^{M} (\sum_{t=1}^{T} \|\boldsymbol{w}_t^m\|_{\mathcal{H}_m}^q)^{p/q} + C\sum_{t=1}^{T}\sum_{i=1}^{N} L_{hinge}(y_t^i(\sum_{m=1}^{M}\langle \boldsymbol{w}_t^m, \phi_m(\boldsymbol{x}_t^i)\rangle_{\mathcal{H}_m} + b_t)) \tag{6.5}$$

with $0 < p \leq 1$ and $1 \leq q \leq 2$. By setting different values of $p$ and $q$, task level or kernel level sparsity can be achieved for the $\boldsymbol{w}_t^m$'s.

The fifth model involves the most straightforward and commonly encountered MT-MKL model with averaged objective function over the $T$ task objectives:

$$\begin{aligned}
&\min_{\boldsymbol{w},\boldsymbol{b},\boldsymbol{\theta}} \sum_{t=1}^{T} \frac{\|\boldsymbol{w}_t\|_{\mathcal{H}_\theta}^2}{2} + C\sum_{t=1}^{T}\sum_{i=1}^{N_t} L_{hinge}(y_t^i(\langle \boldsymbol{w}_t, \phi(\boldsymbol{x}_t^i)\rangle_{\mathcal{H}_\theta} + b_t)) \\
&\quad \text{s.t. } \boldsymbol{w}_t \in \mathcal{H}_{\boldsymbol{\theta}}, t = 1, \cdots, T \\
&\qquad \boldsymbol{\theta} \in \Omega(\boldsymbol{\theta})
\end{aligned} \tag{6.6}$$

Again, the $L_p$-norm MKL setting is employed, to be consistent with the other formulations that we compare with.

The last model serves as a baseline approach, where each task is trained independently via a single-task MKL model, as shown below:

$$\min_{\boldsymbol{w},b,\boldsymbol{\theta}\in\Omega(\boldsymbol{\theta})} \frac{\|\boldsymbol{w}\|^2_{\mathcal{H}_{\boldsymbol{\theta}}}}{2} + C\sum_{i=1}^{N} L_{hinge}(y_i(\langle \boldsymbol{w}, \phi(\boldsymbol{x}_i)\rangle_{\mathcal{H}_{\boldsymbol{\theta}}} + b)) \tag{6.7}$$

The above formulation is trained and tested on each task independently. Since any MT-MKL model should take advantage of the existence of multiple tasks, it is expected that the baseline should obtain the worst generalization performance in the most experiments.

Besides the Letter data set that is employed for the experiments in the last section, $5$ more real world application data sets are used for this set of comparison.

The *MNIST*[1] and *USPS*[2] are two widely used handwritten digit recognition data sets. Each one contains a collection of handwritten digit images, from "0" to "9". For these two data sets, each digit is represented by a series pixel values, which are concatenated into a feature vector. Therefore, for the MNIST data set, each sample is a $784$ dimensional feature vector, while the samples from the USPS data set are $256$ dimensional. As $10$-class classification problems, each data set is manually converted to $45$ binary classification tasks via a one-versus-one strategy.

The Wall-Following Robot Navigation data set, or *Robot* in short, is collected by University of California in Irvine (UCI) Machine Learning Repository [33]. It is aimed to determine the move of the robot, given its location. Four classes are involved, namely "Move-Forward", "Slight-Right-Turn", "Sharp-Right-Turn" and "Slight-Left-Turn". Each entry is consisted with $4$ features, which describe the position of the robot. In order to adapt this multi-class data set in our MTL, setting, a one-versus-one strategy is employed to cast the data set to $6$ binary classification tasks.

[1]Available at: `http://yann.lecun.com/exdb/mnist/`

[2]Available at: `http://www.cs.nyu.edu/~roweis/data.html`

The *Vehicle* data set, which is also collected by the UCI Machine Learning Repository, contains a collection of vehicle images. Each image, after performing feature extraction, is represented by a $18$ dimensional feature vector. The objective of this data set is to correctly classify each vehicle image to one of the four classes, namely "4 Opel", "SAAB", "Bus" and "Van". Similar to MNIST and USPS, this $4$-class classification problem is also converted to a MTL problem, by the one-versus-one technique.

Finally, the *Landmine*[3] data set contains $29$ binary classification tasks. Each task is aimed to detect landmines in a specific region, given radar images. Specifically, tasks $1-15$ corresponds to regions that are relatively highly foliated, and the other $14$ tasks correspond to bare earth or desert regions. Each radar image, after performing feature extraction, is represented by a $9$ dimensional feature vector. The characteristics of each data set is summarized in Table 6.1.

Table 6.1: Characteristics of the Data Sets

| Data Set | # Attributes | # Tasks | # Samples for Each Task |
|---|---|---|---|
| Letter | 128 | 8 | 397 |
| MNIST | 784 | 45 | 200 |
| USPS | 256 | 45 | 200 |
| Robot | 4 | 6 | 500 |
| Vehicle | 18 | 6 | 423 |
| Landmine | 9 | 29 | 62 |

Note that for each data set, the number of samples for each task that is reported in Table 6.1 is an averaged number. For the Letter, Vehicle and Landmine data sets, the number of samples for each task varies slightly.

In order to thoroughly examine the performance of each model with variable sizes of training set,

[3]Available at: `http://people.ee.duke.edu/~lcarin/LandmineData.zip`

similar to the previous experiments, $10\%$, $20\%$ and $50\%$ of the original data sets are used for training. One exception is the Landmine data set; one of the $29$ tasks contains only 30 samples, which is too few, if only $10\%$ of them are used for training. Therefore, for the case of $10\%$ training data, we did not experiment on the Landmine data set.

The averaged classification accuracy, with $20$ runs on randomly drawn training set, is recorded and reported. All hyper-parameters are selected via the cross-validation method. When $s > 2$, our model is solved by the general purpose algorithm that is proposed in Chapter 5, and when $1 \leq s \leq 2$, it is optimized via the block coordinate descent algorithm, as discussed in Chapter 4. The experimental results are reported in Table 6.2 - Table 6.4.

In order to test the statistical significance of the differences between the results of our method and that of the other six methods, we utilized the $t$-test to compare the mean accuracies with a significant level of $\alpha = 0.05$. In each of the three tables, the underlined number represents a statistical significant worse result, compared to that of our method. Besides, in the last row of each table, we report the $s$ value that gives the highest averaged classification accuracy for our model.

Table 6.2: Experimental Results with $10\%$ Data for Training

| | Letter | MNIST | USPS | Robot | Vehicle |
|---|---|---|---|---|---|
| Conic MTL | 83.00 | 93.59 | 94.61 | <u>95.83</u> | <u>80.10</u> |
| Pareto-Path MTL | 83.95 | <u>89.30</u> | <u>90.22</u> | <u>95.07</u> | <u>80.05</u> |
| Tang's Method | <u>80.86</u> | <u>92.37</u> | <u>93.20</u> | <u>93.93</u> | <u>78.47</u> |
| Sparse MTL | 83.00 | 93.48 | 94.52 | <u>94.69</u> | <u>79.28</u> |
| Average MTL | <u>81.45</u> | <u>88.80</u> | <u>90.11</u> | <u>95.16</u> | <u>79.77</u> |
| Baseline Method | <u>81.33</u> | <u>88.71</u> | <u>89.02</u> | <u>95.54</u> | <u>78.01</u> |
| Our Method | 83.50 | 93.33 | 94.37 | 96.11 | 82.23 |
| Optimal $s$ | 1.5 | 1.5 | 1.3 | 2.0 | 2.0 |

Table 6.3: Experimental Results with $20\%$ Data for Training

| | Letter | MNIST | USPS | Robot | Vehicle | Landmine |
|---|---|---|---|---|---|---|
| Conic MTL | 87.13 | 96.08 | 97.44 | 97.11 | 84.69 | 70.18 |
| Pareto-Path MTL | 87.51 | 95.02 | 96.26 | 96.11 | 85.33 | 69.59 |
| Tang's Method | 82.95 | 95.94 | 97.37 | 96.36 | 83.98 | 66.60 |
| Sparse MTL | 87.09 | 95.96 | 97.53 | 96.56 | 84.44 | 68.89 |
| Average MTL | 86.42 | 94.95 | 96.25 | 95.90 | 85.22 | 67.24 |
| Baseline Method | 86.39 | 94.81 | 96.17 | 95.75 | 84.37 | 66.64 |
| Our Method | 86.57 | 95.74 | 97.39 | 97.08 | 86.89 | 72.58 |
| Optimal $s$ | 1.7 | 1.7 | 1.7 | 2.0 | 2.0 | 1.0 |

Table 6.4: Experimental Results with $50\%$ Data for Training

| | Letter | MNIST | USPS | Robot | Vehicle | Landmine |
|---|---|---|---|---|---|---|
| Conic MTL | 90.47 | 97.44 | 98.98 | 98.41 | 89.90 | 78.26 |
| Pareto-Path MTL | 90.61 | 96.92 | 98.51 | 96.80 | 88.04 | 77.42 |
| Tang's Method | 84.87 | 97.47 | 98.96 | 97.21 | 88.13 | 76.08 |
| Sparse MTL | 90.65 | 97.53 | 98.98 | 98.09 | 88.57 | 75.82 |
| Average MTL | 90.01 | 96.98 | 98.59 | 96.59 | 87.93 | 76.96 |
| Baseline Method | 89.80 | 97.04 | 98.49 | 96.31 | 87.64 | 76.29 |
| Our Method | 89.54 | 97.33 | 98.89 | 98.25 | 90.60 | 77.99 |
| Optimal $s$ | 1.7 | 1.9 | 1.9 | 1.5 | 2.0 | 2.0 |

Several observations can be drawn from these three tables. First, it can be seen that our proposed method outperforms the other models in the most experiments. In fact, the advantageous is even statistically significant, in many scenarios. Such a result verified the benefit of our model and the HSs, which are introduced in Chapter 4.

Secondly, it is worth to point out that compared to the Conic MTL and Pareto-Path MTL methods, our method is very competitive, and in some experiments, even a statistically significantly better result is obtained by our model. This is an unexpected result, because both of these two formulations are based on non-averaged objective functions, which search a large region of the Pareto-Front of the corresponding Multi-Objective Optimization problems. On the other hand, our method minimizes the average of the $T$ empirical losses (see 4.12), which leads to a specific point on the Pareto-Front. This surprising result reflects the good design of our proposed HS. It is also expected that, with an improvement on the design of the HSs for Conic MTL and Pareto-Path MTL, better generalization performance may be obtained by these two models.

Thirdly, it is of interest to see that, although the six MTL methods are advantages in most cases, compared to the baseline model, their advantages become less and less obvious, when the number of training data increases. This phenomenon not only matches theoretical analysis, but also is intuitively straightforward. Theoretically, the last term of the MTL generalization bound (2.62) decreases with respect to $T$ and $N$ in an order of $O(\sqrt{\frac{1}{TN}})$. Assuming the ERC term also decreases with $T$ and $N$, a common property for most MTL models, we immediately see that, when $N$ is very large, the effect of $T$ in decreasing the generalization bound is less obvious, compared to when $N$ is relatively small. Therefore, it is not a surprise that the MTL models are less beneficial, when the number of training samples is large. This is also intuitively straightforward: Each task can be independently trained well enough, as long as there is a sufficient amount of data for training. Therefore, in this case, if trained with a MTL method, there is very limited improvement can be done, even though there is shared information provided by the other tasks.

