Advanced Kernel Methods for Multi-Task Learning

by

Carlos Ruiz Pastor

What is the essence of life? To serve others and to do good.

Aristotle.

Abstract

.

Resumen

Small.

Acknowledgements

.

Contents

Abbreviations

ADF	**A**ssumed **D**ensity **F**iltering
AF	**A**cquisition **F**unction
BO	**B**ayesian **O**ptimization
DGP	**D**eep **G**aussian **P**rocess
EI	**E**xpected **I**mprovement
EP	**E**xpectation **P**ropagation
GP	**G**aussian Process
KL	**K**ullback Liebler
MCMC	Markov Chain Monte Carlo
PPESMOC	**P**arallel **P**redictive **E**ntropy **S**earch for **M**ultiobjective **O**ptimization with **C**onstraints
PES	**P**redictive **E**ntropy **S**earch
PESMOC	**P**redictive **E**ntropy **S**earch for **M**ultiobjective **O**ptimization with **C**onstraints
RS	**R**andom **S**earch
UCB	**U**pper **C**onfidence **B**ound

To my family

Chapter 1

Introduction

We begin this manuscript...

1.1 Introduction

1.2 Publications

This section presents, in chronological order, the work published during the doctoral period in which this thesis was written. We also include other research work related to this thesis, but not directly included on it. Finally, this document includes content that has not been published yet and is under revision.

Related Work

Work In Progress

1.3 Summary by Chapters

In this section. . .

Chapter 2 provides an introduction to GPs and the expectation propagation algorithm. Both are necessary concepts for the BO methods that we will describe in the following chapters. This chapter reviews the fundamentals of GPs and why they are so interesting for BO. More concretely, we review the most popular kernels, the analysis of the posterior and predictive distribution and how to tune the hyper-parameters of GPs: whether by maximizing the marginal likelihood or by generating samples from the hyper-parameter posterior distribution. Other alternative probabilistic surrogate models are also described briefly. Some of the proposed approaches of this thesis are extensions of an acquisition function called predictive entropy search, that is based on the expectation propagation approximate inference technique. That is why we provide in this chapter an explanation of the expectation propagation algorithm.

Chapter 3 introduces the basics of BO and information theory. BO works with probabilistic models such as GPs and with acquisition functions such as predictive entropy search, that uses information theory. Having studied GPs in Chapter 2, BO can be now understood and it is described in detail. This chapter will also

describe the most popular acquisition functions, how information theory can be applied in BO and why BO is useful for the hyper-parameter tuning of machine learning algorithms.

Chapter 4 describes an information-theoretical mechanism that generalizes BO to simultaneously optimize multiple objectives under the presence of several constraints. This algorithm is called predictive entropy search for multi-objective BO with constraints (PESMOC) and it is an extension of the predictive entropy search acquisition function that is described in Chapter 3. The chapter compares the empirical performance of PESMOC with respect to a state-of-the-art approach to constrained multi-objective optimization based on the expected improvement acquisition function. It is also compared with a random search through a set of synthetic, benchmark and real experiments.

Chapter 5 addresses the problem that faces BO when not only one but multiple input points can be evaluated in parallel that has been described in Section **??**. This chapter introduces an extension of PESMOC called parallel PESMOC (PPESMOC) that adapts to the parallel scenario. PPESMOC builds an acquisition function that assigns a value for each batch of points of the input space. The maximum of this acquisition function corresponds to the set of points that maximizes the expected reduction in the entropy of the Pareto set in each evaluation. Naive adaptations of PESMOC and the method based on expected improvement for the parallel scenario are used as a baseline to compare their performance with PPESMOC. Synthetic, benchmark and real experiments show how PPESMOC obtains an advantage in most of the considered scenarios. All the mentioned approaches are described in detail in this chapter.

Chapter ?? addresses a transformation that enables standard GPs to deliver better results in problems that contain integer-valued and categorical variables. We can apply BO to problems where we need to optimize functions that contain integer-valued and categorical variables with more guarantees of obtaining a solution with low regret. A critical advantage of this transformation, with respect to other approaches, is that it is compatible with any acquisition function. This transformation makes the uncertainty given by the GPs in certain areas of the space flat. As a consequence, the acquisition function can also be flat in these zones. This phenomenom raises an issue with the optimization of the acquisition function, that must consider the flatness of these areas. We use a one exchange neighbourhood approach to optimize the resultant acquisition function. We test our approach in synthetic and real problems, where we add empirical evidence of the performance of our proposed transformation.

Chapter ?? shows a real problem where BO has been applied with success. In this problem, BO has been used to obtain the optimal parameters of a hybrid Grouping Genetic Algorithm for attribute selection. This genetic algorithm is combined with an Extreme Learning Machine (GGA-ELM) approach for prediction of ocean wave features. Concretely, the significant wave height and the wave energy flux at a goal marine structure facility on the Western Coast of the USA is predicted. This chapter illustrates the experiments where it is shown that BO improves the performance of the GGA-ELM approach. Most importantly, it also outperforms a random search of the hyper-parameter space and the human expert criterion.

Chapter ?? provides a summary of the work done in this thesis. We include the conclusions retrieved by the multiple research lines covered in the chapters. We also illustrate lines for future research.

1.4   Definitions and Notation

Chapter 2

Foundations and Concepts

This chapter presents...

2.1 Introduction

2.2 Kernels

2.2.1 Motivation and Definition

2.2.2 Reproducing Kernel Hilbert Spaces

2.2.3 Examples and Properties

2.3 Risk Functions and Regularization

2.3.1 Empirical and Expected Risk

2.3.2 Regularized Risk Functional

2.3.3 Representer Theorem

2.4 Optimization

2.4.1 Convex Optimization

2.4.2 Unconstrained Problems

2.4.3 Constrained Problems

2.5 Statistical Learning

2.5.1 Uniform Convergence and Consistency

2.5.2 VC dimension and Structural Learning

2.6 Support Vector Machines

2.6.1 Linearly Separable Case

2.6.2 Non-Linearly Separable Case

2.6.3 Kernel Extension

2.6.4 SVM properties

2.6.5 Connection with Structural Learning

2.6.6 SVM Variants

2.7 Conclusions

In this chapter, we covered...

Chapter 3

Multi-Task Learning

This chapter presents. . .

3.1 Introduction

3.2 Why does Multi-Task Learning work?

Typically in Machine Learning the goal is to find the candidate from a space of hypotheses that minimizes some risk. This risk is the expectation of a loss function over the input space, that can differ depending on what kind of problem we are facing, regression, classification or density estimation. This expectation is taken using an unknown probability distribution that expresses how the data points are distributed in the input space. Then, the best candidate can be selected according to different inductive principles, which define a method of approximating a global function from a training set. The most common principle is Empirical Risk Minimization (ERM), where a training set is sampled from the unknown distribution, and the empirical risk corresponding to this sample is minimized as a proxy of the true expected risk. That is, we select the candidate, from the set of hypotheses, that achieves minimum risk in the training set. Several models use the ERM principle to generalize from data such as the Neural Network (NN) or the Support Vector Machine (SVM), each considering a different space of hypotheses from which to choose a candidate. The definition of such space determines the bias for these problems and its selection is crucial. If it does not contain any good hypothesis, the learner will not be able to learn. Also, if the hypothesis space is too large, the learning process will be more difficult. The best hypothesis space we can provide is the one containing only the optimal hypothesis, but this is equivalent the original problem. In other words, in a single-task scenario there is no difference between learning the optimal hypothesis space (bias learning), and ordinary learning of the optimal hypothesis function.

Instead of this single task scenario, we focus on the situation where we want to solve multiple related tasks, estimating multiple functions. In the case where we have a fixed known number of scenarios, this is called Multi-Task Learning (MTL), while the case where different unknown tasks can be sampled and the goal is to learn a good hypothesis space, it is called Learning to Learn (LTL). In both cases, we need a hypothesis space that contains good solutions for the different tasks.

In this Section we give an overview of some of the theoretical results that set the foundation of MTL and help to understand the advantages it brings. We first review the

work of Baxter (2000), where MTL is tackled from the more general view of LTL. After this, the more MTL-focused works of Ben-David and Borbely (2008) and Ben-David and Schuller (2003) are discussed. These two works help to understand, from the theoretical point of view, how combining multiple tasks can be helpful in the learning process. Finally, other important theoretical results of MTL are surveyed for completeness.

3.2.1 Multi-Task Learning and Learning to Learn

In Baxter (2000) an effort is made to define the concepts needed to construct the theory about inductive bias learning or LTL, which can be seen as a generalization of strict MTL. This is done by defining an environment of tasks and extending the work of Vapnik (2000), which defines the capacity of a space of hypothesis; in his work, Baxter defines the capacity of a family of spaces of hypothesis.

Before presenting the concepts defined for Bias Learning, and to establish an analogy to those of ordinary learning, we briefly review some statistical learning concepts.

Ordinary Learning

In the ordinary statistical learning a theoretical scenario with the following elements is used:

- an *input space* $\mathcal{X}$ and an *output space* $\mathcal{Y}$;

- a *probability distribution* P, which is unknown, defined over $\mathcal{X} \times \mathcal{Y}$;

- a *loss function* $\ell(\cdot, \cdot) : \mathcal{Y} \times \mathcal{Y} \to \mathbb{R}$; and

- a *hypothesis space* $\mathcal{H} = \{h(\cdot, \alpha), \alpha \in A \subset \mathbb{R}^m\}$ with hypothesis $h(\cdot, \alpha) : \mathcal{X} \to \mathcal{Y}$.

The goal for the learner is to select a hypothesis $h(\cdot, \alpha) \in \mathcal{H}$, or equivalently $\alpha \in A$, that minimizes the expected risk

$$R_P(h(\cdot, \alpha)) = \int_{\mathcal{X} \times \mathcal{Y}} \ell(h(x, \alpha), y) \, dP(x, y).$$

The distribution P is unknown, but we have a training set $z = \{(x_1, y_1), \ldots, (x_n, y_n)\}$ of samples drawn from P. Then, the approach is to apply the ERM inductive principle, minimizing the empirical risk

$$\hat{R}_z(h(\cdot, \alpha)) = \frac{1}{n} \sum_{i=1}^{n} \ell(h(x_i, \alpha), y_i).$$

Thus, a learner $\mathcal{A}$ maps the set of training samples to a set of hypotheses:

$$\mathcal{A} : (\mathcal{X} \times \mathcal{Y})^n \to \mathcal{H}.$$

Although $\hat{R}_z$ is an unbiased estimator of R_P, it has been shown (Vapnik, 2000) that this approach, despite being the most evident one, is not the best principle that can be followed. This has to do with two facts: the first one is that the unbiased property is an asymptotical one, the second one has to do with overfitting. Vapnik answers to the question of what can be said about R_P when $h(\cdot, \alpha^*)$ minimizes $\hat{R}_z$, and, moreover, his results are valid also for small number of training samples n. More specifically, Vapnik sets the sufficient and necessary conditions for the consistency of an inductive learning

process. He also defines the capacity of a hypothesis space and use it to derive bounds on the rate of this convergence for any $\alpha \in A$ and, more importantly, bounds on the difference $\inf_{\alpha \in A} \hat{R}_z(h(\cdot, \alpha)) - \inf_{\alpha \in A} R_P(h(\cdot, \alpha))$. Under some general conditions, he proves that

$$\inf_{\alpha \in A} \hat{R}_z(h(\cdot, \alpha)) - \inf_{\alpha \in A} R_P(h(\cdot, \alpha)) \leq B(\text{VCdim}(\mathcal{H})/n), \tag{3.1}$$

where B is some non-decreasing function and $\text{VCdim}(\mathcal{H})$ is the capacity of the space $\mathcal{H}$, also named the VC-dimension $\mathcal{H}$. This means that the generalization ability of a learning process can be controlled in terms of two factors:

- The number of training samples n. A greater number of training samples ensures a better generalization of the learning process. This looks intuitive and could be already inferred from the asymptotical properties.

- The VC-dimension $\text{VCdim}(\mathcal{H})$ of the hypothesis space $\mathcal{H}$, which is desirable to be small. This term is not intuitive and is the most important term in Vapnik theory.

The VC-dimension measures the capacity of a set of hypotheses $\mathcal{H}$. If the capacity of the set $\mathcal{H}$ is too large, we may find a hypothesis $h(x, \alpha^*)$ that minimizes $\hat{R}_z$ but does not generalize well and therefore, does not minimize R_P. This is the overfitting problem. On the other side, if we use a simple $\mathcal{H}$, with low capacity, we could be in a situation where there is not a good hypothesis $h(x, \alpha) \in \mathcal{H}$, so the empirical risk $\inf_{\alpha \in A} R_P$ is too large. This is the underfitting problem.

Bias Learning: Concept and Components

In Baxter (2000) the goal is not to learn the optimal hypothesis $h(\cdot, \alpha^*)$ from a fixed space $\mathcal{H}$ but to learn a good space $\mathcal{H}$ from which we can obtain an optimal hypothesis in different situations. Two main concepts are defined: the *family of hypothesis spaces* and an *environment* of related tasks. For simplicity we write $h(x)$ instead of $h(x, \alpha)$ and h instead of $h(\cdot, \alpha)$. Using these concepts, the bias learning problem has the following components:

- an *input space* $\mathcal{X}$ and an *output space* $\mathcal{Y}$;

- an *environment* $(\mathcal{P}, Q)$ where $\mathcal{P}$ is a set of distributions P defined over $\mathcal{X} \times \mathcal{Y}$, and we can sample from $\mathcal{P}$ according to a distribution Q;

- a *loss function* $\ell(\cdot, \cdot) : \mathcal{Y} \times \mathcal{Y} \to \mathbb{R}$; and

- a *family of hypothesis spaces* $\mathbb{H} = \{\mathcal{H}_\beta, \beta \in B\}$, where each element $\mathcal{H}_\beta$ is a set of hypotheses and B is a set of parameters.

As in ordinary learning, the goal is to minimize the expected risk, defined as

$$R_Q(\mathcal{H}_\beta) = \int_{\mathcal{P}} \inf_{h \in \mathcal{H}_\beta} R_P(h) dQ(P) = \int_{\mathcal{P}} \inf_{h \in \mathcal{H}_\beta} \int_{\mathcal{X} \times Y} \ell(h(x), y) dP(x, y) dQ(P) \tag{3.2}$$

for all $\beta \in B$. Observe that this risk is not a function of a specific hypothesis but it depends on the selection of a whole hypothesis space $\mathcal{H}_\beta$. Again, we do not know $\mathcal{P}$ nor Q, but we have a training set sampled from the environment $(\mathcal{P}, Q)$, obtained in the following way:

1. Sample T times from Q obtaining $P_1, \ldots, P_T \in \mathcal{P}$

2. For $r = 1, \ldots, T$ sample m_r pairs $z_r = \{(x_1^r, y_1^r), \ldots, (x_m^r, y_m^r)\}$ according to P_r where $(x_i^r, y_i^r) \in X \times Y$.