Finally, as analyzed in Chapter 4, for our method, a small value of $s$ should provide high classification accuracy. This statement is not only verified in the previous section on the Letter data set, but also in Table 6.2 - Table 6.4: We see that the optimal $s$ is always around the value of $1.5$. Such a consistent result convincingly demonstrated the power and value of theoretical analysis in

helping practical application, specifically in terms of model selection: Instead of having to conduct cross-validation with respect to $s$ over the range of $[1, +\infty)$, it is confidently safe to set $s$ around $1.5$, or, at least shrink the range of cross-validation to as small as $[1, 2]$.

# CHAPTER 7: CONCLUSION

MTL is an inductive transfer machine learning paradigm that seeks improvements of generalization performance for each task, by intentionally building the model in an information sharing manner. It is hoped that the training of each task can be benefited from the information that is shared by the other tasks. Previously, one of the most widely utilized information sharing approach is to map the data from all tasks to a (partially) shared common feature space. Any kernel-based MTL formulation with (partially) common kernel function, and, especially the MT-MKL method, falls into this category.

Although an important class of MTL methods, its theoretical studies, in terms of the research of its generalization bound, has been largely limited to a small range of HSs. The HSs that are considered in the previous two works did not include kernel-based MTL and MT-MKL. Besides, they both unnecessarily set all the task weights to be constrained in norm balls, whose radii are the same. This inflexibility in the radius size directly lead to poor performance of the corresponding MTL models, as have been shown earlier.

Besides, in the practical perspective of MT-MKL, one problem that exists in previous research is that, it is usually difficult to develop practical algorithm to solve the corresponding MT-MKL models. Such an algorithmic difficulty not only forces researchers to develop model-specific algorithms, but also requires users to interpret and implement several difficult algorithms, when model comparison and selection is needed.

This dissertation largely addresses the above two issues for kernel-based MTL and MT-MKL. Theoretically, we proposed two HSs, one for single-kernel MTL, and another one for MT-MKL. The same kernel function is utilized for all tasks, so that the data from all tasks are mapped to a shared feature space. Besides, each task weight is constrained in a norm-ball, whose radius is

learned during the training phase. After theoretical analysis of the ERCs of these two HSs, we showed their monotonicity with respect to the parameter $s$. This important result serves as a guide for model selection: A small value of $s$ is preferred, since it produces a small ERC value, and thus, a potentially small generalization bound, which leads to better generalization performance. As a byproduct, we demonstrated that the equal radii HS, which is introduced in previous research, is only the worst special case of our HSs, since it gives the largest ERC value, and may produces deteriorated generalization performance.

By deriving the upper bound of the ERC terms, several useful insights regarding the HSs can be clearly observed. Specifically, for the single-kernel MTL HS, we have shown that, in the worst case, when $s \to +\infty$, the ERC bound does not decrease with $T$, which reveals the fact that, such a setting does not take advantage of the existence of multiple tasks and is similar to independent learning of each task. On the other hand, in the best scenario, when $s$ is close to $1$, the optimal $O(\frac{\sqrt{\log T}}{T})$ bound is achieved, which decreases with $T$ in a fast speed. Similar conclusion is drawn from the ERC bound of the MT-MKL HS. Moreover, it is shown that the proposed MT-MKL HS not only preserves the optimal $O(\sqrt{\log M})$ bound encountered in single-task MKL, but also preserves the optimal $O(\frac{\sqrt{\log T}}{T})$ bound encountered in the single-kernel MTL case. This fact indicates a good design of the HS.

Inspired by the appealing properties of the HSs, corresponding models are proposed. Experiments based on the models verified the above arguments: Indeed, the best performance is always achieved when $s$ is of small value, and when $s$ increases, the generalization performance deteriorates. A series of comparison with other popular MT-MKL models further demonstrated the strength of our models and HSs: Our model outperforms the other ones in most experiments, and the advantage is usually statistically significant.

In order to tackle the algorithm issue for MT-MKL, we first propose a general framework for

kernel-based MTL models in the dual domain, which features a min-max optimization problem. We show that, the framework not only covers our proposed MT-MKL model, but also embraces a large amount of existing MT-MKL models as special cases. Besides, when only one task is considered, the framework degrades to a general MKL formulation, which also covers many existing single-task MKL models. Then, we developed a general purpose algorithm to solve the framework, which therefore solves all the models that are special cases of the framework. We demonstrated that the algorithm is simple and easy to implement. In each iteration, it contains only two steps, each of which are usually easy to solve, either by existing efficient software packages, such as `LIBSVM`, or by deriving and utilizing closed-form solutions.

# APPENDIX: PROOFS OF THEORETICAL RESULTS

Proof to Theorem 2

Let $\boldsymbol{z}_i \triangleq (\boldsymbol{x}_i, y_i)$, and $g(\boldsymbol{z}_i) \triangleq L(y_i f(\boldsymbol{x}_i))$, based on Theorem 1, we immediately have

$$E\{L(Yf(X))\} \leq \frac{1}{N}\sum_{i=1}^{N} L(y_i f(\boldsymbol{x}_i)) + \hat{R}(L \circ \mathcal{F}) + \sqrt{\frac{9\ln\frac{1}{\delta}}{2N}} \tag{1}$$

Then, (2.9) is proved by noticing that $L$ upper-bound the $0/1$ loss function and using Lemma 1. ((2.9)) can be similarly proved.

Proof to Theorem 4

We first define the margin loss as below

$$L_{margin}(x) = \begin{cases} 0 & \text{if } \gamma \leq x \\ 1 - x/\gamma & \text{if } 0 \leq x \leq \gamma \\ 1 & \text{if } x \leq 0 \end{cases} \tag{2}$$

By applying Theorem 2, we get

$$er(f) \leq \hat{er}_{L_{margin}}(f) + \frac{1}{\gamma}\hat{R}(\mathcal{F}) + \sqrt{\frac{9\ln\frac{1}{\delta}}{2N}} \tag{3}$$

Noticing that the hinge loss upper-bound the margin loss when $\gamma = 1$, we proved the first inequality

of (2.28). To prove the second inequality, we upper bound the ERC $\hat{R}(\mathcal{F})$:

$$\begin{aligned}\hat{R}(\mathcal{F}) &= \frac{2}{N} E_{\boldsymbol{\sigma}} \left\{ \sup_{\|\boldsymbol{w}\|_{\mathcal{H}} \le r, b \in \mathbb{R}} \sum_{i=1}^{N} \sigma_i (\langle \boldsymbol{w}, \phi(\boldsymbol{x}_i) \rangle_{\mathcal{H}} + b) \right\} \\ &= \frac{2}{N} E_{\boldsymbol{\sigma}} \left\{ \sup_{\|\boldsymbol{w}\|_{\mathcal{H}} \le r} \sum_{i=1}^{N} \sigma_i \langle \boldsymbol{w}, \phi(\boldsymbol{x}_i) \rangle_{\mathcal{H}} \right\} + \frac{2}{N} E_{\boldsymbol{\sigma}} \left\{ \sup_{b \in \mathbb{R}} \sum_{i=1}^{N} \sigma_i b \right\}\end{aligned} \tag{4}$$

Note that $\sup_{b \in \mathbb{R}} \sum_{i=1}^{N} \sigma_i b = (\sum_{i=1}^{N} \sigma_i) \sup_{b \in \mathbb{R}} b$, we have that

$$E_{\boldsymbol{\sigma}} \left\{ \sup_{b \in \mathbb{R}} \sum_{i=1}^{N} \sigma_i b \right\} = E_{\boldsymbol{\sigma}} \left\{ (\sum_{i=1}^{N} \sigma_i) \sup_{b \in \mathbb{R}} b \right\} = E_{\boldsymbol{\sigma}} \left\{ \sum_{i=1}^{N} \sigma_i \right\} \sup_{b \in \mathbb{R}} b = 0 \tag{5}$$

since $E\{\sigma_i\} = 0$. Therefore,

$$\begin{aligned}\hat{R}(\mathcal{F}) &= \frac{2}{N} E_{\boldsymbol{\sigma}} \left\{ \sup_{\|\boldsymbol{w}\|_{\mathcal{H}} \le r} \sum_{i=1}^{N} \sigma_i \langle \boldsymbol{w}, \phi(\boldsymbol{x}_i) \rangle_{\mathcal{H}} \right\} \\ &\le \frac{2}{N} E_{\boldsymbol{\sigma}} \left\{ \sup_{\|\boldsymbol{w}\|_{\mathcal{H}} \le r} \|\boldsymbol{w}\|_{\mathcal{H}} \, \| \sum_{i=1}^{N} \sigma_i \phi(\boldsymbol{x}_i) \|_{\mathcal{H}} \right\} \\ &= \frac{2r}{N} E_{\boldsymbol{\sigma}} \left\{ \| \sum_{i=1}^{N} \sigma_i \phi(\boldsymbol{x}_i) \|_{\mathcal{H}} \right\} \\ &= \frac{2r}{N} E_{\boldsymbol{\sigma}} \left\{ \sqrt{\sum_{i=1}^{N} \sum_{j=1}^{N} \sigma_i \sigma_j k(\boldsymbol{x}_i, \boldsymbol{x}_j)} \right\} \\ &\le \frac{2r}{N} \sqrt{E_{\boldsymbol{\sigma}} \left\{ \sum_{i=1}^{N} \sum_{j=1}^{N} \sigma_i \sigma_j k(\boldsymbol{x}_i, \boldsymbol{x}_j) \right\}} \\ &= \frac{2r}{N} \sqrt{\sum_{i=1}^{N} k(\boldsymbol{x}_i, \boldsymbol{x}_i)} \\ &= \frac{2r}{N} \sqrt{tr(\boldsymbol{K})}\end{aligned} \tag{6}$$

where the first inequality is based on Cauchy-Schwarz inequality, and the second inequality is application of Jensen's inequality.

Proof to Lemma 2

First notice that $\mathcal{F}_s$ is equivalent to the following HS:

$$\begin{aligned}\mathcal{F}_s \triangleq &\{\boldsymbol{x} \mapsto (\lambda_1 \langle \boldsymbol{w}_1, \phi(\boldsymbol{x})\rangle, \cdots, \lambda_T \langle \boldsymbol{w}_T, \phi(\boldsymbol{x})\rangle)' : \\ &\|\boldsymbol{w}_t\|^2 \leq R, \boldsymbol{\lambda} \in \Omega_s(\boldsymbol{\lambda})\}\end{aligned} \tag{7}$$

According to the same reasoning of Equations (1) and (2) in [26], we know that $\boldsymbol{w}_t = \sum_{i=1}^{N} \alpha_t^i \phi(\boldsymbol{x}_t^i)$, and the constraint $\|\boldsymbol{w}_t\|^2 \leq R$ is equivalent to $\boldsymbol{\alpha}_t^{'} \boldsymbol{K}_t \boldsymbol{\alpha}_t \leq R$. Therefore, based on the definition of ERC that is given in Equation (2.6), we have that

$$\hat{R}(\mathcal{F}_s) = \frac{2}{TN} E_\sigma \{ \sup_{\boldsymbol{\alpha}_t \in \mathcal{F}_s} \sum_{t=1}^{T} \lambda_t \boldsymbol{\sigma}_t^{'} \boldsymbol{K}_t \boldsymbol{\alpha}_t \} \tag{8}$$

where $\mathcal{F}_s = \{\boldsymbol{\alpha}_t \mid \boldsymbol{\alpha}_t^{'} \boldsymbol{K}_t \boldsymbol{\alpha}_t \leq \lambda_t^2 R, \forall t; \boldsymbol{\lambda} \in \Omega_s(\boldsymbol{\lambda})\}$. To solve the maximization problem with respect to $\boldsymbol{\alpha}_t$, we observe that the $T$ problems are independent and thus can be solved individually. Based on Cauchy-Schwartz inequality, the optimal $\boldsymbol{\alpha}_t$ is achieved when $\boldsymbol{K}_t^{\frac{1}{2}} \boldsymbol{\alpha}_t = c_t \boldsymbol{K}_t^{\frac{1}{2}} \boldsymbol{\sigma}_t$ where $c_t$ is a constant.