Although, in a general MTL problem each task can have a different number of patterns m_r, Baxter considers a sample $\boldsymbol{z} = \{(x_i^r, y_i^r), r = 1, \ldots, T, i = 1, \ldots, m\}$, with m examples from the T learning tasks, which can be expressed as

$$
\boldsymbol{z} = \begin{pmatrix} (x_1^1, y_1^1) & \cdots & (x_m^1, y_m^1) \\ \vdots & \ddots & \vdots \\ (x_1^T, y_1^T) & \cdots & (x_m^T, y_m^T) \end{pmatrix}
$$

and is named a (T, m)-sample. Using $\boldsymbol{z}$ we can define the empirical loss as

$$
\hat{R}_{\boldsymbol{z}}(\mathcal{H}_\beta) = \sum_{r=1}^{T} \inf_{h \in \mathcal{H}_\beta} \hat{R}_{z_r}(h) = \sum_{r=1}^{T} \inf_{h \in \mathcal{H}_\beta} \sum_{i=1}^{m} \ell(h(x_i^r), y_i^r), \tag{3.3}
$$

which is an average of the empirical losses of each task. Note, however, that in the case of the bias learner this estimate is biased, since $R_{P_r}(h)$ does not coincide with $\hat{R}_{z_r}(h)$. Putting all together, a bias learner $\mathcal{A}$ maps the set of all (T, m)-samples to a family of hypothesis spaces:

$$
\mathcal{A} : (\mathcal{X} \times \mathcal{Y})^{(T,m)} \to \mathbb{H}.
$$

To follow an analogous path to that of ordinary learning, the milestones in bias learning theory should include:

- Checking the consistency of the Bias Learning methods, i.e. proving that $\hat{R}_{\boldsymbol{z}}(\mathcal{H}_\beta)$ converges uniformly in probability to $R_Q(\mathcal{H}_\beta)$.

- Defining a notion of capacity for a hypothesis space families $\mathbb{H}$.

- Finding a bound of $\hat{R}_{\boldsymbol{z}}(\mathcal{H}_\beta) - R_Q(\mathcal{H}_\beta)$ for any β using the capacity of the hypothesis space family. If possible, finding also a bound for $\inf_{\beta \in B} \hat{R}_{\boldsymbol{z}}(\mathcal{H}_\beta) - \inf_{\beta \in B} R_Q(\mathcal{H}_\beta)$.

To try to achieve these goals some previous definitions are needed. From this point, since any $\mathcal{H}$ is defined by a $\beta \in B$, we omit β and write just $\mathcal{H}$ for simplicity.

Bias Learning: Capacities and Uniform Convergence

In first place, a *sample-driven* pseudo-metric of $(T, 1)$-empirical risks is defined. Consider a sequence of T probabilities $\boldsymbol{P} = (P_1, \ldots, P_T)$ sampled from $\mathcal{P}$ according to the distribution Q.

Definition 3.1 (*sample-driven* pseudometric)**.** Given a $(T, 1)$-sample

$$
\left\{ (x_1^1, y_1^1), (x_1^2, y_1^2), \ldots, (x_1^T, y_1^T) \right\},
$$

consider the set of sequences of T hypotheses

$$
\mathcal{H}^T = \{(h_1, \ldots, h_T), h_1, \ldots, h_T \in \mathcal{H}\}.
$$

For each such sequence $\mathcal{H}^T$, we can define then the set of $(T, 1)$-empirical risks, with one sample per task, as

$$\mathcal{H}_\ell^T = \left\{ h_\ell^T(x_1^1, y_1^1, \ldots, x_1^T, y_1^T) = \sum_{r=1}^{T} \ell(h_r(x_1^r), y_1^r); \ h_1, \ldots, h_T \in \mathcal{H} \right\}$$

Then, the family of sets of $(T, 1)$-risks is $\mathbb{H}_\ell^T = \bigcup_{\mathcal{H} \in \mathbb{H}} \mathcal{H}_\ell^T$. For any pair $h_\ell^T, \widetilde{h_\ell^T} \in \mathbb{H}_\ell^T$, we define

$$d_{\boldsymbol{P}}(h_\ell^T, \widetilde{h_\ell^T}) = \int_{(\mathcal{X} \times \mathcal{Y})^T} \left| h_\ell^T(x_1^1, y_1^1, \ldots, x_1^T, y_1^T) - \widetilde{h_\ell^T}(x_1^1, y_1^1, \ldots, x_1^T, y_1^T) \right|$$
$$dP_1(x_1, y_1) \ldots dP_T(x_T, y_T)$$

as a pseudometric in $\mathbb{H}_\ell^T$.

Then, a *distribution-driven* pseudo-metric is defined.

Definition 3.2 (*distribution-driven* pseudo-metric)**.** Given a distribution P on $\mathcal{X} \times \mathcal{Y}$. Consider the set of infimum expected risk for each $\mathcal{H}$:

$$h_\ell^* = \inf_{h \in \mathcal{H}} R_P(h).$$

The family of such sets is defined as $\mathbb{H}_\ell^* = \{h_\ell^*, \mathcal{H} \in \mathbb{H}\}$. Then, for $h_\ell^*, \widetilde{h_\ell^*} \in \mathbb{H}_\ell^*$,

$$d_Q(h_\ell^*, \widetilde{h_\ell^*}) = \int_{\mathcal{P}} \left| h_\ell^* - \widetilde{h_\ell^*} \right| dQ$$

is a pseudometric in $\mathbb{H}_\ell^*$.

With these two pseudo-metrics, two capacities for families of hypothesis spaces are defined. For that the definition of ϵ-cover is needed. Since we assume that the same loss function ℓ is used for both definitions, we will rename the families $\mathbb{H}_\ell^*, \mathbb{H}_\ell^T$ as $\mathbb{H}^*, \mathbb{H}^T$.

Definition 3.3 (ϵ-cover)**.** Given a pseudo-metric d_S in a space $\mathcal{S}$, a set of l elements $s_1, \ldots, s_l \in \mathcal{S}$ is an ϵ-cover of $\mathcal{S}$ if $\forall s \in S$, $d_S(s, s_i) \leq \epsilon$ for some $i = 1, \ldots, l$.

Let $\mathcal{R}(\epsilon, \mathcal{S}, d_S)$ denote the size of the smallest ϵ-cover. Then, we can define the following capacities of a family space $\mathbb{H}$:

- The *sample-driven capacity* $C\left(\epsilon, \mathbb{H}^T\right) = \sup_{\boldsymbol{P}} \mathcal{R}(\epsilon, \mathbb{H}^T, d_{\boldsymbol{P}})$.

- The *distribution-driven capacity* $C\left(\epsilon, \mathbb{H}^*\right) = \sup_Q \mathcal{R}(\epsilon, \mathbb{H}^*, d_Q)$.

Using these capacities, the convergence (uniformly over all $\mathcal{H} \in \mathbb{H}$) of bias learners can be proved (Baxter, 2000, Theorem 2). Moreover, given $\epsilon > 0$ and $0 < \nu < 1$, the bias expected risk is bounded as

$$\hat{R}_{\boldsymbol{z}}(\mathcal{H}) \leq R_Q(\mathcal{H}) + \epsilon$$

with probability $1 - \eta$, given sufficiently large T and m, namely

$$T \geq \frac{288}{\epsilon^2} \log \frac{8C\left(\frac{\epsilon}{48}, \mathbb{H}^*\right)}{\eta}, \ m \geq \max\left(\frac{288}{T\epsilon^2} \log \frac{8C\left(\frac{\epsilon}{48}, \mathbb{H}^T\right)}{\eta}, \frac{18}{\epsilon^2} \right).$$

Observe that the bound for T depends on the *distribution-driven* definition, while the one for m depends on the *sample-driven* definition. Also, as intuitively expected, the bound for m is inversely proportional to T, so the more tasks there are, the less data is necessary in each task.

Multi-Task Learning as a special case of Bias Learning

The previous result is for pure Bias Learning, where we have a $(\mathcal{P}, Q)$-environment of tasks. In MTL, we have a fixed number of tasks T and a fixed sequence of distributions $\boldsymbol{P} = (P_1, \ldots, P_T)$, where P_i is a distribution over $(\mathcal{X} \times \mathcal{Y})^m$. The goal is not learning a hypothesis space $\mathcal{H}$ but a sequence of hypothesis $\boldsymbol{h} = (h_1, \ldots, h_T)$, $h_1, \ldots, h_T \in \mathcal{H}$. Thus, the Multi-Task expected risk is

$$R_{\boldsymbol{P}}(\boldsymbol{h}) = \sum_{r=1}^{T} R_{P_r}(h_r) = \sum_{r=1}^{T} \int_{\mathcal{X} \times Y} \ell(h_r(x), y) dP_r(x, y), \tag{3.4}$$

and the empirical risk is defined as

$$\hat{R}_{\boldsymbol{z}}(\boldsymbol{h}) = \sum_{r=1}^{T} \hat{R}_{z_r}(h_r) = \sum_{r=1}^{T} \sum_{i=1}^{m} \ell(h_r(x_i^r), y_i^r). \tag{3.5}$$

Again, for $\epsilon > 0$ and $0 < \nu < 1$, a similar result to that of Bias Learning is given for MTL (Baxter, 2000, Theorem 4):

$$\hat{R}_{\boldsymbol{z}}(\boldsymbol{h}) \le R_{\boldsymbol{P}}(\boldsymbol{h}) + \epsilon$$

with probability $1 - \eta$ given that the number of samples per task

$$m \ge \max \left(\frac{64}{T\epsilon^2} \log \frac{4C\left(\frac{\epsilon}{16}, \mathbb{H}^T\right)}{\eta}, \frac{16}{\epsilon^2} \right).$$

Observe that we do not need the *distribution-driven* capacity in this case, just the *sample-driven* capacity, that is the one bounding the number of samples per task.

Feature Learning

Feature Learning is a common way to encode bias. The most popular example are Neural Networks, where all the hidden layers can be seen as a Feature Learning engine that learns a mapping from the original space $\mathcal{X}$ to a space with "strong" features $\mathcal{V}$. In general, a set of "strong" feature maps is defined as $\mathcal{F} = \{f, f : \mathcal{X} \to \mathcal{V}\}$. Using these features, functions $g \in \mathcal{G}$ (which are tipically simple) are built: $\mathcal{X} \to_f \mathcal{V} \to_g \mathcal{Y}$. Thus, for each map f, the hypothesis space can be expressed as $\mathcal{H}_f = \{h = g \circ f, g \in \mathcal{G}\}$, and the family of hypothesis spaces is $\mathbb{H} = \{\mathcal{H}_f, f \in \mathcal{F}\}$. Now, the Bias Learning problem is the problem of finding a good mapping f. It is proved (Baxter, 2000, Theorem 6) that in the Feature Learning case the capacities of $\mathbb{H}$ can be bounded by the capacities of $\mathcal{F}$ and $\mathcal{G}$ as

$$C\left(\epsilon, \mathbb{H}^T\right) \le C\left(\epsilon_1, \mathcal{G}\right)^T C_{\mathcal{G}_\ell}(\epsilon_2, \mathcal{F}),$$
$$C\left(\epsilon, \mathbb{H}^*\right) \le C_{\mathcal{G}_\ell}(\epsilon, \mathcal{F})$$

with $\epsilon = \epsilon_1 + \epsilon_2$. Here, $C_{\mathcal{G}_\ell}(\epsilon, \mathcal{F})$ is defined as $C_{\mathcal{G}_\ell}(\epsilon, \mathcal{F}) = \sup_P \mathcal{N}(\epsilon, \mathcal{F}, d_{[P,\mathcal{G}_\ell]})$, where

$$d_{[P,\mathcal{G}_\ell]}(f, f') = \int_{\mathcal{X} \times \mathcal{Y}} \sup_{g \in \mathcal{G}} \left| \ell\left(g \circ f(x), y\right) - \ell\left(g \circ f'(x), y\right) \right| dP(x, y)$$

is a pseudo-metric. Using these results alongside those presented for Bias Learning is useful to establish bounds for Feature Learning models like Neural Networks.

Generalized VC-Dimension for MTL

The results presented until now rely on the concepts of two capacities of a family of hypothesis spaces $\mathbb{H}$ to establish bounds in the difference $\hat{R}_{\boldsymbol{z}}(\boldsymbol{h}) - R_Q(\boldsymbol{h})$, that is, the probability of deviations between the empirical and expected risks for a given hypothesis sequence. However, it would be more useful to find some result concerning the best empirical error and the best expected error. To achieve this, a generalized VC-dimension is developed in Baxter (2000) for MTL with Boolean hypotheses, whose image is in $\{0, 1\}$.

Definition 3.4. Let $\mathcal{H}$ be a space of Boolean functions and $\mathbb{H}$ a Boolean hypothesis space family. Denote the set of $T \times m$ matrices in $\mathcal{X}$ as $\mathcal{X}^{T \times m}$. For each $X \in \mathcal{X}^{T \times m}$ and each $\mathcal{H} \in \mathbb{H}$ define the set of binary $T \times m$ matrices

$$\mathcal{H}_{|X} = \left\{ \begin{pmatrix} h_1(x_1^1) & \dots & h_1(x_m^1) \\ \vdots & \ddots & \vdots \\ h_T(x_1^T) & \dots & h_T(x_m^T) \end{pmatrix}, h_1, \dots, h_T \in \mathcal{H} \right\},$$

and the corresponding family of such sets as

$$\mathbb{H}_{|X} = \bigcup_{\mathcal{H} \in \mathbb{H}} \mathcal{H}_{|X}.$$

For each $T, m \geq 0$ define the number of binary matrices obtainable with $\mathbb{H}$ as

$$\Pi_{\mathbb{H}}(T, m) = \max_{X \in \mathcal{X}^{T \times m}} \left| \mathbb{H}_{|X} \right|,$$

where $|A|$ is the size of the set A. Note that $\Pi_{\mathbb{H}}(T, m) \leq 2^{Tm}$ and if $\Pi_{\mathbb{H}}(T, m) = 2^{Tm}$ we say that $\mathbb{H}$ shatters $\mathcal{X}^{T \times m}$. For each $T > 0$, the generalized VC-dimension is defined as

$$d_{\mathbb{H}}(T) = \max_{m : \Pi_{\mathbb{H}}(T, m) = 2^{Tm}} m.$$

Also define

$$\overline{d}(\mathbb{H}) = \text{VCdim}\left(\bigcup_{\mathcal{H} \in \mathbb{H}} \mathcal{H} \right),$$

$$\underline{d}(\mathbb{H}) = \max_{\mathcal{H} \in \mathbb{H}} \text{VCdim}\left(\mathcal{H} \right).$$

Then, it can be shown that

$$d_{\mathbb{H}}(T) \geq \max\left(\left\lfloor \frac{\overline{d}(\mathbb{H})}{T} \right\rfloor, \underline{d}(\mathbb{H}) \right).$$

where it can be observed that

$$\bar{d}(\mathbb{H}) \geq d_{\mathbb{H}}(T) \geq \underline{d}(\mathbb{H}). \tag{3.6}$$

Proof. Let split the proof in three inequalities:

- $d_{\mathbb{H}}(T) \geq \underline{d}(\mathbb{H})$
  Let $\mathcal{H}^{\max}$ be such that $\mathrm{VCdim}\,(\mathcal{H}^{\max}) = \underline{d}(\mathbb{H}) = \max_{\mathcal{H} \in \mathbb{H}} \mathrm{VCdim}\,(\mathcal{H})$. Then, for all $T, m \geq 0$

$$\Pi_{\mathbb{H}}(T, m) = \max_{X \in \mathcal{X}^{T \times m}} \left|\mathbb{H}_{|X}\right| \geq \max_{X \in \mathcal{X}^{T \times m}} \left|\{\mathcal{H}^{\max}\}_{|X}\right| = \Pi_{\{\mathcal{H}^{\max}\}}(T, m).$$

  Since $\left\{m : \Pi_{\{\mathcal{H}^{\max}\}}(T, m) = 2^{Tm}\right\} \subset \left\{m : \Pi_{\mathbb{H}}(T, m) = 2^{Tm}\right\}$,

$$d_{\mathbb{H}}(T) = \max_{m : \Pi_{\mathbb{H}}(T,m)=2^{Tm}} m \geq \max_{m : \Pi_{\{\mathcal{H}^{\max}\}}(T,m)=2^{Tm}} m \geq \underline{d}(\mathbb{H}).$$

- $d_{\mathbb{H}}(T) \geq \left\lfloor \frac{\bar{d}(\mathbb{H})}{T} \right\rfloor$
  If $\mathcal{X}^{\bar{d}(\mathbb{H})}$ is shattered by $\bigcup_{\mathcal{H} \in \mathbb{H}} \mathcal{H}$, then the space of $T \times \left\lfloor \frac{\bar{d}(\mathbb{H})}{T} \right\rfloor$ matrices X, $\mathcal{X}^{T \times \bar{d}(\mathbb{H})}$, is shattered by $\bigcup_{\mathcal{H} \in \mathbb{H}} \mathcal{H}$ and therefore $d_{\mathbb{H}}(T) = \max_{m : \Pi_{\mathbb{H}}(T,m)=2^{Tm}} m \geq \left\lfloor \frac{\bar{d}(\mathbb{H})}{T} \right\rfloor$.

- $d_{\mathbb{H}}(T) \leq \bar{d}(\mathbb{H})$
  The space $\mathcal{X}^{T \times d_{\mathbb{H}}(T)}$ is shattered by $\mathbb{H}$. Taking, without loss of generality, the first row of such matrices; then, the space of sequences with length $d_{\mathbb{H}}(T)$ is shattered by $\bigcup_{\mathcal{H} \in \mathbb{H}}$, thus, $\bar{d}(\mathbb{H}) \geq d_{\mathbb{H}}(T)$.

$\square$

Now we can present the relevant result expressed in (Baxter, 2000, Corollary 13).

Theorem 3.5. *Given a sequence $\boldsymbol{P} = (P_1, \ldots, P_T)$ on $(\mathcal{X} \times \{0,1\})^T$, and a sample $\boldsymbol{z}$ from this distribution. Consider also a sequence $\boldsymbol{h} = (h_1, \ldots, h_T)$ of Boolean hypothesis $h_i \in \mathcal{H}$. Then for every $\epsilon > 0$ and $0 < \nu < 1$*

$$\left|R_{\boldsymbol{P}}(\boldsymbol{h}) - \hat{R}_{\boldsymbol{z}}(\boldsymbol{h})\right| \leq \epsilon,$$

with probability $1 - \eta$ given that the number of samples per task verifies

$$m \geq \frac{88}{\epsilon^2} \left[2d_{\mathbb{H}}(T) \log \frac{22}{\epsilon} + \frac{1}{T} \log \frac{4}{\eta} \right]. \tag{3.7}$$

Here, since $d_{\mathbb{H}}(T) \geq d_{\mathbb{H}}(T+1)$, it is easy to see that as the number of task T increases, the number of examples needed per task can decrease. Moreover, as shown in (Baxter, 2000, Theorem 14), if this bound on m is not fulfilled, then, for any $\epsilon > 0$, we can always find a sequence of distributions $\boldsymbol{P}$ such that

$$\inf_{h \in \mathcal{H}} \hat{R}_{\boldsymbol{z}}(\boldsymbol{h}) > \inf_{h \in \mathcal{H}} R_{\boldsymbol{P}}(\boldsymbol{h}) + \epsilon.$$

With this results we can see that the condition (3.7) has some important properties:

- It is a computable bound, given that we know how to compute $d_{\mathbb{H}}(T)$.

- It provides a sufficient condition for the uniform convergence (in probability) of the empirical risk to the expected risk.

- It provides a necessary condition for the consistency of Multi-Task Learners, i.e. uniform convergence of the best empirical risk to the best expected risk.

3.2.2 Learning with Related Tasks

Using the work of Baxter (2000) as the foundation, several important notions and results are presented in Ben-David and Borbely (2008) for Boolean hypothesis functions defined over $\mathcal{X} \times \{0,1\}$. One of the main contributions of this work is a notion of task relatedness. In Baxter (2000) the tasks are related by sharing a common inductive bias that can be learned. In Ben-David and Borbely (2008) a precise mathematical definition for task relatedness is given. The other important contribution is the focus on the individual risk of each task. In Baxter (2000) all the results are given for the Multi-Task empirical and expected risks, which are an average of the risks of each task. However, bounding this average does not establish a sharp bound of the risk of each particular task. This is specially relevant if we are in a Transfer Learning scenario, where there is a target task that we want to solve and the remaining tasks can be seen as an aid to improve the performance in the target.

A Notion of Task Relatedness: $\mathcal{F}$-Related Tasks

The main concept for the theory developed in Ben-David and Borbely (2008) is a set of $\mathcal{F}$ of transformations $f : \mathcal{X} \to \mathcal{X}$. Given a probability distribution P over $\mathcal{X} \times \{0,1\}$, a set of tasks with distributions $P_1, \ldots, P_T$ are $\mathcal{F}$-related if, for each task there exists some $f_i \in \mathcal{F}$ such that $P_i = f_i(P)$.

Definition 3.6 ($\mathcal{F}$-related task)**.** Consider a measurable space $(\mathcal{X}, \mathcal{A})$ and the corresponding measurable product space $(\mathcal{X} \times \{0,1\}, \mathcal{A} \times \wp(\{0,1\}))$, where $\wp(\Omega)$ is the powerset of set Ω. Consider P a probability distribution over this product space and a function $f : \mathcal{X} \to \mathcal{X}$, then we define the distribution $f[P]$ such that for any $S \in \mathcal{A}$,

$$f[P](S) = P(\{(f(x), b), (x, b) \in S\}).$$

Let $\mathcal{F}$ be a set of transformations $f : \mathcal{X} \to \mathcal{X}$ that is a group under function composition, and let P_1, P_2 be distributions over $(\mathcal{X} \times \{0,1\}, \mathcal{A} \times \wp(\{0,1\}))$, then the distributions P_1, P_2 are $\mathcal{F}$-related if $f[P_1] = P_2$ or $f[P_2] = P_1$ for some $f \in \mathcal{F}$.

This notion establishes a clear definition of related tasks but we are interested in how a learner can use this relatedness to improve the learning process. For that, considering that $\mathcal{F}$ is a group under function composition, we regard at the action of the group $\mathcal{F}$ over the set of hypotheses $\mathcal{H}$. This action defines the following equivalence relation in $\mathcal{H}$:

$$h_1 \sim_{\mathcal{F}} h_2 \iff \exists f \in \mathcal{F}, h_1 \circ f = h_2.$$

This equivalence relation defines equivalence classes $[h]$, that is let $h' \in \mathcal{H}$ be an hypothesis, then $h' \in [h]$ iff $h' \sim_{\mathcal{F}} h$. We consider the quotient space

$$\mathcal{H}_{\mathcal{F}} = \mathcal{H}/\sim_{\mathcal{F}} = \{[h], h \in \mathcal{H}\}.$$

It is important to observe that $\mathcal{H}_{\mathcal{F}} = \mathbb{H}'$ is a hypothesis space family, since it is a set of equivalence classes $[h] = \mathcal{H}'$, which are sets of hypotheses.

The Multi-Task Empirical Risk Minimization

This equivalence classes are useful to divide the learning process in two stages; this is called the *Multi-Task ERM*. Consider the samples $z_1, \ldots, z_T$ from T different tasks; then

1. Select the best hypothesis class $[h^{\mathcal{F}}] \in \mathcal{H}_{\mathcal{F}}$:

$$[h^{\mathcal{F}}] = \min_{[h] \in \mathcal{H}_{\mathcal{F}}} \inf_{h_1, \ldots, h_T \in [h]} \frac{1}{T} \sum_{r=1}^{T} \hat{R}_{z_i}(h_i),$$

2. Select the best hypothesis $h^{\diamond}$ for the target task (without loss of generality, consider that the target task is the first one):

$$h^{\diamond} = \inf_{h \in [h^{\mathcal{F}}]} \hat{R}_{z_1}(h).$$

For example, consider the handwritten digits recognition problem; we might integrate T different datasets designed in different conditions. Each dataset has been created using certain conditions of light and some specific scanner for getting the images. Even different pens or pencils might be influential in the stroke of the numbers. All these conditions are the $\mathcal{F}$ transformations, and each $f \in \mathcal{F}$ generate a different bias for the dataset. However, there exists a probability for "pure" digits, e.g., the pixels of digit one have a higher probability of being black around a line in the middle of the picture than in the sides. This "pure" probability distribution P_0 and all the distributions $P_1, \ldots, P_T$ from which our datasets have been sampled might be $\mathcal{F}$-related among them and with P_0. If we first determine the $\mathcal{F}$-equivalent class of hypothesis $[h]$ suited for digit recognition in the first stage, that is the step 1 above, then it will be easier to select $h_1, \ldots, h_T \in [h]$ for each dataset in the second one.

Bounds for $\mathcal{F}$-Related Tasks

The results of Theorem 3.5 can be applied to the hypothesis quotient space of equivalent classes $\mathcal{H}_{\mathcal{F}}$. However, the following results is needed first. Let P_1, P_2 be $\mathcal{F}$-related distributions, then the following result can be proved (Ben-David and Borbely, 2008, Lemma 2):

$$\inf_{h \in \mathcal{H}} R_{P_1}(h) = \inf_{h \in \mathcal{H}} R_{P_2}(h). \tag{3.8}$$

This indicates that the the expected risk is invariant under transformations of $\mathcal{F}$. Now, one of the main results in Baxter (2000, Theorem 2) can be given.

Theorem 3.7. *Let $\mathcal{F}$ be a set of transformations $f : \mathcal{X} \to \mathcal{X}$ that is a group under function composition. Let $\mathcal{H}$ be a hypothesis space so that $\mathcal{F}$ acts as a group over $\mathcal{H}$, and consider the quotient space $\mathcal{H}_{\mathcal{F}} = \{[h], h \in \mathcal{H}\}$. Consider $\boldsymbol{P} = (P_1, \ldots, P_T)$ a sequence of $\mathcal{F}$-related distributions over $\mathcal{X} \times \{0, 1\}$, and $\boldsymbol{z} = (z_1, \ldots, z_T)$ the corresponding sequence of samples where z_i is sampled using P_i, then for every $[h] \in \mathcal{H}_{\mathcal{F}}$ and $\epsilon, \nu > 0$*

$$\left| \inf_{h_1, \ldots, h_T \in [h]} \frac{1}{T} \sum_{r=1}^{T} \hat{R}_{z_r}(h_r) - \inf_{h' \in [h]} R_{P_1}(h') \right| \leq \epsilon$$

with probability greater than η if the number of samples from each distribution satisfies

$$|z_i| \geq \frac{88}{\epsilon^2} \left[2d_{\mathcal{H}_{\mathcal{F}}}(T) \log \frac{22}{\epsilon} + \frac{1}{T} \log \frac{4}{\eta} \right]. \tag{3.9}$$

Note that, in contrast to Theorem 3.5, this result bounds the expected risk of a single task, not the average risk. This is the consequence of applying Theorem 3.5 and substituting the average empirical error using the result from (3.8). Also observe that here the hypothesis space family used is the quotient space $\mathcal{H}_{\mathcal{F}}$, and the VC-dimension of such family, namely $d_{\mathcal{H}_{\mathcal{F}}}(T)$, is used. Using this result, a bound for learners using the Multi-Task ERM principle is given (Ben-David and Borbely, 2008, Theorem 3)

Theorem 3.8. *Consider $\mathcal{F}$ and $\mathcal{H}$ as in the previous theorem. Consider also the previous sequence of distributions $(P_1, \ldots, P_T)$ and corresponding samples $(z_1, \ldots, z_T)$. Consider $\underline{d}(\mathcal{H}_{\mathcal{F}}) = \max_{h \in \mathcal{H}} VCdim([h])$. Let $h^\diamond$ be the hypothesis selected using the Multi-Task ERM principle. Then for every $\epsilon_1, \epsilon_2 > 0$*

$$\hat{R}_{z_1}(h^\diamond) - \inf_{h' \in \mathcal{H}} R_{P_1}(h') \leq 2(\epsilon_1 + \epsilon_2)$$

with probability greater than η if

$$|z_1| \geq \frac{64}{\epsilon^2} \left[2\underline{d}(\mathcal{H}_{\mathcal{F}}) \log \frac{12}{\epsilon} + \frac{1}{T} \log \frac{8}{\eta} \right], \tag{3.10}$$

and for $i \neq 1$

$$|z_i| \geq \frac{88}{\epsilon^2} \left[2d_{\mathcal{H}_{\mathcal{F}}}(T) \log \frac{22}{\epsilon} + \frac{1}{T} \log \frac{8}{\eta} \right]. \tag{3.11}$$

The idea of the proof of this theorem helps to understand how using different tasks can help to improve the performance in the target task. Consider $h^* = \inf_{h \in \mathcal{H}} R_{P_1}(h)$ the best hypothesis for the P_1 distribution. In the first stage of Multi-Task ERM principle, we select the hypothesis class $[h^{\mathcal{F}}]$ that minimizes $\inf_{\boldsymbol{h} \in [h]} R_{\boldsymbol{P}}(\boldsymbol{h})$ where $\boldsymbol{h}$ is a sequence of hypothesis of $\mathcal{H}_{\mathcal{F}}$. According to Theorem 3.7, for $[h^{\mathcal{F}}]$ we have that

$$\inf_{h' \in [h^{\mathcal{F}}]} R_{P_1}(h') \leq \inf_{h_1, \ldots, h_T \in [h^{\mathcal{F}}]} \frac{1}{T} \sum_{r=1}^{T} \hat{R}_{z_r}(h_r) + \epsilon_1.$$

According to Theorem 3.7, for $[h^*]$ we have that

$$\inf_{h_1, \ldots, h_T \in [h^*]} \frac{1}{T} \sum_{r=1}^{T} \hat{R}_{z_r}(h_r) \leq \inf_{h' \in [h^*]} R_{P_1}(h') + \epsilon_1.$$

Using these two inequalities we get

$$\inf_{h' \in [h^{\mathcal{F}}]} R_{P_1}(h') \leq \inf_{h' \in [h^*]} R_{P_1}(h') + 2\epsilon_1$$

under the condition (3.9). This bounds the risk of the hypothesis space given by the equivalence class of $h^{\mathcal{F}}$ and establishes the inequality (3.11).