Substituting this result into each of the $T$ maximization problems, we have the following:

$$\begin{aligned}&\max_{c_t} c_t \boldsymbol{\sigma}_t^{'} \boldsymbol{K}_t \boldsymbol{\sigma}_t \\ &\text{s.t. } c_t^2 \boldsymbol{\sigma}_t^{'} \boldsymbol{K}_t \boldsymbol{\sigma}_t \leq R\end{aligned} \tag{9}$$

Obviously, the optimal $c_t$ is obtained when $c_t = \sqrt{\frac{R}{\boldsymbol{\sigma}_t^{'}\boldsymbol{K}_t\boldsymbol{\sigma}_t}}$. Therefore the ERC becomes now

$$\hat{R}(\mathcal{F}_s) = \frac{2}{TN} E_\sigma\{ \sup_{\boldsymbol{\lambda}\in\Omega_s(\boldsymbol{\lambda})} \sum_{t=1}^{T} \lambda_t \sqrt{\boldsymbol{\sigma}_t^{'}\boldsymbol{K}_t\boldsymbol{\sigma}_t R}\} \tag{10}$$

Since $s \geq 1$, based on Theorem 15, it is not difficult to get the solution of the maximization problem with respect to $\boldsymbol{\lambda}$, which gives

$$\begin{aligned} \hat{R}(\mathcal{F}_s) &= \frac{2\sqrt{R}}{TN} E_\sigma\{[\sum_{t=1}^{T} (\boldsymbol{\sigma}_t^{'}\boldsymbol{K}_t\boldsymbol{\sigma}_t)^{\frac{s^*}{2}}]^{\frac{1}{s^*}}\} \\ &= \frac{2\sqrt{R}}{TN} E_\sigma\{\|\boldsymbol{u}\|_{s^*}\} \end{aligned} \tag{11}$$

Proof to Theorem 8

First note that $\forall s_1 > s_2 \geq 1$, we have that $1 \leq s_1^* < s_2^*$, which means $\|\boldsymbol{u}\|_{s_1^*} \geq \|\boldsymbol{u}\|_{s_2^*}$. Based on Equation (11), we immediately have $\hat{R}(\mathcal{F}_{s_1}) \geq \hat{R}(\mathcal{F}_{s_2})$. This gives the monotonicity of $\hat{R}(\mathcal{F}_s)$ with respect to $s$.

Proof to Theorem 9

Similar to the proof to Lemma 2, we write the ERC of $\tilde{\mathcal{F}}$:

$$\hat{R}(\tilde{\mathcal{F}}) = \frac{2}{TN} E_\sigma\{ \sup_{\boldsymbol{\alpha}_t\in\tilde{\mathcal{F}}} \sum_{t=1}^{T} \boldsymbol{\sigma}_t^{'}\boldsymbol{K}_t\boldsymbol{\alpha}_t\} \tag{12}$$

Optimize with respect to $\boldsymbol{\alpha}_t$ gives

$$\hat{R}(\tilde{\mathcal{F}}) = \frac{2\sqrt{R}}{TN} E_\sigma \{ \sum_{t=1}^{T} \sqrt{\boldsymbol{\sigma}_t' \boldsymbol{K}_t \boldsymbol{\sigma}_t} \} \tag{13}$$

Based on Lemma 2, we immediately obtain $\hat{R}(\tilde{\mathcal{F}}) = \hat{R}(\mathcal{F}_{+\infty})$.

Proof to Theorem 10

According to Equation (11) and Jensen's Inequality, we have

$$\begin{aligned} \hat{R}(\mathcal{F}_s) &\leq \frac{2\sqrt{R}}{TN} (\sum_{t=1}^{T} E_\sigma \{ (\boldsymbol{\sigma}_t' \boldsymbol{K}_t \boldsymbol{\sigma}_t)^{\frac{s^*}{2}} \})^{\frac{1}{s^*}} \\ &= \frac{2\sqrt{R}}{TN} (\sum_{t=1}^{T} E_\sigma \{ \| \sum_{i=1}^{N} \sigma_t^i \phi(\boldsymbol{x}_t^i) \|_2^{s^*} \})^{\frac{1}{s^*}} \end{aligned} \tag{14}$$

Based on Theorem 6, we have that

$$\begin{aligned} \hat{R}(\mathcal{F}_s) &\leq \frac{2\sqrt{R}}{TN} (\sum_{t=1}^{T} (s^* \text{tr}(\boldsymbol{K}_t))^{\frac{s^*}{2}})^{\frac{1}{s^*}} \\ &= \frac{2}{TN} \sqrt{R s^* \| \text{tr}(\boldsymbol{K}_t)_{t=1}^{T} \|_{\frac{s^*}{2}}} \end{aligned} \tag{15}$$

where $\| \text{tr}(\boldsymbol{K}_t)_{t=1}^{T} \|_{\frac{s^*}{2}}$ denotes the $l_{\frac{s^*}{2}}$-norm of vector $[\text{tr}(\boldsymbol{K}_1), \cdots, \text{tr}(\boldsymbol{K}_T)]'$. Since we assumed

that $k(\boldsymbol{x}, \boldsymbol{x}) \leq 1, \forall \boldsymbol{x}$, we have

$$\begin{aligned} \hat{R}(\mathcal{F}_s) &\leq \frac{2}{TN}\sqrt{Rs^*(\sum_{t=1}^{T} N^{\frac{s^*}{2}})^{\frac{2}{s^*}}} \\ &= \frac{2}{T\sqrt{N}}\sqrt{RT^{\frac{2}{s^*}}s^*} \end{aligned} \tag{16}$$

Note that this bound can be further improved for the interval $s \in [1, \rho^*]$. To make this improvement, we first prove that $\hat{R}(\mathcal{F}_s) \leq T^{\frac{1}{s'}-\frac{1}{s}}\hat{R}(\mathcal{F}_{s'})$ for any $s' \geq s \geq 1$.

$$\begin{aligned} \hat{R}(\mathcal{F}_s) &= \frac{2}{TN}E_\sigma\{\sup_{\boldsymbol{\lambda}\succeq\mathbf{0}, \|\boldsymbol{\lambda}\|_s \leq 1} \sum_{t=1}^{T} \lambda_t \sqrt{\boldsymbol{\sigma}_t' \boldsymbol{K}_t \boldsymbol{\sigma}_t R}\} \\ &\leq \frac{2}{TN}E_\sigma\{\sup_{\boldsymbol{\lambda}\succeq\mathbf{0}, \|\boldsymbol{\lambda}\|_{s'} \leq T^{\frac{1}{s'}-\frac{1}{s}}} \sum_{t=1}^{T} \lambda_t \sqrt{\boldsymbol{\sigma}_t' \boldsymbol{K}_t \boldsymbol{\sigma}_t R}\} \\ &= \frac{2}{TN}E_\sigma\{\sup_{\boldsymbol{\lambda}\succeq\mathbf{0}, \|\boldsymbol{\lambda}\|_{s'} \leq 1} \sum_{t=1}^{T} T^{\frac{1}{s'}-\frac{1}{s}} \lambda_t \sqrt{\boldsymbol{\sigma}_t' \boldsymbol{K}_t \boldsymbol{\sigma}_t R}\} \\ &= T^{\frac{1}{s'}-\frac{1}{s}}\hat{R}(\mathcal{F}_{s'}) \end{aligned} \tag{17}$$

Based on this conclusion, we have that $\forall s \in [1, \rho^*]$,

$$\begin{aligned}
\hat{R}(\mathcal{F}_s) &\leq T^{\frac{1}{\rho^*}-\frac{1}{s}} \hat{R}(\mathcal{F}_{\rho^*}) \\
&= T^{\frac{1}{\rho^*}-\frac{1}{s}} \frac{2}{TN} \sqrt{2eRN \log T} \\
&= T^{\frac{1}{\rho^*}-1+1-\frac{1}{s}} \frac{2}{TN} \sqrt{2eRN \log T} \\
&= \frac{T^{\frac{1}{s^*}}}{T^{\frac{1}{\rho}}} \frac{2}{TN} \sqrt{2eRN \log T} \\
&= \frac{T^{\frac{1}{s^*}}}{\sqrt{e}} \frac{2}{TN} \sqrt{2eRN \log T} \\
&= \frac{2}{T\sqrt{N}} \sqrt{RT^{\frac{2}{s^*}} \rho}
\end{aligned} \tag{18}$$

Note that this is always less than $\frac{2}{T\sqrt{N}} \sqrt{RT^{\frac{2}{s^*}} s^*}$ that is given in (16); $\rho^*$ is the global minimizer of the expression in (16) as a function of $s$. In summary, we have $\hat{R}(\mathcal{F}_s) \leq \frac{2}{T\sqrt{N}} \sqrt{RT^{\frac{2}{s^*}} \rho}$ when $s \in [1, \rho^*]$, and $\hat{R}(\mathcal{F}_s) \leq \frac{2}{T\sqrt{N}} \sqrt{RT^{\frac{2}{s^*}} s^*}$ when $s > \rho^*$.

## Proof to Lemma 3

Define $\boldsymbol{K}_t \triangleq \sum_{m=1}^{M} \theta_m \boldsymbol{K}_t^m, \forall t$, then we can write

$$\hat{R}(\mathcal{F}_{s,r}) = \frac{2}{TN} E_\sigma \{ \sup_{\boldsymbol{\alpha}_t \in \mathcal{F}_{s,r}} \sum_{t=1}^{T} \lambda_t \boldsymbol{\sigma}_t^{'} \boldsymbol{K}_t \boldsymbol{\alpha}_t \} \tag{19}$$

where $\mathcal{F}_s = \{\boldsymbol{\alpha}_t \mid \boldsymbol{\alpha}_t^{'} \boldsymbol{K}_t \boldsymbol{\alpha}_t \leq \lambda_t^2 R, \forall t; \boldsymbol{\lambda} \in \Omega_s(\boldsymbol{\lambda}); \boldsymbol{\theta} \in \Omega_r(\boldsymbol{\theta})\}$. Then using the similar proof of

Lemma 2, we have that

$$\begin{aligned}\hat{R}(\mathcal{F}_{s,r}) &= \frac{2\sqrt{R}}{TN} E_\sigma\{ \sup_{\boldsymbol{\theta}\in\Omega_r(\boldsymbol{\theta})} [\sum_{t=1}^{T} (\boldsymbol{\sigma}_t' \boldsymbol{K}_t \boldsymbol{\sigma}_t)^{\frac{s^*}{2}}]^{\frac{1}{s^*}}\} \\ &= \frac{2\sqrt{R}}{TN} E_\sigma\{ \sup_{\boldsymbol{\theta}\in\Omega_r(\boldsymbol{\theta})} \sum_{t=1}^{T} (\boldsymbol{\theta}' \boldsymbol{u}_t)^{\frac{s^*}{2}}\}^{\frac{1}{s^*}}\end{aligned} \tag{20}$$

This gives the first equation in Equation (4.8). To prove the second equation, we simply optimize (20) with respect to $\boldsymbol{\theta}$, which directly gives the result.