Once we select $[h^{\mathcal{F}}]$, the second stage is just ERM using this hypothesis space. According to the Vapnik (1982),

$$\inf_{h \in \mathcal{H}} R_{z_1}(h) - \inf_{h \in \mathcal{H}} R_{P_1}(h) \leq \epsilon_2$$

if

$$|z_1| \geq \frac{64}{\epsilon^2} \left[2\text{VCdim}(\mathcal{H}) \log \frac{12}{\epsilon} + \frac{1}{T} \log \frac{8}{\eta} \right].$$

Since the ERM will not use the whole space $\mathcal{H}$ but the subset $[h^{\mathcal{F}}] \subset \mathcal{H}$, and

$$\text{VCdim}\left([h^{\mathcal{F}}]\right) \leq \max_{[h] \in \mathcal{H}_{\mathcal{F}}} \text{VCdim}([h]).$$

and, as defined in Definition 3.4, $\max_{[h] \in \mathcal{H}_{\mathcal{F}}} \text{VCdim}([h]) = \underline{d}(\mathcal{H}_{\mathcal{F}})$; then we can write the inequality (3.10) of the theorem. The advantage of using multiple tasks is then illustrated in this bound and it will be defined by the gap between $\text{VCdim}(\mathcal{H})$ and $\underline{d}(\mathcal{H}_{\mathcal{F}})$. If $\underline{d}(\mathcal{H}_{\mathcal{F}})$ is smaller than $\text{VCdim}(\mathcal{H})$, the number of samples needed to solve the target task will also be smaller. Also, the sample complexity of the rest of tasks is given by $d_{\mathcal{H}_{\mathcal{F}}}(T)$.

That is, MTL allows to select a subset of hypotheses from which a learner can use the ERM principle. In this stage, the sample complexity is controlled by the generalized VC-dimension of the set of equivalent classes of hypotheses. Once the best equivalent class has been selected, the VC-dimension of this subset, compared to the VC-dimension of the whole set of hypotheses, is what marks the difference between Single Task and MTL.

Analysis of generalized VC-dimension with $\mathcal{F}$-related tasks

As we have seen in Theorem 3.8, the VC-dimensions $\text{VCdim}(\mathcal{H}), \underline{d}(\mathcal{H}_{\mathcal{F}})$ and $d_{\mathcal{H}_{\mathcal{F}}}(T)$, defined in Definition 3.4, are crucial for stating the advantage of Multi-Task over Single Task Learning. To understand better how these concepts interact, Ben-David *et al.* give some theoretical results. Recall that, given a hypothesis space $\mathcal{H}$, $\mathcal{H}_{\mathcal{F}}$ is a family of hypothesis spaces composed by the hypothesis spaces $[h], h \in \mathcal{H}$, then

$$d_{\mathcal{H}_{\mathcal{F}}}(T) = \max_{\left\{m, \Pi_{\mathcal{H}_{\mathcal{F}}} = 2^{Tm}\right\}} m,$$

$$\underline{d}(\mathcal{H}_{\mathcal{F}}) = \max_{[h] \in \mathcal{H}_{\mathcal{F}}} \text{VCdim}([h]),$$

$$\overline{d}(\mathcal{H}_{\mathcal{F}}) = \text{VCdim}\left(\bigcup_{[h] \in \mathcal{H}_{\mathcal{F}}} [h] \right) = \text{VCdim}(\mathcal{H}).$$

Using the result from (3.6) we observe that

$$\underline{d}(\mathcal{H}_{\mathcal{F}}) \leq d_{\mathcal{H}_{\mathcal{F}}}(T) \leq \text{VCdim}(\mathcal{H}).$$

That is, the best we can hope when bounding the sample complexity shown in (3.11) of Theorem 3.8 is $\underline{d}(\mathcal{H}_{\mathcal{F}}) = d_{\mathcal{H}_{\mathcal{F}}}(T)$. Ben-David *et al.* give evidence that, with some restrictions on $\mathcal{H}$, this lower bound can be achieved (Ben-David and Borbely, 2008, Theorem 4).

Theorem 3.9. *If the support of h is bounded, i.e. $|\{x \in \mathcal{X}, h(x) = 1\}| < M$, for all $h \in \mathcal{H}$, then there exists T_0 such that for all $T > T_0$*

$$d_{\mathcal{H}_\mathcal{F}}(T) = \underline{d}(\mathcal{H}_\mathcal{F}).$$

Thus, a sufficient condition on the hypothesis space $\mathcal{H}$ to achieve the lowest $d_{\mathcal{H}_\mathcal{F}}(T)$ is a bounded support of any hypothesis. Although this condition may be too restricting, it can also be proved that the upper limit of $d_{\mathcal{H}_\mathcal{F}}(T)$, that is, VCdim $(\mathcal{H})$, under some conditions on $\mathcal{F}$ is not achieved. The following result (Ben-David and Borbely, 2008, Theorem 6) shows this.

Theorem 3.10. *If $\mathcal{F}$ is finite and $\frac{T}{\log(T)} \geq VCdim(\mathcal{H})$, then*

$$d_{\mathcal{H}_\mathcal{F}}(T) \leq 2\log(|\mathcal{F}|)$$

This inequality indicates that, given a finite set of transformation $\mathcal{F}$, there are scenarios when VCdim $(\mathcal{H})$ is arbitrarily large but $d_{\mathcal{H}_\mathcal{F}}(T)$ is bounded, and therefore, the right-hand side of inequality (3.11) is also bounded. That is, the Multi-Task bound, which substitutes VCdim $(\mathcal{H})$ by $d_{\mathcal{H}_\mathcal{F}}(T)$, is a better one in these cases.

3.2.3 Other bounds for Multi-Task Learning

The work of Baxter (2000) set the foundations for the theoretical analysis of MTL and LTL. In this work, an MTL extension to the VC-dimension is given, and it is used to develop some results bounding the difference between the Multi-Task empirical and expected risks

$$\left| R_{\boldsymbol{P}}(\boldsymbol{h}) - \hat{R}_{\boldsymbol{z}}(\boldsymbol{h}) \right|$$

for any sequence of hypothesis $\boldsymbol{h} \in \mathcal{H}^T$, see Theorem 3.5. This is a necessary condition for the consistency of Multi-Task Learners, but not a sufficient one. Then, Ben-David *et al.* define a notion of task relatedness (Ben-David and Borbely, 2008; Ben-David and Schuller, 2003), see Definition 3.6. Using this notion and building an appropiate hypothesis space $\mathcal{H} = [h]$, they bound the *excess risk* of an Empirical Multi-Task Learner, that is, the difference between the best empirical risk, achieved by such Learner, and the best possible expected risk (Theorem 3.8)

$$\left| \inf_{\boldsymbol{h} \in \mathcal{H}^T} R_{\boldsymbol{P}}(\boldsymbol{h}) - \inf_{\boldsymbol{h} \in \mathcal{H}^T} \hat{R}_{\boldsymbol{z}}(\boldsymbol{h}) \right|.$$

Moreover, using this task relatedness definition not only the Multi-Task average excess risk is bounded, but the individual excess risk of each task (Theorem 3.10).

The works discussed until this point use the VC-dimension, and the corresponding extensions to the MTL framework expressed in Definition 3.4, to bound the differences between empirical and expected risks. However in Ando and Zhang (2005), the authors rely on another notion of complexity, the Rademacher Complexity (Bartlett and Mendelson, 2002), which measures how well a family of hypothesis can approximate random noise. The Rademacher complexity, unlike the VC-dimension, is distribution-dependent. That is the VC-dimension only uses properties of the hypothesis space $\mathcal{H}$, while the Rademacher complexity also depends on the data distribution P. Other theoretical works obtain bounds for linear feature extractor methods for MTL, such as Cavallanti et al. (2010); Maurer (2006a,b). In the more general case of LTL, some improved bounds

are found for specific cases like the trace norm regularized MTL models (Maurer et al., 2013). Then, in Maurer et al. (2016) bounds for a wide class of MTL models based on Multi-Task Representation Learning (MTRL) are given, in both an MTL and an LTL setting. These bounds are not dependent on the data dimensions, as other bounds for linear models, and use an approach based on empirical process theory instead of the generalized VC-dimension to derive these bounds.

3.2.4 Learning Under Privileged Information

Another important motivation for MTL can be found in the Learning Under Privileged Information (LUPI) paradigm of Vapnik and Izmailov (2015). The standard ML paradigm tries to find the hypothesis h from a set of hypotheses $\mathcal{H}$ that minimizes the expected risk $\hat{R}_z$ given a set of training samples. Vapnik is one of the main contributors to the theory of statistical learning, see Vapnik (2000). In this theory several important results are provided: necessary and sufficient conditions for the consistency of learning processes and bounds for the rate of convergence, which use the notion of VC-dimension. A new inductive principle, Structural Risk Minimization (SRM), and an algorithm, Support Vector Machine (SVM), that makes use of this notion to improve the learning process are given.

Nowadays learning approaches based on Deep Neural Networks, which are not focused on controlling the capacity of the set of hypotheses, outperform the SVM approaches in many problems. However, these popular Deep Learning approaches require large amounts of data to learn good hypothesis. It is commonly believed that machines need much more samples to learn than humans do. The authors in Vapnik and Izmailov (2015); Vapnik and Vashist (2009) reflect on this belief and states that humans typically learn under the supervision of an Intelligent Teacher. This Teacher shares important knowledge by providing metaphors, examples or clarifications that are helpful for the students.

Privileged Information
The additional knowledge provided by the Teacher is the Privileged Information that is available only during the training stage. To incorporate the concept of Intelligent Teacher in the Machine Learning framework, Vapnik introduces the paradigm of Learning Under Privileged Information (LUPI). In this paradigm, given a set of i.i.d. triplets

$$z = \{(x_1, x_1^*, y_1), \ldots, (x_n, x_n^*, y_n)\}, \ x \in \mathcal{X}, x^* \in \mathcal{X}^*, y \in \mathcal{Y}$$

generated according to an unknown distribution $P(x, x^*, y)$, the goal is to find the hypothesis $h(x, \alpha^*)$ from a set of hypotheses $\mathcal{H} = \{h(x, \alpha), \alpha \in A\}$ that minimizes some expected risk

$$R_P = \int \ell\left(h\left(x, \alpha\right), y\right) dF(x, y).$$

Note that the goal is the same that in the standard paradigm; however with the LUPI approach we are provided additional information, which is available only during the training stage. This additional information is encoded in the elements x^* of a space $\mathcal{X}^*$, which is different from $\mathcal{X}$. The goal of the Teacher is, given a pair (x_i, y_i), to provide an information $x_i^* \in \mathcal{X}^*$ given some probability $P(x^* \mid x)$. That is, the "intelligence" of the Teacher is defined by the choice of the space $\mathcal{X}^*$ and the conditional probability $P(x^* \mid x)$. To understand better this paradigm consider the following example. The goal is to find a decision rule that classifies biopsy images into cancer or non-cancer. Here, $\mathcal{X}$ is the space of images, i.e. the matrix of pixels, for example $[0, 1]^{64 \times 64}$. The label

space is $\mathcal{Y} = \{0, 1\}$. An Intelligent Teacher might provide a student of medicine with commentaries about the images, for example: "There is an area of unusual concentration of cells of Type A." or "There is an aggresive proliferation of cells of Type B". These commentaries are the elements x^* of certain space X^* and the Teacher also chooses the probability $P(x^* \mid x)$, which defines when to express this additional information.

Analysis of convergence rates
To get a better insight of how the Privileged Information can help in the learning process, Vapnik provides a theoretical analysis of its influence on the learning rates. In the standard learning paradigm, how well the expected risk R_P can be bounded is controlled by two factors: the empirical risk $\hat{R}_z$ and the VC-dimension of the set of hypotheses $\mathcal{H}$. In the case of classification, where $\mathcal{Y} = \{-1, 1\}$ and the hinge loss $\ell(h(x, \alpha), y) = \mathbf{1}_{yh(x,\alpha) \leq 0}$, the risks can be expressed as

$$R_P(\alpha) = \int \mathbf{1}_{yh(x,\alpha) \leq 0} dP(x, y) = P(yh(x, \alpha) \leq 0),$$

$$\hat{R}_z(h(\cdot, \alpha)) = \sum_{i=1}^{n} \mathbf{1}_{y_i h(x_i, \alpha) \leq 0}.$$

Recall In (Vapnik, 1982, Theorem 6.8), the following bound for the rate of convergence is given with probability $1 - \eta$:

$$P(yh(x, \alpha) \leq 0) \leq \nu(\alpha) + O\left(\frac{d \log\left(\frac{2n}{d}\right) - \log \eta}{n} \sqrt{\nu(\alpha) \frac{n}{d \log\left(\frac{2n}{d}\right) - \log \eta}} \right).$$

That is, the bound is controlled by the ratio d/n, where d is the $\mathrm{VCdim}(\mathcal{H})$. If this VC-dimension is finite, the bound goes to zero as n grows. However, two different cases can be considered.

Separable case: the training data can be classified in two groups without errors. That is, there exists $\alpha_n \in A$ such that $y_i h(x_i, \alpha_n) > 0$ for $i = 1, \ldots, n$, and thus $\nu(\alpha_n) = 0$. In this case, the following bound for the rate of converge holds

$$P(yh(x, \alpha_n) \leq 0) \leq O\left(\frac{d \log\left(\frac{2n}{d}\right) - \log \eta}{n} \right).$$

Non-Separable case: the training data cannot be classified in two groups without errors. That is, for all $\alpha \in A$, there exists $i = 1, \ldots, n$, such that $y_i h(x_i, \alpha) \leq 0$, and thus $\nu(\alpha) > 0$. In this case, the following bound for the rate of converge holds

$$P(yh(x, \alpha) \leq 0) \leq \nu(\alpha) + O\left(\sqrt{\frac{d \log\left(\frac{2n}{d}\right) - \log \eta}{n}} \right).$$

Note that there is an important difference here in the rate of convergence. The separable case has a convergence rate of d/n, while the non-separable case has a rate of $\sqrt{d/n}$. Vapnik tries to address the question of why there exists such difference.

Oracle SVM

Vapnik tries to answer these question by looking at Support Vector Machines. In the separable case, one has to minimize the functional

$$J(w) = \|w\|^2$$

subject to the constraints

$$y_i\left(\langle w, x_i \rangle + b\right) \geq 1.$$

However, in the non-separable case the functional to minimize is

$$J(w, \xi_1, \dots, \xi_n) = \|w\|^2 + C \sum_{i=1}^{n} \xi_i$$

subject to the constraints

$$y_i\left(\langle w, x_i \rangle + b\right) \geq 1 - \xi_i.$$

That is, in the separable case d parameters (of w) have to be estimated using n examples, while in the non-separable case $d + n$ parameters (considering w and the slack variables $\xi_1, \dots, \xi_n$) have to be estimated with n examples.