## Proof to Theorem 11

Consider Equation (4.8) and let $g(\boldsymbol{\lambda}) \triangleq \sup_{\boldsymbol{\alpha}\in\Omega(\boldsymbol{\alpha})} \|\boldsymbol{v}\|_{r^*}$. Then

$$\hat{R}(\mathcal{F}_{s,r}) = \frac{2}{TN} E_\sigma\{ \sup_{\boldsymbol{\lambda}\in\Omega_s(\boldsymbol{\lambda})} g(\boldsymbol{\lambda})\} \tag{21}$$

Note that $\forall 1 \leq s_1 < s_2$, we have the relation $\Omega_{s_1}(\boldsymbol{\lambda}) \subseteq \Omega_{s_2}(\boldsymbol{\lambda})$. Therefore, let $\hat{\boldsymbol{\lambda}}_1$, $\hat{\boldsymbol{\lambda}}_2$ be the solution of problems $\sup_{\boldsymbol{\lambda}\in\Omega_{s_1}(\boldsymbol{\lambda})} g(\boldsymbol{\lambda})$ and $\sup_{\boldsymbol{\lambda}\in\Omega_{s_2}(\boldsymbol{\lambda})} g(\boldsymbol{\lambda})$ correspondingly, we must have $g(\hat{\boldsymbol{\lambda}}_1) \leq g(\hat{\boldsymbol{\lambda}}_2)$. This directly implies $\hat{R}(\mathcal{F}_{s_1,r}) \leq \hat{R}(\mathcal{F}_{s_2,r})$.

## Proof to Theorem 12

Define $\boldsymbol{K}_t \triangleq \sum_{m=1}^{M} \theta_m \boldsymbol{K}_t^m, \forall t$, then

$$\hat{R}(\tilde{\mathcal{F}}_r) = \frac{2}{TN} E_\sigma\{ \sup_{\boldsymbol{\alpha}_t\in\tilde{\mathcal{F}}} \sum_{t=1}^{T} \boldsymbol{\sigma}_t' \boldsymbol{K}_t \boldsymbol{\alpha}_t\} \tag{22}$$

Fix $\boldsymbol{\theta}$ and optimize with respect to $\boldsymbol{\alpha}_t$ gives

$$\hat{R}(\tilde{\mathcal{F}}_r) = \frac{2}{TN}\sqrt{R}E_\sigma\{\sup_{\boldsymbol{\theta}\in\Omega_r(\boldsymbol{\theta})}\sum_{t=1}^{T}\sqrt{\boldsymbol{\theta}'\boldsymbol{u}_t}\} \tag{23}$$

Based on Equation (4.8), we immediately obtain $\hat{R}(\tilde{\mathcal{F}}_r) = \hat{R}(\mathcal{F}_{+\infty,r})$.

Proof to Theorem 13

Based on Equation (4.8) and Hölder's Inequality, let $c \triangleq \max\{0, \frac{1}{r^*} - \frac{2}{s^*}\}$ we have that

$$\begin{aligned}
\hat{R}(\mathcal{F}_{s,r}) &\leq \frac{2}{TN}\sqrt{R}E_\sigma\{\sup_{\boldsymbol{\theta}\in\Omega_r(\boldsymbol{\theta})}\sum_{t=1}^{T}(\|\boldsymbol{\theta}\|_r\|\boldsymbol{u}_t\|_{r*})^{\frac{s^*}{2}}\}^{\frac{1}{s^*}} \\
&= \frac{2}{TN}\sqrt{R}E_\sigma\{\sum_{t=1}^{T}\|\boldsymbol{u}_t\|_{r*}^{\frac{s^*}{2}}\}^{\frac{1}{s^*}} \\
&\leq \frac{2}{TN}\sqrt{RM^c}E_\sigma\{\sum_{t=1}^{T}\|\boldsymbol{u}_t\|_{\frac{s^*}{2}}^{\frac{s^*}{2}}\}^{\frac{1}{s^*}}
\end{aligned} \tag{24}$$

Applying Jensen's Inequality, we have that

$$\begin{aligned}
\hat{R}(\mathcal{F}_{s,r}) &\leq \frac{2}{TN}\sqrt{RM^c}(\sum_{t,m=1}^{T,M}E_\sigma\{(u_t^m)^{\frac{s^*}{2}}\})^{\frac{1}{s^*}} \\
&= \frac{2}{TN}\sqrt{RM^c}(\sum_{t,m=1}^{T,M}E_\sigma\|\sum_{i=1}^{N}\sigma_t^i\phi_m(\boldsymbol{x}_t^i)\|_2^{s^*})^{\frac{1}{s^*}}
\end{aligned} \tag{25}$$

Using Theorem 6, we have that

$$\hat{R}(\mathcal{F}_{s,r}) \leq \frac{2}{TN}\sqrt{RM^c s^*}(\sum_{t,m=1}^{T,M}(\text{tr}(\boldsymbol{K}_t^m))^{\frac{s^*}{2}})^{\frac{1}{s^*}} \tag{26}$$

Since we assume that $k_m(\boldsymbol{x},\boldsymbol{x}) \leq 1, \forall m, \boldsymbol{x}$, we have that

$$\hat{R}(\mathcal{F}_{s,r}) \leq \frac{2}{T\sqrt{N}}\sqrt{Rs^* T^{\frac{2}{s^*}} M^{\max\{\frac{1}{r^*},\frac{2}{s^*}\}}} \tag{27}$$

Proof to Corollary 1

First, by following the same proof of $\hat{R}(\mathcal{F}_s) \leq T^{\frac{1}{s'}-\frac{1}{s}}\hat{R}(\mathcal{F}_{s'})$ for any $s' \geq s \geq 1$, we can directly obtain the conclusion that $\hat{R}(\mathcal{F}_{s,r}) \leq T^{\frac{1}{s'}-\frac{1}{s}}\hat{R}(\mathcal{F}_{s',r})$ for any $s' \geq s \geq 1$.

When $r^* \leq \log T$ and $s \in [1, \rho^*]$, where $\rho = 2\log T$, we have that

$$\begin{aligned}\hat{R}(\mathcal{F}_{s,r}) &\leq T^{\frac{1}{\rho^*}-\frac{1}{s}}\hat{R}(\mathcal{F}_{\rho^*,r}) \\ &= \frac{2T^{\frac{1}{s^*}}}{T\sqrt{Ne}}\sqrt{2eRM^{\frac{1}{r^*}}\log T} \\ &= \frac{2}{T\sqrt{N}}\sqrt{RT^{\frac{2}{s^*}}\rho M^{\frac{1}{r^*}}}\end{aligned} \tag{28}$$

When $s > \rho^*$, obviously, we have $\hat{R}(\mathcal{F}_{s,r}) \leq \frac{2}{T\sqrt{N}}\sqrt{RT^{\frac{2}{s^*}}s^* M^{\frac{1}{r^*}}}$

Similarly, for $r^* \geq \log MT$ and $s \in [1, \rho^*]$, where $\rho = 2\log MT$, we have that

$$\begin{aligned}\hat{R}(\mathcal{F}_{s,r}) &\leq T^{\frac{1}{\rho^*}-\frac{1}{s}}\hat{R}(\mathcal{F}_{\rho^*,r}) \\ &= \frac{T^{\frac{1}{s^*}}}{\sqrt{T^{\frac{1}{\log MT}}}}\frac{2}{T\sqrt{N}}\sqrt{2Re\log MT} \\ &= \frac{2}{T\sqrt{N}}\sqrt{2RT^{\frac{2}{s^*}}M^{\frac{1}{\log MT}}\log MT} \\ &= \frac{2}{T\sqrt{N}}\sqrt{RT^{\frac{2}{s^*}}\rho M^{\frac{2}{\rho}}}\end{aligned} \tag{29}$$

When $s > \rho^*$, obviously, we have $\hat{R}(\mathcal{F}_{s,r}) \leq \frac{2}{T\sqrt{N}}\sqrt{RT^{\frac{2}{s^*}}s^*M^{\frac{2}{s^*}}}$

Proof to Theorem 14

We have already show that the ERC of $\mathcal{F}_s$ is

$$\hat{R}(\mathcal{F}_s) = \frac{2}{TN}E_\sigma\{\sup_{\boldsymbol{\alpha},\boldsymbol{\lambda}}\sum_{t=1}^{T}\boldsymbol{\sigma}_t'\boldsymbol{K}_t\boldsymbol{\alpha}_t\} \tag{30}$$

It is not difficult to see that optimizing the following problem

$$\begin{aligned}&\sup_{\boldsymbol{\alpha},\boldsymbol{\lambda}}\sum_{t=1}^{T}\boldsymbol{\sigma}_t'\boldsymbol{K}_t\boldsymbol{\alpha}_t \\ &\text{s.t. } \boldsymbol{\alpha}_t'\boldsymbol{K}_t\boldsymbol{\alpha}_t \leq \lambda_t^2 R \\ &\|\boldsymbol{\lambda}\|_s \leq 1\end{aligned} \tag{31}$$

with respect to $\boldsymbol{\alpha}_t$ must achieves its optimum at the boundary, *i.e.*, the optimal $\boldsymbol{\alpha}_t$ must satisfy

$\boldsymbol{\alpha}_t'\boldsymbol{K}\boldsymbol{\alpha}_t = \lambda_t^2 R$. Therefore, Problem (31) can be re-written as

$$\begin{aligned} &\sup_{\boldsymbol{\alpha},\boldsymbol{\lambda}} \sum_{t=1}^{T} \boldsymbol{\sigma}_t'\boldsymbol{K}_t\boldsymbol{\alpha}_t \\ &\text{s.t. } \boldsymbol{\alpha}_t'\boldsymbol{K}_t\boldsymbol{\alpha}_t = \lambda_t^2 R \\ &\|\boldsymbol{\lambda}\|_s \leq 1 \end{aligned} \tag{32}$$

Substituting the first constraint into the second one directly leads to the result. The proof regarding to $\mathcal{F}_{s,r}$ is similar, and therefore we omit it.

Proof to Theorem 16

First, note that if $\hat{\boldsymbol{x}}$ solves Problem (5.12), then the $\hat{\boldsymbol{a}}_s$'s with $\hat{\boldsymbol{a}}_s \in \boldsymbol{\Omega}(\boldsymbol{a})$that satisfy $g(\hat{\boldsymbol{a}}_s, \hat{\boldsymbol{x}}) = 0$ are local maxima of $g(\boldsymbol{a}, \hat{\boldsymbol{x}})$. According to the assumption, there is a finite number of such $\hat{\boldsymbol{a}}_s$'s, which means that the problem has only a finite number of active constraints. We can therefore provide the KKT necessary condition for $\hat{\boldsymbol{x}}$ to solve Problem (5.12):

**KKT condition for Problem (5.12)** Let $\hat{\boldsymbol{x}}$ solves Problem (5.12); assume there are $S$ $\hat{\boldsymbol{a}}_s$'s such that $g(\hat{\boldsymbol{a}}_s,\ \hat{\boldsymbol{x}}) = 0,\ s = 1, \cdots, S$. Then, there exist $\hat{\lambda}_s,\ s = 0, \cdots, S$; $\hat{\rho}_u,\ u = 1, \cdots, U$; $\hat{\mu}_v,\ v = 1, \cdots, V$, not all zero, such that

$$\hat{\lambda}_0\phi_j(\hat{\boldsymbol{x}}) + \sum_{s=1}^{S} \hat{\lambda}_s\psi_j(\hat{\boldsymbol{a}}_s, \hat{\boldsymbol{x}}) + \sum_{u=1}^{U} \hat{\rho}_u\tau_{u,j}(\hat{\boldsymbol{x}}) + \sum_{v=1}^{V} \hat{\mu}_v\omega_{v,j}(\hat{\boldsymbol{x}}) = 0,\ j = 1, \cdots, n \tag{33}$$

where $\phi_j(\hat{\boldsymbol{x}}),\ \psi_j(\hat{\boldsymbol{a}}_s, \hat{\boldsymbol{x}}),\ \tau_{u,j}(\hat{\boldsymbol{x}}),\ \omega_{v,j}(\hat{\boldsymbol{x}})$ denote $\frac{\partial f(\boldsymbol{x})}{\partial x_j}|_{\boldsymbol{x}=\hat{\boldsymbol{x}}},\ \frac{\partial g(\hat{\boldsymbol{a}}_s,\boldsymbol{x})}{\partial x_j}|_{\boldsymbol{x}=\hat{\boldsymbol{x}}},\ \frac{\partial l_u(\boldsymbol{x})}{\partial x_j}|_{\boldsymbol{x}=\hat{\boldsymbol{x}}},\ \frac{\partial r_v(\boldsymbol{x})}{\partial x_j}|_{\boldsymbol{x}=\hat{\boldsymbol{x}}}$ respectively.

This is a direct extension of the KKT condition of the following standard SIP problem provided in [95]

$$\begin{aligned} &\min_{\boldsymbol{x}\in\mathbb{R}^n} f(\boldsymbol{x}) \\ &\text{s.t. } g(\boldsymbol{a},\boldsymbol{x})\leq 0,\ \forall \boldsymbol{a}\in\boldsymbol{\Omega}(\boldsymbol{a}). \end{aligned} \tag{34}$$

For the EPF-based Problem (5.14), we can similarly construct the KKT condition:

**KKT condition for Problem (5.14)** Let $\hat{\boldsymbol{x}}$ solve Problem (5.14), there exist $\hat{\lambda}'_0$ and $\hat{\lambda}'_s,\ s\in I(\hat{\boldsymbol{x}});\ \hat{\rho}'_u,\ u=1,\cdots,U;\ \hat{\mu}'_v,\ v=1,\cdots,V$, not all zero, such that

$$\hat{\lambda}'_0\phi_j(\hat{\boldsymbol{x}})+\sum_{s\in I(\hat{\boldsymbol{x}})}\hat{\lambda}'_s\psi_j(\hat{\boldsymbol{a}}_s,\hat{\boldsymbol{x}})+\sum_{u=1}^{U}\hat{\rho}'_u\tau_{u,j}(\hat{\boldsymbol{x}})+\sum_{v=1}^{V}\hat{\mu}'_v\omega_{v,j}(\hat{\boldsymbol{x}})=0, j=1,\cdots,n \tag{35}$$

Based on these conditions, we arrive at the conclusion.

## Proof to Theorem 17

Before delving into the details, we introduce the sketch of the proof. The proof includes two major stages. In the first stage, we prove that $\boldsymbol{d}^{(k)}=\hat{\boldsymbol{x}}-\boldsymbol{x}^{(k)}$ is a descent direction if $\hat{\boldsymbol{x}}^{(k)}$ is the solution to the following problem:

$$\begin{aligned} &\min_{\boldsymbol{x}} f(\boldsymbol{x})+\nu\sum_{i\in I(\boldsymbol{x}_k)} g(\boldsymbol{a}_i^{(k)},\boldsymbol{x})_+ \\ &\text{s.t. } l_u(\boldsymbol{x})=0,\forall u;\ r_v(\boldsymbol{x})\leq 0,\forall v. \end{aligned} \tag{36}$$

Note that since $\boldsymbol{x}^{(k)}$ is given in the $k$-th iteration and thus is fixed, $I(\boldsymbol{x}^{(k)})$ and $\boldsymbol{a}_i^{(k)}$'s are also fixed.

In the second stage, we prove that the $\hat{\boldsymbol{x}}$ described in the theorem is a solution to Problem (36) when $\nu > 1$.

**Stage 1**. We need to prove that the directional derivative of the objective function of Problem (36) is negative, given the direction $\boldsymbol{d}^{(k)} = \hat{\boldsymbol{x}} - \boldsymbol{x}^{(k)}$. Since we only consider $\bar{g}$ to be affine in the individual entries of the kernel matrix, therefore, $g(\boldsymbol{a}_i^{(k)}, \boldsymbol{x}^{(k)})$ could be written as

$$g(\boldsymbol{a}_i^{(k)}, \boldsymbol{x}^{(k)}) = \sum_{t=1}^{T}[\sum_{m=1}^{M} \theta_t^{m,(k)}(\boldsymbol{b}_t^1(\boldsymbol{a}_i^{(k)})' \hat{\boldsymbol{K}}_t^m + b_t^2(\boldsymbol{a}_i^{(k)}))] - \omega^{(k)} \tag{37}$$

where $\boldsymbol{b}_t^1(\cdot)$ is a vector-valued function, $b_t^2(\cdot)$ is a scalar function, $\hat{\boldsymbol{K}}_t^m$ is a vector with elements $k_m(\boldsymbol{x}_t^i, \boldsymbol{x}_t^j),\ i, j = 1, \cdots, N_t$. Define $I_+(\boldsymbol{x}) = \{i \in I(\boldsymbol{x}),\ g(\boldsymbol{a}_i, \boldsymbol{x}) > 0\}$, $I_0(\boldsymbol{x}) = \{i \in I(\boldsymbol{x}), g(\boldsymbol{a}_i, \boldsymbol{x}) = 0\}$, $I_-(\boldsymbol{x}) = \{i \in I(\boldsymbol{x}), g(\boldsymbol{a}_i, \boldsymbol{x}) < 0\}$, the directional derivative can be calculated as

$$\begin{aligned} G(\boldsymbol{x}^{(k)}, \boldsymbol{d}^{(k)}, \nu) = \boldsymbol{d}^{(k)'}\phi(\boldsymbol{x}^{(k)}) &+ \nu \sum_{i \in I_0(\boldsymbol{x}^{(k)})} [\boldsymbol{d}^{(k)'}\psi(\boldsymbol{a}_i^{(k)}, \boldsymbol{x}^{(k)})]_+ \\ &+ \nu \sum_{i \in I_+(\boldsymbol{x}^{(k)})} \boldsymbol{d}^{(k)'}\psi(\boldsymbol{a}_i^{(k)}, \boldsymbol{x}^{(k)}) \end{aligned} \tag{38}$$

Let $\boldsymbol{d}^{(k)} = \hat{\boldsymbol{x}} - \boldsymbol{x}^{(k)}$, $H_{t,i}^{m,(k)} = \boldsymbol{b}_t^1(\boldsymbol{a}_i^{(k)})' \hat{\boldsymbol{K}}_t^m + b_t^2(\boldsymbol{a}_i^{(k)})$, $\boldsymbol{H}_i^{(k)}$ be the vector with elements $H_{t,i}^{m,(k)}$, $m = 1, \cdots, M$, $t = 1, \cdots, T$. Then, we obtain

$$\begin{aligned} G(\boldsymbol{x}^{(k)}, \boldsymbol{d}^{(k)}, \nu) = (\hat{\omega} - \omega^{(k)}) &\\ &+ \nu \sum_{i \in I_0(\boldsymbol{x}^{(k)})} [\hat{\boldsymbol{\theta}}^{'} \boldsymbol{H}_i^{(k)} - \hat{\omega} - \boldsymbol{\theta}^{(k)'} \boldsymbol{H}_i^{(k)} + \omega^{(k)}]_+ \\ &+ \nu \sum_{i \in I_+(\boldsymbol{x}^{(k)})} (\hat{\boldsymbol{\theta}}^{'} \boldsymbol{H}_i^{(k)} - \hat{\omega} - \boldsymbol{\theta}^{(k)'} \boldsymbol{H}_i^{(k)} + \omega^{(k)}) \end{aligned} \tag{39}$$

Note that $\boldsymbol{\theta}^{(k)'}\boldsymbol{H}_i^{(k)} - \omega^{(k)} = g(\boldsymbol{a}_i^{(k)}, \boldsymbol{x}^{(k)}) = 0$, when $i \in I_0(\boldsymbol{x}^{(k)})$. Then we get

$$\begin{aligned} G(\boldsymbol{x}^{(k)}, \boldsymbol{d}^{(k)}, \nu) = f(\hat{\boldsymbol{x}}) + \nu \sum_{i \in I(\boldsymbol{x}^{(k)})} g(\boldsymbol{a}_i^{(k)}, \hat{\boldsymbol{x}})_+ \\ - f(\boldsymbol{x}^{(k)}) - \nu \sum_{i \in I(\boldsymbol{x}^{(k)})} g(\boldsymbol{a}_i^{(k)}, \boldsymbol{x}^{(k)})_+ \end{aligned} \tag{40}$$

Since $\hat{\boldsymbol{x}}$ minimizes (36), we have that $G(\boldsymbol{x}^{(k)}, \boldsymbol{d}^{(k)}, \nu) \leq 0$.

**Stage 2**. Denote the objective function in (36) as

$$F(\omega, \boldsymbol{\theta}) = F(\boldsymbol{x}) = f(\boldsymbol{x}) + \nu \sum_{i \in I(\boldsymbol{x}^{(k)})} g(\boldsymbol{a}_i^{(k)}, \boldsymbol{x})_+ \tag{41}$$

We first prove that in Problem (36), for $\nu > 1$ and fixed $\boldsymbol{\theta}$, the optimal solution to $\omega$ is $\hat{\omega} = \max_{i \in I(\boldsymbol{x}^{(k)})} \sum_{t=1}^{T} \bar{g}(\boldsymbol{\alpha}_{t,i}^{(k)}, \sum_{m=1}^{M} \theta_t^m \boldsymbol{K}_t^m)$, which gives $F(\hat{\omega}, \boldsymbol{\theta}) = \hat{\omega}$. To justify this is true, first consider another $\bar{\omega} > \hat{\omega}$. This gives $F(\bar{\omega}, \boldsymbol{\theta}) = \bar{\omega} > \hat{\omega}$. On the other hand, consider $\bar{\omega} < \hat{\omega}$, then we have that

$$F(\hat{\omega}, \boldsymbol{\theta}) - F(\bar{\omega}, \boldsymbol{\theta}) = \hat{\omega} - \bar{\omega} - \nu \sum_{i \in I(\boldsymbol{x}^{(k)})} (\sum_{t=1}^{T} \bar{g}(\boldsymbol{\alpha}_{t,i}^{(k)}, \sum_{m=1}^{M} \theta_t^m \boldsymbol{K}_t^m) - \bar{\omega})_+ \tag{42}$$

For simplicity, assume $\bar{\omega}$ is such that only one element in the summation over $i$ not zero, then

$$F(\hat{\omega}, \boldsymbol{\theta}) - F(\bar{\omega}, \boldsymbol{\theta}) = \hat{\omega} - \bar{\omega} - \nu(\hat{\omega} - \bar{\omega}) \tag{43}$$

Since $\nu > 1$, the above quantity is negative, which gives $F(\hat{\omega}, \boldsymbol{\theta}) < F(\bar{\omega}, \boldsymbol{\theta})$. When $\bar{\omega}$ is such that more elements in the summation over $i$ are not zero, the rationale is similar. Therefore, we proved

that for fixed $\boldsymbol{\theta}$, the $\hat{\omega}$ defined above is the optimum.