The authors wonder what would happen if the parameters $\xi_1, \dots, \xi_n$ were known. In Vapnik and Izmailov (2015) an *Oracle* SVM is considered. Here, the learner (Student) is supplied with a set of triplets

$$(x_1, \xi_1^0, y_1), \dots, (x_n, \xi_n^0, y_n)$$

where $\xi_1^0, \dots, \xi_n^0$ are the slack variables for the best decision rule:

$$\xi_i^0 = \max\left(0, 1 - h\left(x, \alpha_0\right)\right), \ \forall i = 1, \dots, n,$$

where $\alpha_0 = \arg\inf_{\alpha \in A} R_P(h\left(\cdot, \alpha\right))$. An *Oracle* SVM has to minimize the functional

$$J(w) = \|w\|^2$$

subject to the constraints

$$y_i\left(\langle w, x_i \rangle + b\right) \geq 1 - \xi_i^0.$$

Since the slack variables ξ_i^0 are known in advance, it can be shown (Vapnik and Vashist, 2009) that for the *Oracle* SVM the following bound holds

$$P(yh\left(x, \alpha_n\right) \leq 0) \leq P(1 - \xi^0 \leq 0) + O\left(\frac{d \log\left(\frac{2n}{d}\right) - \log \eta}{n}\right),$$

where $P(1 - \xi^0 \leq 0)$ is the probability error of the hypothesis $h\left(x, \alpha_0\right)$. That is, we recover the rate d/n.

From Oracle to Intelligent Teacher

The *Oracle* SVM is a theoretical construct, but we can approximate it by modelling the slack variables with the information provided by the Teacher in the LUPI paradigm. That is, the Teacher defines a space $\mathcal{X}^*$ and a set of functions $\{f^*(x^*, \alpha^*), \alpha^* \in A^*\}$,

then models the slack variables as

$$\xi^* = f^*(x^*, \alpha^*).$$

From the pairs sampled from some unknown distribution, the Teacher also defines the probability $P(x^* \mid x)$ to provide the triplets

$$(x_1, x_1^*, y_1), \ldots, (x_n, x_n^*, y_n).$$

Then, we can consider the problem where the goal is to minimize

$$J(\alpha, \alpha^*) = \sum_{i=1}^{n} \max(0, f^*(x_i^*, \alpha^*))$$

subject to the constraints

$$y_i h(x_i, \alpha) \geq 1 - f^*(x_i^*, \alpha^*).$$

Let $f(x^*, \alpha_n^*)$, $h(x, \alpha_n)$ be the solutions that minimize this empirical risk minimization problem. Then, in Vapnik and Vashist (2009, Proposition 2) the following results for the bound of convergence are given

$$P(h(x, \alpha_n) y \leq 0) \leq P(1 - f^*(x^*, \alpha_n^*) \leq 0) + O\left(\frac{(d + d^*) \log\left(\frac{2n}{(d+d^*)}\right) - \log \eta}{n}\right),$$

where d^* is the VC-dimension of the space of hypothesis $\{f(x, \alpha^*) \in A^*\}$. This result shows that, to maintain the best convergence rate d/n, we need to estimate the $P(1 - f^*(x^*, \alpha_n^*) \leq 0)$. Although this probability is unknown, we can control it. Let α_0^* be the parameter minimizing the expected risk

$$\alpha_0^* = \arg\inf_{\alpha^* \in A^*} \int_{\mathcal{X}^*} \max(0, f^*(x^*, \alpha^*) - 1) dP(x^*),$$

and α_n^* the one minimizing the empirical risk,

$$\alpha_n^* = \arg\inf_{\alpha^* \in A^*} \sum_{i=1}^{n} \max(0, f^*(x_i^*, \alpha^*) - 1).$$

Consider $\{f^*(x^*, \alpha^*), \alpha^* \in A^*\}$ such that $f^*(x^*, \alpha^*) < B, \alpha^* \in A^*$, then

$$\{\max(0, f^*(x^*, \alpha^*) - 1), \alpha^* \in A^*\}$$

is a set of totally bounded non-negative functions, and hence we have the standard bound (Vapnik, 2000),

$$P(1 - f^*(x^*, \alpha_0^*) \leq 0) \leq P(1 - f^*(x^*, \alpha_n^*) \leq 0) + O\left(\sqrt{\frac{d^* \log\left(\frac{2n}{d^*}\right) - \log \eta}{n}}\right),$$

with probability $1 - 2\eta$. That is, to have a rate of d/n for α_n, we need to estimate α_n^*, which has a rate of $\sqrt{d^*/n}$. However, observe that $\mathcal{X}^*$ is the space suggested by the

Teacher, which hopefully has a lower capacity, and thus, the convergence will be faster in this space.

SVM+

Vapnik describes an extension of the SVM that embodies the LUPI paradigm (Vapnik and Izmailov, 2015; Vapnik and Vashist, 2009). Given a set of triplets

$$(x_1, x_1^*, y_1), \ldots, (x_n, x_n^*, y_n),$$

the idea is to model the slack variables of the standard SVM using the elements $x^* \in \mathcal{X}^*$ as

$$\xi(x^*, y) = [y(w^* \phi^*(x^*) + b^*)]_+ = \max\left(y(w^* \phi^*(x^*) + b^*), 0\right).$$

The minimization problem is the following:

$$
\begin{aligned}
&\underset{w,w^*,b,b^*}{\arg\min} \quad C \sum_{i=1}^{n} [y_i(\langle w^*, \phi^*(x_i^*)\rangle + b^*)]_+ + \frac{1}{2}\langle w, w\rangle + \frac{\mu}{2}\langle w^*, w^*\rangle \\
&\text{s.t.} \qquad y_i(\langle w, \phi(x_i)\rangle + b) \geq 1 - [y_i(\langle w^*, \phi^*(x_i^*)\rangle + b^*)]_+ .
\end{aligned}
\tag{3.12}
$$

Here ϕ and ϕ^* are two transformations that can be different. However, note that problem (3.12) is not convex due to the positive part $[]_+$ term in the objective function. Vapnik *et al.* propose a relaxation of this problem to obtain a convex one. The idea is to model the slack variables ξ as

$$\xi(x^*, y) = [y(w^* \phi^*(x^*) + b^*)] + \zeta(x^*, y),$$

where $\zeta(x^*, y) \geq 0$. The minimization problem is then

$$
\begin{aligned}
&\underset{w,w^*,b,b^*,\zeta_i}{\arg\min} \quad C \sum_{i=1}^{n} ([y_i(\langle w^*, \phi^*(x_i^*)\rangle + b^*)] + \zeta_i) + C\Delta \sum_{i=1}^{n} \zeta_i \\
&\qquad\qquad\quad + \frac{1}{2}\langle w, w\rangle + \frac{\mu}{2}\langle w^*, w^*\rangle \\
&\text{s.t.} \qquad y_i(\langle w, \phi(x_i)\rangle + b) \geq 1 - [y_i(\langle w^*, \phi^*(x_i^*)\rangle + b^*) + \zeta_i], \\
&\qquad\qquad y_i(\langle w^*, \phi^*(x_i^*)\rangle + b^*) + \zeta_i \geq 0, \\
&\qquad\qquad \zeta_i \geq 0, \\
&\text{for} \qquad i = 1, \ldots, n,
\end{aligned}
\tag{3.13}
$$

Problem (3.13) is convex and the corresponding dual problem is

$$
\begin{aligned}
&\underset{\alpha_i,\delta_i}{\arg\min} \quad \frac{1}{2}\sum_{i,j=1}^{n} y_i y_j \alpha_i \alpha_j k(x_i, x_j) + \frac{1}{2\mu}\sum_{i,j=1}^{n} y_i y_j (\alpha_i - \delta_i)(\alpha_j - \delta_i) k^*(x_i^*, x_j^*) - \sum_{i=1}^{n} \alpha_i \\
&\text{s.t.} \qquad 0 \leq \delta_i \leq C \\
&\qquad\qquad 0 \leq \alpha_i \leq C + \delta_i, \\
&\qquad\qquad \sum_{i=1}^{n} \delta_i y_i = 0, \ \sum_{i=1}^{n} \alpha_i y_i = 0, \\
&\text{for} \qquad i = 1, \ldots, n.
\end{aligned}
\tag{3.14}
$$

where we use the kernel functions

$$k(x_i, x_j) = \langle \phi(x_i), \phi(x_j) \rangle, \ k^*(x_i^*, x_j^*) = \langle \phi^*(x_i^*), \phi^*(x_j^*) \rangle.$$

We can observe in Problem (3.14) that the LUPI paradigm exerts a similarity control, correcting the similarity in space $\mathcal{X}$ with the similarity in the privileged space $\mathcal{X}^*$. For that reason, $\mathcal{X}$ and $\mathcal{X}^*$ are named Decision Space and Correction Space, respectively.

Connection between SVM+ and MTLSVM

In Liang and Cherkassky (2008) the connection between SVM+ and MTL SVM (MTLSVM) is discussed. The MTLSVM proposed in Liang and Cherkassky (2008) is a MTL model based on the SVM. It solves the primal problem

$$
\begin{aligned}
\underset{w,b,v_r,b_r,\xi_i^r}{\arg\min} \quad & C \sum_{r=1}^{T} \sum_{i=1}^{m_r} \xi_i^r + \frac{1}{2} \langle w, w \rangle + \sum_{r=1}^{T} \frac{\mu}{2} \langle v_r, v_r \rangle \\
\text{s.t.} \quad & y_i^r (\langle w, \phi(x_i^r) \rangle + b + \langle v_r, \phi_r(x_i^r) \rangle + b_r) \geq 1 - \xi_i^r, \\
& \xi_i^r \geq 0, \\
\text{for} \quad & r = 1, \dots, T; \ i = 1, \dots, m_r.
\end{aligned}
\tag{3.15}
$$

Here, a combination of a common model for all tasks

$$\langle w, \phi(x_i) \rangle + b$$

and a task-specific model

$$\langle v_r, \phi_r(x_i^r) \rangle + b_r$$

is used, where the common transformation ϕ and the task-independent ones ϕ_r can be different The dual problem corresponding to (3.15) is

$$
\begin{aligned}
\underset{\alpha_i}{\arg\min} \quad & \frac{1}{2} \sum_{r,s=1}^{T} \sum_{i,j=1}^{m_r,m_s} y_i^r y_j^s \alpha_i^r \alpha_j^s k(x_i^r, x_j^s) + \frac{1}{2\mu} \sum_{r,s=1}^{T} \sum_{i,j=1}^{m_r,m_s} y_i^r y_j^s \alpha_i^r \alpha_j^s \delta_{rs} k_r(x_i^r, x_j^s) \\
& - \sum_{r=1}^{T} \sum_{i=1}^{m_r,m_s} \alpha_i^r \\
\text{s.t.} \quad & 0 \leq \alpha_i^r \leq C, \\
& \sum_{i=1}^{m_r} \alpha_i^r y_i^r = 0, \\
\text{for} \quad & r = 1, \dots, T; \ i = 1, \dots, m_r,
\end{aligned}
\tag{3.16}
$$

where δ_{rs} is the Dirac's delta. In Liang and Cherkassky (2008) some similarities between MTLSVM and SVM+ are pointed out. Problem (3.15) can be regarded as an adaptation of (3.13) to solve MTL problems, where different tasks are incorporated and multiple correcting spaces are defined using the transformations ϕ_r. If we consider the problem (3.15) with a single task, it is a modification of the SVM+ problem (3.13) where the slack variables are modeled as

$$\xi(x, y) = y(w^* \phi^*(x) + b^*).$$

That is, it is a relaxation of the original problem (3.13), where the second constraint to model the positive part of the slack variables disappears. This relaxation gives place to some important differences between both models. Since the auxiliary primal variables ζ_i are no longer required, this is reflected in a simpler dual form (3.16), where only n dual variables have to be estimated, instead of the $2n$ dual variables of (3.14). The Multi-Task part in (3.16) resides in the δ_{rs} function, which makes the correction of similarity only possible between elements of the same task.

A major remark can be made about the differences between MTLSVM and SVM+. The results for the improved rate of convergence with an Intelligent Teacher may not be valid with MTLSVM, since we are not modelling the slack variables ξ adequately. It is still a work in progress to study the rate of convergence of MTLSVM and to establish clearer links with SVM+.

3.3 Kernels for Multi-Task Learning

Kernels are functions that are used as a measure of similarity between data points, and have a notable relevance in ML due to the success of kernel methods, such as the GP or SVM. Moreover, when using kernel methods, the parametric point of view for estimating a function of the primal formulation is changed for a non-parametric one in the dual problem, where the complexity of the model increase with the number of examples.

However, in the case of MTL different functions have to be estimate and the relation between those task-specific functions is desirable to be captured. Using kernels for these goals imposes some new challenges, that can be tackled from different perspectives. One of them is interpreting MTL paradigm can be seen as learning a vector-valued function, thus using vector-valued Reproducing Kernel Hilbert Space (RKHS). We define this and some related contents, and also describe an extension of the Representer Theorem for vector-valued Reproducing Kernel Hilbert Space (RKHS) problems. Also, here we propose a reformulation where tensor product of scalar Reproducing Kernel Hilbert Space (RKHS) is used instead of the vector-valued ones, which leads to some more general results, including another extension of the Representer Theorem. Finally, we take these definitions to show how to use them to solve MTL problems and also give some examples of commonly used Multi-Task kernels.

3.3.1 Vector-Valued Reproducing Kernel Hilbert Spaces

Consider $\mathcal{Y}$ a Hilbert space with inner product $[.,.]_{\mathcal{Y}}$, and $\mathcal{L}(\mathcal{Y})$ the space of linear operators from $\mathcal{Y}$ to $\mathcal{Y}$, then we can study the Hilbert spaces $\mathcal{H}$ of functions

$$
\begin{array}{rlll}
f : & \mathcal{X} & \to & \mathcal{Y} \\
& x & \to & f(x)
\end{array}
,
$$

with inner product $\langle \cdot, \cdot \rangle$. The kernels in such spaces are operator-valued functions $K : \mathcal{X} \times \mathcal{X} \to \mathcal{L}(\mathcal{Y})$. We look at these spaces from three different and equivalent perspectives: continuous evaluation functionals, positive-definite kernels and feature maps.

Continuous evaluation functionals. Consider the vector-valued Hilbert space $\mathcal{H}$ with inner product $\langle \cdot, \cdot \rangle$ of functions defined in $\mathcal{X}$ and values in $\mathcal{Y}$, and the functionals

$$L_{x,y}f = [y, f(x)]_{\mathcal{Y}},$$

$$
\begin{array}{cccccc}
L_{x,y}: & \mathcal{H} & \to & \mathcal{Y} & \to & \mathbb{R} \\
& f & \to & f(x) & \to & [y, f(x)]_{\mathcal{Y}}
\end{array}
.
$$

TODO: Explicar Riesz Representation Theorem (Chapter 2)

If these functionals are continuous, we can apply Riesz Representation theorem. That is, for every $x \in \mathcal{X}, y \in \mathcal{Y}$ we can find an unique $g_{x,y} \in \mathcal{H}$ such that for all $f \in \mathcal{H}$,

$$L_{x,y}f = [y, f(x)]_{\mathcal{Y}} = \langle g_{x,y}, f \rangle_{\mathcal{H}}. \tag{3.17}$$

We can now give the definition of vector-valued Hilbert space from the point of view of continuous functionals Micchelli and Pontil (2005, Definition 2.1).

Definition 3.11 (vector-valued RKHS)**.** We say that $\mathcal{H}$ is a vector-valued RKHS when for any $x \in \mathcal{X}$ and $y \in \mathcal{Y}$, the functional $L_{x,y}f = [y, f(x)]_{\mathcal{Y}}$ is continuous.

Note that this is a definition similar to the scalar case but we use the inner product of $\mathcal{Y}$ to construct the scalar-valued functionals $L_{x,y}$. The price we pay is that it is necessary to express the Riesz representation as dependent of the elements $y \in \mathcal{Y}$. To get rid of this dependence, for every $x \in \mathcal{X}$ we can define the linear operator

$$
\begin{array}{cccc}
g_x: & \mathcal{Y} & \to & \mathcal{H} \\
& y & \to & g_x y = g_{x,y}
\end{array}
.
$$

This operator is well defined because $g_{x,y}$ is unique for every $x \in \mathcal{X}, y \in \mathcal{Y}$ and its linearity is easy to check from the linearity of the inner product $[.,.]_{\mathcal{Y}}$.

Using these results, we can now define the operator

$$
\begin{array}{cccc}
K(x,\hat{x}): & \mathcal{Y} & \to & \mathcal{Y} \\
& y & \to & K(x,\hat{x})y = (g_{\hat{x}}y)(x)
\end{array}
\tag{3.18}
$$

for every $x, \hat{x} \in \mathcal{X}$. Observe that $K(x,\hat{x})$ is linear since $\forall x$, g_x is linear, i.e. $g_x(\lambda_1 y_1 + \lambda_2 y_2) = \lambda_1 g_x(y_1) + \lambda_2 g_x(y_2)$. It is possible then to prove that $K(x,\hat{x})$ is a reproducing kernel in $\mathcal{H}$ as seen (Micchelli and Pontil, 2005, Propositon 2.1). To do that, first we have to define a vector-valued kernel and the corresponding reproducing property.

Definition 3.12 (operator-valued Kernel)**.** If $\mathcal{Y}$ is a finite-dimensional Hilbert space an operator-valued kernel is a function

$$K: \mathcal{X} \times \mathcal{X} \to \mathcal{L}(\mathcal{Y})$$

which is symmetric and positive definite.

For the clarity of the text an operator-valued K will be referred just as kernel unless an explicit distinction is needed.

Definition 3.13 (Reproducing Property of operator-valued operators)**.** If $\mathcal{Y}$ is a finite-dimensional Hilbert space, and $\mathcal{H}$ is a Hilbert space of $\mathcal{Y}$-valued functions with domain in $\mathcal{X}$, a function

$$K: \mathcal{X} \times \mathcal{X} \to \mathcal{L}(\mathcal{Y})$$

has the reproducing property if $\forall x \in \mathcal{X}, y \in \mathcal{Y}, \forall f \in \mathcal{H}$, $[y, f(x)]_{\mathcal{Y}} = \langle K(\cdot, x)y, f \rangle_{\mathcal{H}}$.

A kernel with the reproducing property is also called a reproducing kernel. The next proposition shows how we can build a reproducing kernel for a space $\mathcal{H}$ in which the evaluation functionals are continuous.

Proposition 3.14. *If for every $x, \hat{x} \in \mathcal{X}$, the function $K(x, \hat{x})$ is defined as in Equation (3.18), then the function*

$$K : \mathcal{X} \times \mathcal{X} \to \mathcal{L}(\mathcal{Y})$$

is a reproducing kernel.

Proof. To prove that K is a reproducing kernel we need to check:

1. $K(x, \hat{x})$ is bounded for every $x, \hat{x} \in \mathcal{X}$ so K is well defined.

2. K is symmetric: $K(x, \hat{x})^{\#} = K(\hat{x}, x)$, where $A^{\#} \in \mathcal{L}(\mathcal{Y})$ is the adjoint of $A \in \mathcal{L}(\mathcal{Y})$.

3. K is positive definite: given $n \in \mathbb{N}$, for any $x_1, \ldots, x_n \in \mathcal{X}, y_1, \ldots, y_n \in \mathcal{Y}$,

$$\sum_{i,j=1}^{n} [y_i, K(x_i, x_j) y_j]_{\mathcal{Y}} \geq 0.$$

4. It has the reproducing property: $\forall x \in \mathcal{X}, y \in \mathcal{Y}, \forall f \in \mathcal{H}, [y, f(x)]_{\mathcal{Y}} = \langle K(\cdot, x)y, f \rangle$.

To prove 1 and 2, observe that by applying (3.17) to $f = g_{\hat{x}} \hat{y}$, for every $x \in \mathcal{X}, y \in \mathcal{Y}$ there exists an unique $g_{x,y} = g_x y$ such that

$$[y, (g_{\hat{x}} \hat{y})(x)]_{\mathcal{Y}} = \langle g_x y, g_{\hat{x}} \hat{y} \rangle, \tag{3.19}$$

and combining this result with (3.18) we get

$$[y, K(x, \hat{x}) \hat{y}]_{\mathcal{Y}} = [y, (g_{\hat{x}} \hat{y})(x)]_{\mathcal{Y}} = \langle g_x y, g_{\hat{x}} \hat{y} \rangle.$$

Also, using the definitions,

$$[K(\hat{x}, x)y, \hat{y}]_{\mathcal{Y}} = [(g_x y)(\hat{x}), \hat{y}]_{\mathcal{Y}} = [\hat{y}, (g_x y)(\hat{x})]_{\mathcal{Y}} = \langle g_{\hat{x}} \hat{y}, g_x y \rangle.$$

Since both operators $K(x, \hat{x}), K(\hat{x}, x)$ are linear, by the Uniform Boundness Principle (Akhiezer and Glazman, 1961), $K(x, \hat{x}), K(\hat{x}, x)$ are bounded (hence continuous) and $K(x, \hat{x})^* = K(\hat{x}, x)$.
To prove 3 we write

$$\sum_{i,j=1}^{n} [y_i, K(x_i, x_j) y_j]_{\mathcal{Y}} = \sum_{i,j=1}^{n} \langle g_{x_i} y_i, g_{x_j} y_j \rangle = \left\| \sum_{i=1}^{n} g_{x_i} y_i \right\|^2 \geq 0.$$

Finally, to prove 4 we use that $\forall x \in \mathcal{X}, y \in \mathcal{Y}, \forall f \in \mathcal{H}, \exists g_{x,y} \in \mathcal{H}$ such that

$$[y, f(x)]_{\mathcal{Y}} = \langle g_{x,y}, f \rangle = \langle g_x y, f \rangle = \langle K(\cdot, x)y, f \rangle.$$

$\square$

Positive semi-definite kernels. The second approach changes the point of view. Given a kernel K, the Hilbert space from which K is the reproducing kernel is built. To do this, we use Micchelli and Pontil (2005, Theorem 2.1), which extends the Moore-Aronszanj's Theorem:

Theorem 3.15. *If $K : \mathcal{X} \times \mathcal{X} \to \mathcal{L}(\mathcal{Y})$ is a kernel, then there exists a unique (up to an isometry) RKHS which admits K as its reproducing kernel.*

The proof is similar to that of the Moore-Aronszanj's Theorem, considering the space of the completion of the span of $\{K_x = K(\cdot, x), x \in \mathcal{X}\}$.

Feature map. The last approach is based on feature maps, which provide a very simple way of generating kernels.

Lemma 3.16. *Given some Hilbert space $\mathcal{W}$, any feature map $\Phi : \mathcal{X} \to \mathcal{L}(\mathcal{W}, \mathcal{Y})$ defines a kernel as*

$$K(x, \hat{x}) = \Phi(x)\Phi(\hat{x})^{\#} : \mathcal{Y} \to \mathcal{Y}. \tag{3.20}$$

Proof. We need to prove that it is bounded, symmetric and positive definite:

1. Since $\Phi(x)$ is continuous, its adjoint $\Phi(x)^{\#}$ is continuous and the composition $\Phi(x) \circ \Phi(\hat{x})^{\#}$ is also continuous.

2. It is symmetric since $(\Phi(x) \circ \Phi(\hat{x})^{\#})^{\#} = ((\Phi(x)^{\#})^{\#} \circ \Phi(\hat{x})^{\#}) = (\Phi(x) \circ \Phi(\hat{x})^{\#})$.

3. It is semipositive definite since, given any $x_1, \ldots, x_n \in \mathcal{X}, y_1, \ldots, y_n \in \mathcal{Y}$

$$\sum_{i,j=1}^{n} [y_i, K(x_i, x_j)y_j]_{\mathcal{Y}} = \sum_{i,j=1}^{n} \left[y_i, \Phi(x_i)\Phi(x_j)^{\#}y_j \right]_{\mathcal{Y}}$$

$$= \sum_{i,j=1}^{n} \left[\Phi(x_i)^{\#}y_i, \Phi(x_j)^{\#}y_j \right]_{\mathcal{Y}} = \left\| \sum_{i=1}^{n} \Phi(x_i)^{\#}y_i \right\|^2 \geq 0.$$

$\square$

Since K as defined in (3.20) is a kernel, according to Theorem 3.15 we can find its corresponding vector-valued Hilbert space $\mathcal{H}$.

Representer Theorem for Operator-Valued Kernels

The Representer Theorem is a crucial result in Optimization and Machine Learning. Given a regularized empirical risk, under some assumptions, the theorem gives a precise description of the minimizer f^* as a finite linear combination of functions $K(\cdot, x_i)$ where x_i are part of the empirical sample. This result is extended in Micchelli and Pontil (2005, Theorem 4.2) for operator-valued kernels through the next theorem.

Theorem 3.17. *Let $\mathcal{Y}$ be a Hilbert space and let $\mathcal{H}$ be the Hilbert space of $\mathcal{Y}$-valued functions with an operator-valued reproducing kernel K. Let $V : \mathcal{Y}^n \times \mathbb{R}_+ \to \mathbb{R}$ be a function strictly increasing in its second variable and consider the problem of minimizing the functional*

$$E(f) = V((f(x_1), \ldots, f(x_n)), \|f\|^2) \tag{3.21}$$

in $\mathcal{H}$. If f_0 minimizes E, then $f_0 = \sum_{j=1}^{n} K(\cdot, x_j)c_j$ where $c_j \in \mathcal{Y}$. In addition, if V is strictly convex, the minimizer is unique.

The proof can be found in Micchelli and Pontil (2005).

Bijection between Scalar and Vector-Valued Kernels

The scalar-valued kernels are well known and studied but this is not the case for operator-valued kernels. However, as shown in Baldassarre et al. (2012); Hein et al. (2004), we can find a bijection between operator-valued kernels and scalar-valued ones.

Lemma 3.18. *Let $\mathcal{Y}$ be a finite-dimensional Hilbert space and $K : \mathcal{X} \times \mathcal{X} \to \mathcal{L}(\mathcal{Y})$ be an operator-valued kernel. Consider also the scalar-valued kernel $L : (\mathcal{X}, \mathcal{Y}) \times (\mathcal{X}, \mathcal{Y}) \to \mathbb{R}$ such that $L((x, z), (\hat{x}, \hat{z})) = [z, K(x, \hat{x})\hat{z}]_{\mathcal{Y}}$. Then the map $K \to L$ is a bijection.*

Moreover, if we focus on finite dimensional Hilbert spaces, that is, isomorphic to $\mathbb{R}^d$ given an operator-valued kernel K the corresponding scalar-valued kernel L is defined using the normal basis as

$$L((x, e_r), (\hat{x}, e_s)) = [e_r, K(x, \hat{x})e_s]_{\mathcal{Y}} = K(x, \hat{x})_{rs}; \ r, s = 1, \dots, d.$$

That is, each pair $(x, \hat{x})$ defines a matrix which contains the information of how the different outputs, or tasks, are related.

3.3.2 Tensor Product of Reproducing Kernel Hilbert Spaces

Recall that, given two vector spaces V and W, the tensor product of these spaces, $V \otimes W$, is associated with the bilinear map

$$
\begin{aligned}
V \times W &\to V \otimes W \\
(v, w) &\to v \otimes w
\end{aligned}
.
$$

Also, consider $\mathcal{H}_1 \otimes \mathcal{H}_2$ as the space of the tensor product of two scalar-valued RKHS' $\mathcal{H}_1$ and $\mathcal{H}_2$ with reproducing kernels K_1, K_2, where the functions $f \in \mathcal{H}_i$ are defined as $f : \mathcal{X}_i \to \mathcal{Y}_i$ for $i = 1, 2$. This tensor space is also a Hilbert space endowed with the inner product:

$$
\begin{aligned}
\langle, \rangle : (\mathcal{H}_1 \otimes \mathcal{H}_2) &\times (\mathcal{H}_1 \otimes \mathcal{H}_2) \to \mathbb{R} \\
(f_1 \otimes f_2) &\quad (\hat{f}_1 \otimes \hat{f}_2) \to \left\langle f_1, \hat{f}_1 \right\rangle \left\langle f_2, \hat{f}_2 \right\rangle
\end{aligned}
\tag{3.22}
$$

It is easy to check that this inner product is symmetric because the inner products of both $\mathcal{H}_1$ and $\mathcal{H}_2$ are symmetric. It is linear because the inner products of both $\mathcal{H}_1$ and $\mathcal{H}_2$ are linear and the tensor product is linear. Finally, it is positive definite since $\langle f_1, f_1 \rangle \langle f_2, f_2 \rangle \geq 0$ and $\langle f_1, f_1 \rangle \langle f_2, f_2 \rangle = 0 \implies \langle f_i, f_i \rangle = 0$ for $i = 1$ or $i = 2$; taking $i = 1$ without loss of generality, then $f_1 = 0$, so $f_1 \otimes f_2 = 0 \in \mathcal{H}_1 \otimes \mathcal{H}_2$. To apply the Riesz Theorem in this Hilbert space it is necessary to check wether the evaluation functionals are continuous or, equivalently, bounded.

Proposition 3.19 (RKHS as tensor product of RKHS')**.** *Let $\mathcal{H}_1$ and $\mathcal{H}_2$ be RKHS' then the space $\mathcal{H} = \mathcal{H}_1 \otimes \mathcal{H}_2$, with evaluation functionals $L_{x_1 \otimes x_2}$ defined as*

$$L_{x_1 \otimes x_2}(f_1 \otimes f_2) = L_{x_1}(f_1) \otimes L_{x_2}(f_2) = f_1(x_1) \otimes f_2(x_2)$$

for $x_1 \otimes x_2 \in \mathcal{X}_1 \otimes \mathcal{X}_2$, is an RKHS with the inner product defined in (3.22).

Proof. It is necessary to ensure that the operators $L_{x_1 \otimes x_2}$ are bounded. Given any $f_1 \otimes f_2 \in \mathcal{H}_1 \otimes \mathcal{H}_2$,

$$\|L_{x_1 \otimes x_2}(f_1 \otimes f_2)\| = \langle f_1(x_1), f_1(x_1) \rangle \langle f_2(x_2), f_2(x_2) \rangle \leq \|f_1(x_1)\| \|f_2(x_2)\|.$$

Then, since $\mathcal{H}_1$ and $\mathcal{H}_2$ are bounded, the product $\|f_1(x_1)\| \|f_2(x_2)\|$ is also bounded. $\quad\square$

We also need to define the kernel function for this tensor product space.

Proposition 3.20. *The kernel function*

$$
\begin{aligned}
K_1 \otimes K_2 : (\mathcal{X}_1 \otimes \mathcal{X}_2) \times \quad &(\mathcal{X}_1 \otimes \mathcal{X}_2) \to \qquad\qquad \mathbb{R} \\
(x_1 \otimes x_2) \qquad &(\hat{x}_1 \otimes \hat{x}_2) \to \quad K_1(x_1, x_2) K_2(\hat{x}_1, \hat{x}_2)
\end{aligned}
$$

is a reproducing kernel for the Hilbert space $\mathcal{H}_1 \otimes \mathcal{H}_2$.