Based on the above fact, to minimize $F(\omega, \boldsymbol{\theta})$, we need find $\boldsymbol{\theta}$ such that $\max_{i\in I(\boldsymbol{x})} \sum_{t=1}^{T} \bar{g}(\boldsymbol{\alpha}_{t,i}, \sum_{m=1}^{M} \theta_t^m \boldsymbol{K}_t^m)$ is minimized:

$$\min_{\boldsymbol{\theta}} \max_{i\in I(\boldsymbol{x}^{(k)})} \sum_{t=1}^{T} \bar{g}(\boldsymbol{\alpha}_{t,i}^{(k)}, \sum_{m=1}^{M} \theta_t^m \boldsymbol{K}_t^m) \tag{44}$$

This problem is equivalent to the following:

$$\begin{aligned} &\min_{\boldsymbol{\theta}} \lambda \\ &\text{s.t. } \sum_{t=1}^{T} \bar{g}(\boldsymbol{\alpha}_{t,i}^{(k)}, \sum_{m=1}^{M} \theta_t^m \boldsymbol{K}_t^m) \leq \lambda, \forall i \in I(\boldsymbol{x}^{(k)}). \end{aligned} \tag{45}$$

Therefore, we only need to solve the $\left|I^{(k)}\right|$ problems $\hat{\boldsymbol{\theta}}_i = \arg\min_{\boldsymbol{\theta}} \sum_{t=1}^{T} \bar{g}(\boldsymbol{\alpha}_{t,i}^{(k)}, \sum_{m=1}^{M} \theta_t^m \boldsymbol{K}_t^m)$, $\forall i \in I(\boldsymbol{x}^{(k)})$, and find the one, say $\hat{\boldsymbol{\theta}}_{i_0}$, that minimizes the quantity $\max_{j\in I(\boldsymbol{x}^{(k)})}\{\sum_{t=1}^{T} \bar{g}(\boldsymbol{\alpha}_{t,j}^{(k)}, \sum_{m=1}^{M} \hat{\theta}_{t,i_0}^m \boldsymbol{K}_t^m)\}$.

## Proof to Proposition 2

The results for $p = 1, q = 1$ and $\boldsymbol{c}_t \succeq \mathbf{0}$ are obvious. We in the following assume that there is at least one negative element in each $\boldsymbol{c}_t$. We first prove the result for $p > 1, q > 1$. In this situation, the problem can be written as

$$\begin{aligned} &\min_{\boldsymbol{\theta}\in\mathbb{R}^n, \boldsymbol{\sigma}\in\mathbb{R}^T} \boldsymbol{c}'\boldsymbol{\theta} \\ &\text{s.t. } \boldsymbol{\theta} \succeq \mathbf{0},\ \|\boldsymbol{\theta}_t\|_p = \sigma^t,\ \forall\ t; \|\boldsymbol{\sigma}\|_q \leq a. \end{aligned} \tag{46}$$

Note that the above problem is equivalent to the following:

$$\begin{aligned} &\min_{\boldsymbol{\theta}\in\mathbb{R}^n,\boldsymbol{\sigma}\in\mathbb{R}^T} \boldsymbol{c}'\boldsymbol{\theta} \\ &\text{s.t. } \boldsymbol{\theta} \succeq \mathbf{0},\ \|\boldsymbol{\theta}_t\|_p \leq \sigma^t,\ \forall\ t; \|\boldsymbol{\sigma}\|_q \leq a. \end{aligned} \tag{47}$$

since the optimum $\boldsymbol{\theta}_t$ for the above problem must achieve on the boundary. The Lagrangian of Problem (47) is formed as

$$L = \sum_{t=1}^{T} (\boldsymbol{c}_t'\boldsymbol{\theta}_t + \alpha_t(\|\boldsymbol{\theta}_t\|_p - \sigma_t) - \boldsymbol{\beta}_t'\boldsymbol{\theta}_t) + \gamma(\|\boldsymbol{\sigma}\|_q - 1) \tag{48}$$

where $\alpha$ and $\boldsymbol{\beta}$ are the Lagrangian multipliers of the inequality constraints. Set its partial derivative with respect to $\boldsymbol{\theta}_t$ and $\boldsymbol{\sigma}$ to $0$ and add the complementary slackness conditions:

$$\begin{cases} \boldsymbol{c}_t + \alpha_t \|\boldsymbol{\theta}_t\|_p^{1-p} (\boldsymbol{\theta}_t)^{p-1} - \boldsymbol{\beta}_t = \mathbf{0};\ \forall t \\ \gamma \|\boldsymbol{\sigma}\|_q^{1-q} \boldsymbol{\sigma}^{q-1} - \boldsymbol{\alpha} = \mathbf{0}; \\ \alpha_t(\|\boldsymbol{\theta}_t\|_p - \sigma_t) = 0;\ \forall t \\ \beta_t^m \theta_t^m = 0;\ \forall m,\ t \\ \gamma(\|\boldsymbol{\sigma}\|_q - 1) = 0. \end{cases} \tag{49}$$

Note that $\boldsymbol{\alpha}$ should not be $\mathbf{0}$, otherwise the minimum of $L$ would be either minus infinity or $0$, which are trivial cases. Therefore, we have

$$\begin{cases} \|\boldsymbol{\theta}_t\|_p = \sigma_t;\ \forall t \\ \boldsymbol{\theta}_t = \sigma_t [\frac{1}{\alpha_t}(\boldsymbol{\beta}_t - \boldsymbol{c}_t)]^{\frac{1}{p-1}}\ \forall t \end{cases} \tag{50}$$

Considering (49) and that $\alpha_t > 0$, assume $\sigma_t > 0$, we have $\beta_t^m(\beta_t^m - c_t^m)^{\frac{1}{p-1}} = 0,\ \forall m$. Since $\beta_t^m \geq 0$, we conclude

$$\begin{cases} \beta_t^m = 0 & \text{if } c_t^m \leq 0 \\ \beta_t^m = c_t^m & \text{otherwise.} \end{cases} \tag{51}$$

So, let $\tilde{\boldsymbol{c}}_t = [\tilde{c}_t^1, \cdots, \tilde{c}_t^n]'$ and $\tilde{c}_t^m = \beta_t^m - c_t^m$ , we get $\tilde{c}_t^m = \max\{-c_t^m, 0\},\ \forall m$. We end up with $\boldsymbol{\theta}_t = \sigma_t(\frac{\tilde{\boldsymbol{c}}_t}{\alpha_t})^{\frac{1}{p-1}} \propto (\tilde{\boldsymbol{c}}_t)^{\frac{1}{p-1}}$. Since $\boldsymbol{\theta}_t$ needs to satisfy $\|\boldsymbol{\theta}_t\|_p = \sigma_t$, we normalize $(\tilde{\boldsymbol{c}}_t)^{\frac{1}{p-1}}$ and get

$$\boldsymbol{\theta}_t = \sigma_t \frac{(\tilde{\boldsymbol{c}}_t)^{\frac{1}{p-1}}}{\|\tilde{\boldsymbol{c}}_t\|_{\frac{p}{p-1}}^{\frac{1}{p-1}}} \tag{52}$$

Note that in the above derivation, we assumed that $\sigma_t \neq 0$. If $\sigma_t = 0$, we know immediately from (50) that $\boldsymbol{\theta}_t = \mathbf{0}$. For $\boldsymbol{\sigma}$, with the same logic, we get that $(\sigma_t)^{q-1} \propto \alpha_t = \|\tilde{\boldsymbol{c}}_t\|_{\frac{p}{p-1}}$. Note that the optimal $\boldsymbol{\sigma}$ must achieve on the boundary, we get the solution

$$\sigma_t = \frac{a\|\boldsymbol{c}_t\|_{r+1}^s}{(\sum_{t=1}^T \|\boldsymbol{c}_t\|_{r+1}^{s+1})^{1/q}}. \tag{53}$$

When $q = 1$, the above prove for $\boldsymbol{\theta}_t$ is still valid. Therefore, we plug (52) into (47), and get the following:

$$\begin{aligned} &\min_{\boldsymbol{\sigma}} \sum_{t=1}^T \sigma_t \|\tilde{\boldsymbol{c}}_t\|_{r+1} \\ &\text{s.t. } \boldsymbol{\sigma} > \mathbf{0},\ \sum_{t=1}^t \sigma_t \leq a \end{aligned} \tag{54}$$

Note the constraint $\boldsymbol{\sigma} > \mathbf{0}$ is added since $\sigma_t = \|\boldsymbol{\theta}_t\|_p$. Obviously, the optimum $\boldsymbol{\sigma}$ is $\sigma_t = a[t = \tilde{t}]$, where $\tilde{t} = \arg\min_t\{\|\hat{\boldsymbol{c}}_t\|_{r+1}\}$.

When $p = 1$, we first optimize with respect to $\boldsymbol{\theta}$. It is not difficult to observe that, by fixing $\boldsymbol{\sigma}$, our problem can be split into $T$ minimization problems:

$$\begin{aligned} &\min_{\boldsymbol{\theta}_t} \boldsymbol{c}_t'\boldsymbol{\theta}_t \\ &\text{s.t. } \boldsymbol{\theta}_t \succeq \mathbf{0}, \sum_{i=1}^{N_t} \theta_t^i \leq \sigma_t \end{aligned} \tag{55}$$

The solution to the above problem is obviously $\boldsymbol{\theta}_t^* = \sigma_t \boldsymbol{e}_t^{i_t}$. Plugging this result into Problem (47) and optimize with respect to $\boldsymbol{\sigma}$, by using the strategy similar to the previous proof, we immediately get the result.

# LIST OF REFERENCES


[1] Jonathan Aflalo, Aharon Ben-Tal, Chiranjib Bhattacharyya, Jagarlapudi Saketha Nath, and Sankaran Raman. Variable sparsity kernel learning. *Journal of Machine Learning Research*, 12:565–592, 2011.

[2] Arvind Agarwal, Samuel Gerber, and Hal Daume. Learning multiple tasks using manifold regularization. In *Advances in Neural Information Processing Systems*, pages 46–54, 2010.

[3] Morteza Alamgir, Moritz Grosse-wentrup, and Yasemin Altun. Multitask learning for brain-computer interfaces. In *International Conference on Artificial Intelligence and Statistics*, pages 17–24, 2010.

[4] Rie Kubota Ando and Tong Zhang. A framework for learning predictive structures from multiple tasks and unlabeled data. *Journal of Machine Learning Research*, 6, 2005.

[5] Andreas Argyriou, Theodoros Evgeniou, and Massimiliano Pontil. Multi-task feature learning. In *Advances in Neural Information Processing Systems*, pages 41–48, 2007.

[6] Andreas Argyriou, Theodoros Evgeniou, and Massimiliano Pontil. Convex multi-task feature learning. *Machine Learning*, 73(3):243–272, 2008.

[7] Andreas Argyriou, Massimiliano Pontil, Yiming Ying, and Charles A Micchelli. A spectral regularization framework for multi-task structure learning. In *Advances in Neural Information Processing Systems*, pages 25–32, 2007.