Proof. First, $K_1 \otimes K_2$ is a kernel, that is, it is symmetric and positive-definite. The proof is very similar to the symmetry and positive definiteness proof of the inner product (3.22). To observe that $K_1 \otimes K_2$ has the reproducing property, we write

$$
\langle K_1 \otimes K_2(\cdot, x_1 \otimes x_2), f_1 \otimes f_2 \rangle = \langle K_1(\cdot, x_1), f_1 \rangle \langle K_2(\cdot, f_2), x_2 \rangle = f_1(x_1) f_2(x_2).
$$

$\qquad\qquad\qquad\qquad\qquad\qquad\qquad\qquad\qquad\qquad\qquad\qquad\qquad\qquad\qquad\square$

Separable Kernels

To understand the connection between tensor product RKHS's and vector-valued RKHS's we study a special case of operator-valued kernels. A standard assumption is that the relation among the different outputs is independent of the pair $(x, \hat{x})$, that is, the kernel is separable (Álvarez et al., 2012; Kadri et al., 2016).

Definition 3.21 (Separable Kernel). An operator-valued kernel $K(x, \hat{x}) : \mathcal{Y} \to \mathcal{Y}$ is called separable if

$$
K(x, \hat{x}) = k(x, \hat{x}) M
$$

here $k(\cdot, \cdot)$ is a scalar-valued kernel and M is some fixed operator $M \in \mathcal{L}(\mathcal{Y})$.

That is, the operator $K(x, \hat{x})$ decouples in two parts: the similarity between x and $\hat{x}$ measured by $k(\cdot, \cdot)$ and the interaction between the different outputs expressed by M.

Given a symmetric, positive definite operator M, we can interpret the separable kernels as tensor product of kernels. Let $\mathcal{X}$ a non-empty set and consider the RKHS $\mathcal{H}$ of functions $f : \mathcal{X} \to \mathbb{R}$, with reproducing kernel $k_\mathcal{H}(\cdot, \cdot)$. Also, let $\mathcal{Y}$ be a finite-dimensional Hilbert space, so it is isomorphic to $\mathbb{R}^d$, and consider the kernel $K_\mathcal{Y}(y, \hat{y}) = [y, M\hat{y}]_\mathcal{Y}$. Since M is symmetric and positive definite, we can write $M = B^\intercal B$. Then, considering the map $\Phi(y) = By$, we can express the kernel as

$$
K_\mathcal{Y}(y, \hat{y}) = [y, M\hat{y}]_\mathcal{Y} = [\phi(y), \phi(\hat{y})]_\mathcal{Y},
$$

so $K_\mathcal{Y}(y, \hat{y})$ is a reproducing kernel for $\mathcal{Y}$.
TODO: Explicar en Chapter 2 cómo construir kernels escalares
Now, the separable operator-valued kernel $K : \mathcal{X} \times \mathcal{X} \to \mathcal{L}(\mathcal{Y})$ such that $K(x, \hat{x}) = k(x, \hat{x}) M$ with $M \in \mathcal{L}(\mathcal{Y})$, can be expressed as

$$
k_\mathcal{H} \otimes K_\mathcal{Y}(x \otimes y, \hat{x} \otimes \hat{y}) = k_\mathcal{H}(x, \hat{x}) K_\mathcal{Y}(y, \hat{y}) = k_\mathcal{H}(x, \hat{x}) [y, M\hat{y}]_\mathcal{Y}.
$$

Moreover, observe that using this kernel with the basis of $\mathcal{Y}$, $\{e_1, \ldots, e_d\}$,

$$
k_\mathcal{H} \otimes K_\mathcal{Y}((x \otimes e_r), (\hat{x} \otimes e_s)) = k_\mathcal{H}(x, \hat{x}) (M)_{rs}.
$$

This last result will be useful for expressing Multi-Task Learning problems as standard or Single-Task ones, as it is shown in the following sections.

3.3.3 Using Kernels in Multi-Task Learning

There exists a plethora of work about Single-Task Learning within regularization theory, where the problem to solve is

$$\arg \min_w \sum_{i=1}^{n} \ell(y_i, \langle w, \phi(x_i) \rangle) + \lambda \langle w, w \rangle. \tag{3.23}$$

Here, ℓ is the loss function and ϕ is a transformation to include non-linearity. Popular models such as Ridge Regression or SVMs are particular cases of this formulation for different choices of ℓ and ϕ. One crucial result for these kind of problems is the *Representer Theorem*, which states that the any minimizer of problem (3.23) has the form

$$w = \sum_{i=1}^{n} c_j \phi(x_j). \tag{3.24}$$

TODO: Explicar el Representer Theorem y Kernel Trick en Chapter 2
 Given w represented as in (3.24), we write

$$\langle w, \phi(\hat{x}) \rangle = \sum_{i=1}^{n} c_j \langle \phi(x_j), \phi(\hat{x}) \rangle.$$

This is very useful because we can apply the kernel trick and use the transformations ϕ only implicitly. In this subsection it is shown how a broad class of Multi-Task problems can be expressed as regularized Single-Task problems.

Linear MTL Models

Building upon the ideas discussed in Evgeniou and Pontil (2004), two useful results are presented in Evgeniou et al. (2005), which show how we can apply Single-Task Learning methods to MTL problems. The first result (Evgeniou et al., 2005, Proposition 1) is given for linear models and illustrates under which conditions we can adapt these results in an MTL context. Consider the linear MTL problem where we want to estimate the task parameters $u_r : r = 1, \ldots, T$, so we define $\boldsymbol{u}^\mathsf{T} = (u_1^\mathsf{T}, \ldots, u_T^\mathsf{T}) \in \mathbb{R}^{Td}$. Then we want to minimize

$$R(\boldsymbol{u}) = \sum_{r=1}^{T} \sum_{i=1}^{m} \ell(y_i^r, \langle u_r, x_i^r \rangle) + \mu \left(\boldsymbol{u}^\mathsf{T} E \boldsymbol{u} \right), \tag{3.25}$$

where

$$J(\boldsymbol{u}) = \boldsymbol{u}^\mathsf{T} E \boldsymbol{u} \tag{3.26}$$

is the regularizer, and different choices of $J(\boldsymbol{u})$, i.e. choices of the matrix E, can encode different beliefs about the task structure. For example, if $J(\boldsymbol{u}) = \sum_{r=1}^{T} \|u_r\|^2$ the problem decouples and we get independent task learning, so there is no relation among tasks; if $J(\boldsymbol{u}) = \sum_{r,s=1}^{T} \|u_r - u_s\|^2$ we are enforcing the parameters from different tasks to be close, so we expect all tasks to be similar.

Then, Evgeniou *et al.* propose to consider a vector $\boldsymbol{w} \in \mathbb{R}^p$ with $p \geq Td$ such that we can express $\langle u_r, x \rangle$ as $\langle B_r^\mathsf{T} \boldsymbol{w}, x \rangle$, where B_r is a $p \times d$ matrix yet to be specified. One condition for B_r is to be full rank so we can find such $\boldsymbol{w}$. Note that we can also interpret B_r as a feature map $f : \mathbb{R}^d \to \mathbb{R}^p$ such that $\langle u_r, x \rangle = \langle \boldsymbol{w}, B_r x \rangle$. Observe that using the matrices B_r we have the following MTL kernel

$$\hat{k}(x^r, y^s) = \hat{k}((x, r), (y, s)) = x^\mathsf{T} B_r^\mathsf{T} B_s y.$$

Using these feature maps we would like to write the MTL problem as a Single-Task problem

$$S(\boldsymbol{w}) = \sum_{r=1}^{T} \sum_{i=1}^{m} \ell(y_i^r, \langle \boldsymbol{w}, B_r x_i^r \rangle) + \mu \langle \boldsymbol{w}, \boldsymbol{w} \rangle, \tag{3.27}$$

We also define the feature matrix B as the concatenation $B = (B_r : r = 1, \ldots, T) \in \mathbb{R}^{p \times Td}$, then we present the first result of Evgeniou et al. (2005).

Proposition 3.22. *If the feature matrix B is full rank and we define the matrix E in equation (3.25) as $E = (B^\mathsf{T} B)^{-1}$, then*

$$S(\boldsymbol{w}) = R(B^\mathsf{T} \boldsymbol{w}),$$

and therefore $\boldsymbol{u}^* = B^\mathsf{T} \boldsymbol{w}^*$.

One important consequence of this result is that since we can solve the MTL problem (3.25) as the STL problem (3.23) with ϕ being the identity function, then we can apply the Representer Theorem. That is, the solution $\boldsymbol{w}^*$ of problem (3.23) has the form

$$\boldsymbol{w} = \sum_{r=1}^{T} \sum_{i=1}^{m} c_i B_r x_i^r,$$

and the prediction can be expressed as

$$\langle \boldsymbol{w}, \hat{x}^s \rangle = \sum_{r=1}^{T} \sum_{i=1}^{m} \alpha_i^r (x_i^r)^\mathsf{T} B_r^\mathsf{T} B_s \hat{x}^s = \sum_{r=1}^{T} \sum_{i=1}^{m} c_i \hat{k}(x_i^r, \hat{x}^s).$$

A Posteriori Kernel Extension of MTL Models

Evgeniou *et al.* also extend these results to kernelized models. In the following lemma (Evgeniou et al., 2005, Lemma 2) the give the conditions under which the extension is possible.

Lemma 3.23. *Given a space $\mathcal{T}$ such that for every $r = 1, \ldots, T$ there are prescribed mappings $z_r : \mathcal{X} \to \mathcal{T}$, if G is a kernel on $\mathcal{T} \times \mathcal{T}$, then*

$$K((x, r), (y, s)) = G(z_r(x), z_s(t)), \ x, t \in \mathcal{X}, \ r, s = 1, \ldots, T, \tag{3.28}$$

is a multi-task kernel.

This defines K as a semi-positive functional over the product space $\mathcal{X} \times \mathcal{T}$, which is named a multi-task kernel. The mappings described in Evgeniou et al. (2005) are

$$z_r(x) = B_r x,$$

where B_r are the $p \times d$ matrices previously defined. Then, two examples of multi-task kernels using this lemma are given. The polynomial kernel is defined as

$$K((x,r),(y,s)) = (x^\intercal B_r^\intercal B_s y)$$

and the multi-task Gaussian kernel is defined as

$$K((x,r),(y,s)) = \exp\left(-\gamma \left\| B_r x - B_s y \right\|^2\right).$$

That is, using the result of Proposition 3.22, since $E^{-1} = B^\intercal B$, we can incorporate the task-regularizer information into the Gaussian kernel using that

$$\begin{aligned}
\left\| B_r x - B_s y \right\|^2 &= x^\intercal B_r^\intercal B_r x + y^\intercal B_s^\intercal B_s y - 2x_r^\intercal B_r^\intercal B_s y_s \\
&= x^\intercal E_{rr}^{-1} x + y^\intercal E_{ss}^{-1} y - 2x_r^\intercal E_{rs}^{-1} y_s.
\end{aligned}$$

That is, we use the task information in the original space and then apply the non-linear transformation.

A Priori Kernel Extension of MTL Models

The kernel extension presented in Evgeniou et al. (2005) proposes to use a mapping in the original finite space to incorporate the task information and then apply the kernel trick over the new mapped features. However, these results do not permit to perform the, possibly infinite dimensional, mapping corresponding to a kernel and then incorporate the task information in the new space. Here we show another approach which makes it possible to replicate the results for the infinite-dimensional case by using tensor products. Consider the RKHS $\mathcal{H}$ and the functional

$$R(u_1, \ldots, u_T) = \sum_{r=1}^{T} \sum_{i=1}^{m} \ell(y_i^r, \langle u_r, \phi(x_i^r) \rangle) + \mu \sum_r \sum_s E_{rs} \langle u_r, u_s \rangle, \qquad (3.29)$$

where ϕ is a feature transformation; $u_1, \ldots, u_T \in \mathcal{H}$; and E is a $T \times T$ symmetric, positive definite matrix. The following lemma illustrates how to minimize this functional as a single task problem.

Lemma 3.24. *The predictions $\langle u_r^*, \phi(x) \rangle$ of the solutions $u_1^*, \ldots, u_T^*$ from the Multi-Task risk (3.29) can be obtained solving the problem*

$$S(\boldsymbol{w}) = \sum_{r=1}^{T} \sum_{i=1}^{m} \ell(y_i^r, \langle \boldsymbol{w}, (B_r \otimes \phi(x_i^r)) \rangle) + \mu \boldsymbol{w}^\intercal \boldsymbol{w}, \qquad (3.30)$$

where $\boldsymbol{w} \in \mathbb{R}^p \otimes \mathcal{H}$ with $p \geq T$ and B_r are the columns of a full rank matrix $B \in \mathbb{R}^{p \times T}$ such that $E^{-1} = B^\intercal B$.