[8] Peter L. Bartlett and Shahar Mendelson. Rademacher and gaussian complexities: Risk bounds and structural results. *Journal of Machine Learning Research*, 3:463–482, 2002.

[9] Jonathan Baxter. A model of inductive bias learning. *Journal of Artificial Intelligence Research*, 12:149–198, 2000.

[10] Jinbo Bi, Tao Xiong, Shipeng Yu, Murat Dundar, and R Bharat Rao. An improved multi-task learning approach with applications in medical diagnosis. In *Machine Learning and Knowledge Discovery in Databases*, pages 117–132. Springer, 2008.

[11] Steffen Bickel, Jasmina Bogojeska, Thomas Lengauer, and Tobias Scheffer. Multi-task learning for hiv therapy screening. In *Proceedings of the 25th International Conference on Machine Learning*, pages 56–63, 2008.

[12] Edwin V Bonilla, Felix V Agakov, and Christopher Williams. Kernel multi-task learning using task-specific features. In *International Conference on Artificial Intelligence and Statistics*, pages 43–50, 2007.

[13] Edwin V Bonilla, Kian M Chai, and Christopher Williams. Multi-task gaussian process prediction. In *Advances in Neural Information Processing Systems*, pages 153–160, 2007.

[14] Qiong Cao, Zheng-Chu Guo, and Yiming Ying. Generalization bounds for metric and similarity learning. *arXiv preprint arXiv:1207.5437*, 2012.

[15] Andrea Caponnetto, Charles A. Micchelli, Massimiliano Pontil, and Yiming Ying. Universal multi-task kernels. *Journal of Machine Learning Research*, 9:1615–1646, 2008.

[16] Rich Caruana. Multitask learning. *Machine Learning*, 28:41–75, 1997.

[17] Richard Caruana. Multitask learning: A knowledge-based source of inductive bias. In *Proceedings of the Tenth International Conference on Machine Learning*, pages 41–48, 1993.

[18] Chih-Chung Chang and Chih-Jen Lin. LIBSVM: A library for support vector machines. *ACM Transactions on Intelligent Systems and Technology*, 2:27:1–27:27, 2011. Software available at `http://www.csie.ntu.edu.tw/~cjlin/libsvm`.

[19] Olivier Chapelle, Pannagadatta Shivaswamy, Srinivas Vadrevu, Kilian Weinberger, Ya Zhang, and Belle Tseng. Multi-task learning for boosting with application to web search

ranking. In *Proceedings of the 16th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, pages 1189–1198, 2010.

[20] Jianhui Chen, Ji Liu, and Jieping Ye. Learning incoherent sparse and low-rank patterns from multiple tasks. *ACM Transactions on Knowledge Discovery from Data (TKDD)*, 5(4):22, 2012.

[21] Jianhui Chen, Lei Tang, Jun Liu, and Jieping Ye. A convex formulation for learning shared structures from multiple tasks. In *Proceedings of the 26th Annual International Conference on Machine Learning*, pages 137–144, 2009.

[22] Jianhui Chen, Jiayu Zhou, and Jieping Ye. Integrating low-rank and group-sparse structures for robust multi-task learning. In *Proceedings of the 17th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, pages 42–50, 2011.

[23] Xi Chen, Weike Pan, James T Kwok, and Jaime G Carbonell. Accelerated gradient method for multi-task sparse learning problem. In *Data Mining, 2009. ICDM'09. Ninth IEEE International Conference on*, pages 746–751, 2009.

[24] Corinna Cortes, Mehryar Mohri, and Afshin Rostamizadeh. $l_2$ regularization for learning kernels. In *Proceedings of the Twenty-Fifth Conference on Uncertainty in Artificial Intelligence*, pages 109–116, 2009.

[25] Corinna Cortes, Mehryar Mohri, and Afshin Rostamizadeh. Learning non-linear combinations of kernels. In *Advances in Neural Information Processing Systems*, pages 396–404, 2009.

[26] Corinna Cortes, Mehryar Mohri, and Afshin Rostamizadeh. Generalization bounds for learning kernels. In *Proceedings of the 27th International Conference on Machine Learning (ICML-10)*, pages 247–254, 2010.

[27] Corinna Cortes and Vladimir Vapnik. Support-vector networks. *Machine Learning*, 20:273–297, 1995.

[28] Francesco Dinuzzo. Learning output kernels for multi-task problems. *Neurocomputing*, 118:119–126, 2013.

[29] Miro Dudik, Zaid Harchaoui, Jérôme Malick, et al. Lifted coordinate descent for learning with trace-norm regularization. In *AISTATS-Proceedings of the Fifteenth International Conference on Artificial Intelligence and Statistics-2012*, volume 22, pages 327–336, 2012.

[30] Theodoros Evgeniou, Charles A. Micchelli, and Massimiliano Pontil. Learning multiple tasks with kernel methods. *Journal of Machine Learning Research*, 6:615–637, 2005.

[31] Theodoros Evgeniou and Massimiliano Pontil. Regularized multi–task learning. In *Proceedings of the Tenth ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, pages 109–117, 2004.

[32] Hongliang Fei and Jun Huan. Structured feature selection and task relationship inference for multi-task learning. *Knowledge and Information Systems*, 35(2):345–364, 2013.

[33] A. Frank and A. Asuncion. UCI machine learning repository, 2010.

[34] Joumana Ghosn and Yoshua Bengio. Multi-task learning for stock selection. *Advances in Neural Information Processing Systems*, pages 946–952, 1997.

[35] Pinghua Gong, Jieping Ye, and Chang-shui Zhang. Multi-stage multi-task feature learning. In *Advances in Neural Information Processing Systems*, pages 1988–1996, 2012.

[36] Pinghua Gong, Jieping Ye, and Changshui Zhang. Robust multi-task feature learning. In *Proceedings of the 18th ACM SIGKDD international conference on Knowledge discovery and data mining*, pages 895–903, 2012.

[37] Pinghua Gong, Jiayu Zhou, Wei Fan, and Jieping Ye. Efficient multi-task feature learning with calibration. In *Proceedings of the 20th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, pages 761–770, 2014.

[38] M. Grant and S. Boyd. Graph implementations for nonsmooth convex programs. In V. Blondel, S. Boyd, and H. Kimura, editors, *Recent Advances in Learning and Control*, Lecture Notes in Control and Information Sciences, pages 95–110. Springer-Verlag Limited, 2008.

[39] M. Grant and S. Boyd. CVX: Matlab software for disciplined convex programming, version 1.21, April 2011.

[40] Quanquan Gu, Zhenhui Li, and Jiawei Han. Joint feature selection and subspace learning. In *Proceedings of the Twenty-Second International Joint Conference on Artificial Intelligence-Volume Volume Two*, pages 1294–1299, 2011.

[41] Ke Hou, Zirui Zhou, Anthony Man-Cho So, and Zhi-Quan Luo. On the linear convergence of the proximal gradient method for trace norm regularization. In *Advances in Neural Information Processing Systems*, pages 710–718, 2013.

[42] Ali Jalali, Sujay Sanghavi, Chao Ruan, and Pradeep K Ravikumar. A dirty model for multi-task learning. In *Advances in Neural Information Processing Systems*, pages 964–972, 2010.

[43] Pratik Jawanpuria and J. Saketha Nath. Multi-task multiple kernel learning. In *SIAM Conference on Data Mining*, 2011.

[44] Tony Jebara. Multi-task feature and kernel selection for svms. In *Proceedings of the Twenty-First International Conference on Machine Learning*, page 55, 2004.

[45] Tony Jebara. Multitask sparsity via maximum entropy discrimination. *Journal of Machine Learning Research*, 12:75–110, 2011.

[46] Shuiwang Ji and Jieping Ye. An accelerated gradient method for trace norm minimization. In *Proceedings of the 26th Annual International Conference on Machine Learning*, pages 457–464, 2009.

[47] Sham M. Kakade, Shai Shalev-Shwartz, and Ambuj Tewari. Regularization techniques for learning with matrices. *Journal of Machine Learning Research*, 13:1865–1890, 2012.

[48] Zhuoliang Kang, Kristen Grauman, and Fei Sha. Learning with whom to share in multi-task feature learning. In *Proceedings of the 28th International Conference on Machine Learning (ICML-11)*, pages 521–528, 2011.

[49] Tsuyoshi Kato, Hisashi Kashima, Masashi Sugiyama, and Kiyoshi Asai. Multi-task learning via conic programming. In *Advances in Neural Information Processing Systems*, pages 737–744, 2008.

[50] Marius Kloft, Ulf Brefeld, Pavel Laskov, Klaus-Robert Müller, Alexander Zien, and Sören Sonnenburg. Efficient and accurate $l_p$-norm multiple kernel learning. In *Advances in Neural Information Processing Systems*, pages 997–1005, 2009.

[51] Marius Kloft, Ulf Brefeld, Pavel Laskov, and Soren Sonnenburg. Non-sparse multiple kernel learning. In *NIPS Workshop on Kernel Learning: Automatic Selection of Optimal Kernels*, 2008.

[52] Marius Kloft, Ulf Brefeld, Soren Sonnenburg, and Alexander Zien. $l_p$-norm multiple kernel learning. *Journal of Machine Learning Research*, 12:953–997, 2011.

[53] Abhishek Kumar and Hal Daume. Learning task grouping and overlap in multi-task learning. In *Proceedings of the 29th International Conference on Machine Learning (ICML-12)*, pages 1383–1390, 2012.

[54] Stanislaw Kwapien and Wojbor A. Woyczynski. *Random Series and Stochastic Integrals: Single and Multiple*. Birkhauser, 1992.

[55] Gert R. G. Lanckriet, Nello Cristianini, Peter Bartlett, Laurent El Ghaoui, and Michael I. Jordan. Learning the kernel matrix with semidefinite programming. *Journal of Machine Learning Research*, 5:27–72, December 2004.

[56] Cong Li, Michael Georgiopoulos, and C. Georgios Anagnostopoulos. Conic multi-task classification. In *European Conference on Machine Learning and Principles and Practice of Knowledge Discovery in Databases*, 2014.

[57] Cong Li, Michael Georgiopoulos, and Georgios C Anagnostopoulos. Pareto-path multi-task multiple kernel learning. *ArXiv e-prints*, April 2014.

[58] Jia Li, Yonghong Tian, Tiejun Huang, and Wen Gao. Probabilistic multi-task learning for visual saliency estimation in video. *International Journal of Computer Vision*, 90(2):150–165, 2010.

[59] Han Liu, Mark Palatucci, and Jian Zhang. Blockwise coordinate descent procedures for the multi-task lasso, with applications to neural semantic basis discovery. In *Proceedings of the 26th Annual International Conference on Machine Learning*, pages 649–656, 2009.

[60] Jianqiang Liu, Charles A Micchelli, Rui Wang, and Yuesheng Xu. Finite rank kernels for multi-task learning. *Advances in Computational Mathematics*, 38(2):427–439, 2013.

[61] Jun Liu, Jianhui Chen, Songcan Chen, and Jieping Ye. Learning the optimal neighborhood kernel for classification. In *International Joint Conference on Artificial Intelligence*, 2009.

[62] Jun Liu, Shuiwang Ji, and Jieping Ye. Multi-task feature learning via efficient l 2, 1-norm minimization. In *Proceedings of the Twenty-Fifth Conference on Uncertainty in Artificial Intelligence*, pages 339–348, 2009.

[63] Qi Liu, Qian Xu, Vincent W Zheng, Hong Xue, Zhiwei Cao, and Qiang Yang. Multi-task learning for cross-platform sirna efficacy prediction: an in-silico study. *BMC Bioinformatics*, 11(1):1–16, 2010.

[64] Karim Lounici, Massimiliano Pontil, Alexandre B Tsybakov, and Sara Van De Geer. Taking advantage of sparsity in multi-task learning. *arXiv preprint arXiv:0903.1468*, 2009.

[65] Andreas Maurer. Bounds for linear multi-task learning. *Journal of Machine Learning Research*, 7:117–139, 2006.

[66] Andreas Maurer. The rademacher complexity of linear transformation classes. In Gbor Lugosi and HansUlrich Simon, editors, *Learning Theory*, volume 4005 of *Lecture Notes in Computer Science*, pages 65–78. Springer Berlin Heidelberg, 2006.

[67] Andreas Maurer, Massi Pontil, and Bernardino Romera-paredes. Sparse coding for multitask and transfer learning. In *Proceedings of the 30th International Conference on Machine Learning (ICML-13)*, volume 28, pages 343–351, 2013.

[68] Andreas Maurer and Massimiliano Pontil. Structured sparsity and generalization. *Journal of Machine Learning Research*, 13:671–690, 2012.

[69] Nishant Mehta, Dongryeol Lee, and Alexander G Gray. Minimax multi-task learning and a generalized loss-compositional paradigm for mtl. In *Advances in Neural Information Processing Systems*, pages 2150–2158, 2012.

[70] Charles A Micchelli and Massimiliano Pontil. Kernels for multi–task learning. In *Advances in Neural Information Processing Systems*, pages 921–928, 2004.

[71] Mehryar Mohri, Afshin Rostamizadeh, and Ameet Talwalkar. *Foundations of machine learning*. The MIT Press, 2012.

[72] Guillaume Obozinski, Ben Taskar, and Michael I Jordan. Joint covariate selection and joint subspace selection for multiple classification problems. *Statistics and Computing*, 20(2):231–252, 2010.

[73] Shibin Parameswaran and Kilian Q Weinberger. Large margin multi-task metric learning. In *Advances in Neural Information Processing Systems*, pages 1867–1875, 2010.

[74] Ting Kei Pong, Paul Tseng, Shuiwang Ji, and Jieping Ye. Trace norm regularization: reformulations, algorithms, and multi-task learning. *SIAM Journal on Optimization*, 20(6):3465–3489, 2010.

[75] Massimiliano Pontil and Andreas Maurer. Excess risk bounds for multitask learning with trace norm regularization. In *Conference on Learning Theory*, pages 55–76, 2013.

[76] Jian Pu, Yu-Gang Jiang, Jun Wang, and Xiangyang Xue. Multiple task learning using iteratively reweighted least square. In *Proceedings of the Twenty-Third international joint conference on Artificial Intelligence*, pages 1607–1613, 2013.

[77] Yanjun Qi, Oznur Tastan, Jaime G Carbonell, Judith Klein-Seetharaman, and Jason Weston. Semi-supervised multi-task learning for predicting interactions between hiv-1 and human proteins. *Bioinformatics*, 26(18):i645–i652, 2010.

[78] Alain Rakotomamonjy, Francis R. Bach, Stephane Canu, and Yves Grandvalet. Simplemkl. *Journal of Machine Learning Research*, 9:2491–2521, 2008.

[79] Alain Rakotomamonjy, Remi Flamary, Gilles Gasso, and Stephane Canu. $l_p - l_q$ penalty for sparse linear and sparse multiple kernel multitask learning. *IEEE Transactions on Neural Networks*, 22:1307–1320, 2011.

[80] Bernardino Romera-Paredes, Andreas Argyriou, Nadia Berthouze, and Massimiliano Pontil. Exploiting unrelated tasks in multi-task learning. In *International Conference on Artificial Intelligence and Statistics*, pages 951–959, 2012.

[81] Avishek Saha, Piyush Rai, Suresh Venkatasubramanian, and Hal Daume. Online learning of multiple tasks and their relationships. In *International Conference on Artificial Intelligence and Statistics*, pages 643–651, 2011.

[82] Wojciech Samek, Alexander Binder, and Motoaki Kawanabe. Multi-task learning via non-sparse multiple kernel learning. In Pedro Real, Daniel Diaz-Pernil, Helena Molina-Abril, Ainhoa Berciano, and Walter Kropatsch, editors, *Computer Analysis of Images and Patterns*, volume 6854 of *Lecture Notes in Computer Science*, pages 335–342. Springer Berlin / Heidelberg, 2011.

[83] Craig Saunders, Alexander Gammerman, and Volodya Vovk. Ridge regression learning algorithm in dual variables. In *(ICML-1998) Proceedings of the 15th International Conference on Machine Learning*, pages 515–521, 1998.

[84] Bernhard Schölkopf, Ralf Herbrich, and Alex J Smola. A generalized representer theorem. In *Computational Learning Theory*, pages 416–426, 2001.

[85] Bernhard Schölkopf, John C. Platt, John Shawe-Taylor, and Alex J. Smola. Estimating the support of a high-dimensional distribution. *Neural Computation*, 13:1443–1471, 2001.

[86] John Shawe-Taylor and Nello Cristianini. *Kernel Methods for Pattern Analysis*. Cambridge University Press, 2004.

[87] N. Shawe-Taylor and A. Kandola. On kernel target alignment. In *Advances in Neural Information and Processing Systems*, 2002.

[88] M. Sion. On general minimax theorems. *Pacific Journal of Mathematics*, 8:171–176, 1958.

[89] Matthieu Solnon, Sylvain Arlot, and Francis Bach. Multi-task regression using minimal penalties. *The Journal of Machine Learning Research*, 13(1):2773–2812, 2012.

[90] PK Srijith and Shirish Shevade. Multi-task learning using shared and task specific information. In *Neural Information Processing*, pages 125–132, 2012.

[91] Lei Tang, Jianhui Chen, and Jieping Ye. On multiple kernel learning with multiple labels. In *Proceedings of the Twenty-First International Joint Conference on Artificial Intelligence*, 2009.

[92] David M.J. Tax and Robert P.W. Duin. Support vector domain description. *Pattern Recognition Letters*, 20:1191–1199, 1999.

[93] Vladimir Vapnik. *Statistical Learning Theory*. Wiley-Interscience, 1998.

[94] Xiaogang Wang, Cha Zhang, and Zhengyou Zhang. Boosted multi-task learning for face verification with applications to web image and video search. In *Computer Vision and Pattern Recognition, 2009. CVPR 2009. IEEE Conference on*, pages 142–149, 2009.

[95] G.A. Watson. Globally convergent methods for semi-infinite programming. *BIT Numerical Mathematics*, 21:392–373, 1981.

[96] Christopher Williams, Stefan Klanke, Sethu Vijayakumar, and Kian M Chai. Multi-task gaussian process learning of robot inverse dynamics. In *Advances in Neural Information Processing Systems*, pages 265–272, 2009.

[97] Xinxing Xu, I.W. Tsang, and Dong Xu. Soft margin multiple kernel learning. *IEEE Transactions on Neural Networks and Learning Systems*, 24:749–761, 2013.

[98] Zenglin Xu, Rong Jin, Haiqin Yang, Irwin King, and Michael R Lyu. Simple and efficient multiple kernel learning by group lasso. In *Proceedings of the 27th International Conference on Machine Learning (ICML-10)*, pages 1175–1182, 2010.

[99] Ya Xue, Xuejun Liao, Laurence Carin, and Balaji Krishnapuram. Multi-task learning for classification with dirichlet process priors. *Journal of Machine Learning Research*, 8:35–63, 2007.

[100] Haiqin Yang, Irwin King, and Michael R Lyu. Online learning for multi-task feature selection. In *Proceedings of the 19th ACM International Conference on Information and Knowledge Management*, pages 1693–1696, 2010.

[101] Xiaolin Yang, Seyoung Kim, and Eric P Xing. Heterogeneous multitask learning with joint sparsity constraints. In *Advances in Neural Information Processing Systems*, pages 2151–2159, 2009.

[102] Kai Yu, Volker Tresp, and Anton Schwaighofer. Learning gaussian processes from multiple tasks. In *Proceedings of the 22nd International Conference on Machine Learning*, pages 1012–1019, 2005.

[103] Shipeng Yu, Volker Tresp, and Kai Yu. Robust multi-task learning with t-processes. In *Proceedings of the 24th International Conference on Machine Learning*, pages 1103–1110, 2007.

[104] Xiao-Tong Yuan, Xiaobai Liu, and Shuicheng Yan. Visual classification with multitask joint sparse representation. *Image Processing, IEEE Transactions on*, 21(10):4349–4360, 2012.

[105] Daoqiang Zhang and Dinggang Shen. Multi-modal multi-task learning for joint prediction of multiple regression and classification variables in alzheimer's disease. *Neuroimage*, 59(2):895–907, 2012.

[106] Jian Zhang, Zoubin Ghahramani, and Yiming Yang. Flexible latent variable models for multi-task learning. *Machine Learning*, 73(3):221–242, 2008.

[107] Tianzhu Zhang, Bernard Ghanem, Si Liu, and Narendra Ahuja. Robust visual tracking via multi-task sparse learning. In *Computer Vision and Pattern Recognition (CVPR), 2012 IEEE Conference on*, pages 2042–2049, 2012.

[108] Tong Zhang. Analysis of multi-stage convex relaxation for sparse regularization. *The Journal of Machine Learning Research*, 11:1081–1107, 2010.

[109] Tong Zhang. Multi-stage convex relaxation for feature selection. *arXiv preprint arXiv:1106.0565*, 2011.

[110] Yu Zhang and Dit-Yan Yeung. Multi-task boosting by exploiting task relationships. In *Machine Learning and Knowledge Discovery in Databases*, pages 697–710. Springer, 2012.

[111] Yu Zhang and Dit-Yan Yeung. Multilabel relationship learning. *ACM Transactions on Knowledge Discovery from Data (TKDD)*, 7(2):7, 2013.

[112] Yu Zhang and Dit-Yan Yeung. A regularization approach to learning task relationships in multitask learning. *ACM Transactions on Knowledge Discovery from Data (TKDD)*, 8(3):12, 2014.

[113] Vincent Wenchen Zheng, Sinno Jialin Pan, Qiang Yang, and Jeffrey Junfeng Pan. Transferring multi-device localization models using latent multi-task learning. In *Proceedings of the Twenty-Second AAAI Conference on Artificial Intelligence*, volume 8, pages 1427–1432, 2008.

[114] Wenliang Zhong and James T Kwok. Convex multitask learning with flexible task clusters. In *Proceedings of the 29th International Conference on Machine Learning (ICML-12)*, pages 49–56, 2012.

[115] Jiayu Zhou, Jianhui Chen, and Jieping Ye. Clustered multi-task learning via alternating structure optimization. In *Advances in Neural Information Processing Systems*, pages 702–710, 2011.

[116] Jiayu Zhou, Lei Yuan, Jun Liu, and Jieping Ye. A multi-task learning formulation for predicting disease progression. In *Proceedings of the 17th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, pages 814–822, 2011.

[117] Yang Zhou, Rong Jin, and Steven Hoi. Exclusive lasso for multi-task feature selection. In *International Conference on Artificial Intelligence and Statistics*, pages 988–995, 2010.

[118] Jun Zhu, Ning Chen, and Eric P Xing. Infinite latent svm for classification and multi-task learning. In *Advances in Neural Information Processing Systems*, pages 1620–1628, 2011.