Proof. Replicating the idea of Evgeniou et al. (2005), given an orthonormal base of $\mathbb{R}^T$, which we will name $\{e_1, \ldots, e_T\}$, we can define

$$\boldsymbol{u} = \sum_{t=1}^{T} e_r \otimes u_r,$$

such that $\boldsymbol{u} \in \mathbb{R}^T \otimes \mathcal{H}$. Then, we can reformulate (3.29) as

$$R(\boldsymbol{u}) = \sum_{r=1}^{T} \sum_{i=1}^{m} \ell(y_i^r, \langle \boldsymbol{u}, e_r \otimes \phi(x_i^r) \rangle) + \mu \left(\boldsymbol{u}^\mathsf{T} (E \otimes I) \boldsymbol{u} \right). \tag{3.31}$$

Since $E \in \mathbb{R}^{T \times T}$ is symmetric and positive definite, we can find $B \in \mathbb{R}^{p \times T}, p \geq T$ and rank $B = T$ such that $E^{-1} = B^\mathsf{T} B$, using for example the SVD. Then, with the properties of the tensor product of linear maps,

$$E^{-1} \otimes I = (B^\mathsf{T} B) \otimes I = (B^\mathsf{T} \otimes I)(B \otimes I),$$

Consider the change of variable $\boldsymbol{u} = (B^\mathsf{T} \otimes I) \boldsymbol{w}$, where $\boldsymbol{w} \in \mathbb{R}^p \otimes \mathcal{H}$. Observe that this can always be done because B is full rank. Rewriting (3.31) using $\boldsymbol{w}$,

$$R(\boldsymbol{w}) = \sum_{r=1}^{T} \sum_{i=1}^{m} \ell(y_i^r, \langle (B^\mathsf{T} \otimes I) \boldsymbol{w}, (e_r \otimes \phi(x_i^r)) \rangle) + \mu \boldsymbol{w}^\mathsf{T} (B^\mathsf{T} \otimes I)^\mathsf{T} (E \otimes I)(B^\mathsf{T} \otimes I) \boldsymbol{w}$$

$$= \sum_{r=1}^{T} \sum_{i=1}^{m} \ell(y_i^r, \langle \boldsymbol{w}, (B \otimes I)(e_r \otimes \phi(x_i^r)) \rangle) + \mu \boldsymbol{w}^\mathsf{T} \boldsymbol{w},$$

which is equivalent to

$$S(\boldsymbol{w}) = \sum_{r=1}^{T} \sum_{i=1}^{m} \ell(y_i^r, \langle \boldsymbol{w}, (B_r \otimes \phi(x_i^r)) \rangle) + \mu \boldsymbol{w}^\mathsf{T} \boldsymbol{w}.$$

We are thus considering a regularized functional $S(\boldsymbol{w})$ where we seek the minimum over functions $\boldsymbol{w}$ in the Hilbert space $\mathbb{R}^p \otimes \mathcal{H}$. Note that in this space the inner product is:

$$\begin{array}{cccc} \langle , \rangle : (\mathbb{R}^p \otimes \mathcal{H}) & \times & (\mathbb{R}^p \otimes \mathcal{H}) \to & \mathbb{R} \\ (z_1, \phi(x_1)) & & (z_2, \phi(x_2)) \to & \langle z_1, z_2 \rangle k(x_1, x_2) \end{array},$$

where $k(\cdot, \cdot)$ is the reproducing kernel of the space of functions $\phi(\cdot)$. However we are only interested in those cases where $z = Be_r = B_r$ for some $r = 1, \ldots, T$, then $\langle B_r, B_s \rangle = E_{rs}^{-1}$. Since the regularizer is clearly increasing in $\|w\|^2$, we can apply the Representer theorem, which states that the minimizer of $S(\boldsymbol{w})$ has the form

$$\boldsymbol{w}^* = \sum_{r=1}^{T} \sum_{i=1}^{m} \alpha_i^r (B_r \otimes \phi(x_i^r)),$$

Using the correspondence between $\boldsymbol{u}^*$ and $\boldsymbol{w}^*$, we have

$$\boldsymbol{u}^* = (B^\mathsf{T} \otimes I) \boldsymbol{w}^* = \sum_{r=1}^{T} \sum_{i=1}^{m} \alpha_i^r (\mathrm{vect}(\langle B_1, B_r \rangle, \ldots, \langle B_T, B_r \rangle) \otimes \phi(x_i^r)),$$

where $\text{vect}(a_1, \ldots, a_T)$ is T-dimensional vector with elements $a_1, \ldots, a_T$. Then, we can recover the predictions corresponding to the solutions u_r^* as

$$
\begin{aligned}
\langle u_s, \phi(\hat{x}^s) \rangle &= \langle \boldsymbol{u}, e_s \otimes \phi(\hat{x}^s) \rangle \\
&= \left\langle \sum_{r=1}^{T} \sum_{i=1}^{m} \alpha_i^r (\text{vect}(\langle B_1, B_r \rangle, \ldots, \langle B_T, B_r \rangle) \otimes \phi(x_i^r)), e_s \otimes \phi(\hat{x}^s) \right\rangle \\
&= \sum_{r=1}^{T} \sum_{i=1}^{m} \alpha_i^r \langle B_s, B_r \rangle \langle \phi(x_i^r), \phi(x_s) \rangle \\
&= \sum_{r=1}^{T} \sum_{i=1}^{m} \alpha_i^r (E^{-1})_{rs} k(x_i^r, \hat{x}^s).
\end{aligned}
$$

$\square$

Observe that, applying the corresponding feature map $B \otimes I$, the predictions can be obtained equivalently using the common $\boldsymbol{w}$ as

$$
\begin{aligned}
\langle \boldsymbol{w}, (B \otimes I)(e_s \otimes \phi(\hat{x}^s)) \rangle &= \langle \boldsymbol{w}, (B_s \otimes \phi(\hat{x}^s)) \rangle \\
&= \sum_{r=1}^{T} \sum_{i=1}^{m} \alpha_i^r \langle B_r \otimes \phi(x_i^r), B_s \otimes \phi(\hat{x}^s) \rangle \\
&= \sum_{r=1}^{T} \sum_{i=1}^{m} \alpha_i^r \langle B_r, B_s \rangle \langle \phi(x_i^r), \phi(\hat{x}^s) \rangle \\
&= \sum_{r=1}^{T} \sum_{i=1}^{m} \alpha_i^r (E^{-1})_{rs} k(x_i^r, \hat{x}^s).
\end{aligned}
$$

That is, we have expressed the Multi-Task problem as a Single-Task problem with the Multi-Task kernel is

$$
\hat{k}(x_i^r, x_j^s) = (E^{-1})_{rs} k(x_i^r, x_j^s).
$$

Note that the kernels obtained in this way, unlike those obtained using Lemma 3.28, split the inter-task relations and the similarity between data points. That is, we can implicitly send our data into another, a possibly infinite-dimensional, space and apply the task information after this transformation. These are separable kernels (Álvarez et al., 2012), but, to the best of our knowledge, this is the first time they are constructed using tensor products.

Examples of Multi-Task Kernels

Using the framework for MTL with Kernel methods we can choose different regularizations, induced by the matrix E, which lead to different Multi-Task approaches.

Independent Tasks This is the trivial case when $E = I_T$ and therefore $B = I_T$, that is

$$
B_r^\intercal = (\overset{1}{0}, \ldots, \overset{r}{1}, \ldots, \overset{T}{0}),
$$

and the kernel is

$$\hat{k}(x_i^r, x_j^s) = \langle B_r, B_s \rangle \, k(x_i^r, x_j^s) = (\delta_{rs}) k(x_i^r, x_j^s).$$

This approach is not a proper MTL method because each task is learned separately, and no coupling is being enforced among tasks.

Independent Parts with Shared Common Model   When the matrix B is selected such that its columns are

$$B_r^\mathsf{T} = (\overbrace{0}^{1}, \ldots, \overbrace{1}^{r}, \ldots, \overbrace{0}^{T}, \overbrace{\frac{1}{\mu}}^{T+1}),$$

the corresponding multi-task kernel is:

$$\hat{k}(x_i^r, x_j^s) = \langle B_r, B_s \rangle \, k(x_i^r, x_j^s) = (\frac{1}{\mu} + \delta_{rs}) k(x_i^r, x_j^s).$$

This is equivalent to the approach presented in the work of Evgeniou and Pontil (2004) where it is named regularized MTL. The goal is to find a decision function for each task, each being defined by a vector

$$w_r = w + v_r,$$

where w is common to all tasks and v_r is task-specific. The primal problem of regularized MTL SVM, using the unified formulation, is

$$
\begin{aligned}
\underset{w, v_r, \xi_i^r}{\arg\min} \quad & C \sum_{r=1}^{T} \sum_{i=1}^{m_r} \xi_i^r + \frac{1}{2} \langle w, w \rangle + \sum_{r=1}^{T} \frac{\mu}{2} \langle v_r, v_r \rangle \\
\text{s.t.} \quad & y_i^r (\langle w, x_i^r \rangle + \langle v_r, x_i^r \rangle) \geq p_i^r - \xi_i^r, \\
& \xi_i^r \geq 0, \\
\text{for} \quad & r = 1, \ldots, T; \ i = 1, \ldots, m_r.
\end{aligned}
\tag{3.32}
$$

Note that μ is a parameter that controls the tradeoff between the relevance of common and specific models. That is, when μ tends to infinite, the resulting model approaches a common-task standard SVM; when μ tends to zero, an independent task approach is taken, with one standard SVM problem for each task.
TODO: Explicar unified formulation en Chapter 2
 Moreover, in Evgeniou and Pontil (2004) it is shown that solving (3.32) is equivalent to solving the problem

$$
\begin{aligned}
\underset{w w_r, \xi_i^r}{\arg\min} \quad & C \sum_{r=1}^{T} \sum_{i=1}^{m_r} \xi_i^r + \frac{1}{2} \sum_{r=1}^{T} \|w_r\|^2 + \frac{\mu}{2} \sum_{r=1}^{T} \left\| w_r - \sum_{s=1}^{T} w_s \right\|^2 \\
\text{s.t.} \quad & y_i^r (\langle w_r, x_i^r \rangle) \geq p_i^r - \xi_i^r, \\
& \xi_i^r \geq 0, \\
\text{for} \quad & r = 1, \ldots, T; \ i = 1, \ldots, m_r.
\end{aligned}
$$

Now, only the w_r variables are included, and it is clearer that μ penalizes the variance of the w_r vectors, so all models w_r will tend to a common model as μ grows.

This is a very interesting approach because, as it was pointed out in Liang and Cherkassky (2008), this approach has connections with the SVM+ approach from Vapnik and Izmailov (2015).

Graph Laplacian When the tasks are considered as nodes in a graph, and the non-negative weights of the edges of this graph capture the relation between each pair of tasks, the matrix E can be seen as a Laplacian matrix $L = D - A$. Here A is the adjacency matrix indicating the weights of the edges between each pair of tasks, and D is the degree matrix, a diagonal matrix where each diagonal term is the sum of the corresponding row of A. Observe that using this matrix, the regularization term is

$$
\begin{aligned}
\boldsymbol{u}^{\mathsf{T}}(L \otimes I)\boldsymbol{u} &= \sum_{r=1}^{T}\sum_{s=1}^{T} u_r^{\mathsf{T}} L_{rs} u_s \\
&= \sum_{r=1}^{T}\sum_{s=1}^{T} u_r^{\mathsf{T}} (D-A)_{rs} u_s \\
&= \sum_{r=1}^{T}\sum_{s=1}^{T} u_r^{\mathsf{T}} \left(\delta_{rs} \sum_q A_{rq} \right) u_s - \sum_{r=1}^{T}\sum_{s=1}^{T} u_r^{\mathsf{T}} A_{rs} u_s \\
&= \sum_{r=1}^{T} u_r^{\mathsf{T}} \sum_q A_{rq} u_r + \sum_{s=1}^{T} u_s^{\mathsf{T}} \sum_q A_{sq} u_s - \sum_{r=1}^{T}\sum_{s=1}^{T} u_r^{\mathsf{T}} A_{rs} u_s \\
&= \sum_{r=1}^{T}\sum_{s=1}^{T} u_r^{\mathsf{T}} \left(A_{rs} u_s + u_s^{\mathsf{T}} A_{rs} u_s - u_r^{\mathsf{T}} A_{rs} u_s \right) \\
&= \sum_{r=1}^{T}\sum_{s=1}^{T} A_{rs} \left\| u_r - u_s \right\|^2 \ .
\end{aligned}
$$

That is, the distance between task models is penalized, weighted by the degree of similatiy between the tasks as indicated by the graph.

3.4 Multi-Task Learning Methods: An Overview

In this section a general overview of the MTL methods will be given, categorizing some of the most relevant approaches. Recall that a Multi-Task Learning sample is

$$
\left\{ (x_i^r, y_i^r) \in \mathcal{X} \times \mathcal{Y}; i = 1, \ldots, m_r; r = 1, \ldots, T \right\},
$$

where T is the number of tasks and m_r is the number of examples in task r. The goal is to find task-specialized hypotheses h_r such that the regularized risk

$$
\sum_{r=1}^{T}\sum_{i=1}^{m_r} \ell(h(x_i^r), y_i^r) + \Omega(h_1, \ldots, h_T)
$$

is minimized, where Ω is some regularizer of the hypotheses. To enforce the coupling between tasks, we can consider a taxonomy with three main groups: feature-based, parameter-based and combination-based strategies. The feature-based approaches define the hypotheses as the composition of a common feature-building function f and a task-specific function g_r, i.e. $h_r(x_i^r) = g_r \circ f(x_i^r)$, where f and g_r can be defined in multiple

ways. The parameter-based approach is based on enforcing the coupling through the regularization of the parameters of linear models, which are built over some non-learnable features, i.e. $h_r(x_i^r) = w_r^\intercal \phi(x_i^r)$. Finally, the combination based approach uses the combination of a common part and task specific parts, i.e. $h_r(x_i^r) = g(x_i^r) + g_r(x_i^r)$, where g and g_r can be modeled in multiple ways. In this section we develop a subsection for each one of these approaches. Although most MTL strategies used in deep learning can be categorized as feature-based, and most kernel methods use parameter-based or combination-based strategies, given the relevance of these models, MTL with neural networks and MTL with kernel methods will be treated separately in their specific subsections.

3.4.1   Feature-based Multi-Task Learning

The feature-based methods try to find a set of features that are useful for all tasks. To do that, a shared feature-building function f is considered, and the hypotheses are then $h_r(x_i^r) = g_r \circ f(x_i^r)$. Two main approaches are taken: feature learning, which tries to learn new features from the original ones, and feature-selection, which selects a subset of the original features.

Feature Learning approaches

In the feature learning approach, the feature-building function f can be modeled in different ways. With neural networks, f can be modeled using shared layers of a neural network, while g_r is the linear model corresponding to the output layer, that is $h_r(x_i^r) = w_r^\intercal f(x_i^r, \Theta) + b_r$ where Θ are the parameters of the hidden layers and w_r, b_r are the parameters of the output layer. This is the idea behind the MT NN first given in Caruana (1997), which will be described in Subsection 3.4.4. With linear or kernel models, unlike the case with NN's, the function f is modeled as linear combinations of fixed features defined by ϕ, for example if $f(x_i^r) = (u_1^\intercal \phi(x_i^r), \ldots, u_k^\intercal \phi(x_i^r))$ for some $k \in \mathbb{N}^+$, where ϕ is the implicit transformation that is the identity for linear models. Then, for $r = 1, \ldots, T$, $h_r(x_i^r) = a_r^\intercal(U^\intercal \phi(x_i^r))$, where $a_r \in \mathbb{R}^k$ are the parameters of the linear models g_r. Observe that in the linear case, we can describe this models as $h_r(x_i^r) = w_r^\intercal x_i^r$, with $w_r = U a_r$ where $U = (u_1, \ldots, u_k)$. This idea, named Multi-Task Feature Learning, is presented in Argyriou et al. (2006), where they consider the linear case with $k = d$, the original dimension of the data. Then, the matrix $W = (w_1, \ldots, w_T)$ can be expressed as

$$\underset{d \times T}{W} = \underset{d \times d}{U}\,\underset{d \times T}{A}.$$

The authors impose also some restrictions to enforce that the features represented by $u_1, \ldots, u_T$ capture different information and only a subset of them is necessary for each task. The minimization problem to obtain the optimal model parameters is

$$\underset{\substack{U \in \mathbb{R}^{d \times d} \\ A \in \mathbb{R}^{d \times T}}}{\arg\min} \sum_{r=1}^{T} \sum_{i=1}^{m_r} \ell(y_i^r, \langle U a_r, x_i^r \rangle) + \lambda \|A\|_{2,1}^2 \ \ \text{s.t.} \ \ U^\intercal U = I, \qquad (3.33)$$

where the $L_{2,1}$ regularizer is used to impose row-sparsity across tasks, i.e. forcing some rows of A to be zero, which has the goal of using in all tasks the same subset of the features, which are represented by the columns of U; also, the matrix U is restricted to be orthonormal so that these columns do not contain overlapping information. Although

problem (3.33) is not jointly convex in U and A, in Argyriou et al. (2006) and Argyriou et al. (2008) it is shown to be equivalent to the convex problem

$$\underset{W\in\mathbb{R}^{d\times T},D\in\mathbb{R}^{d\times d}}{\arg\min}\sum_{r=1}^{T}\sum_{i=1}^{m_r}\ell(y_i^r,\langle w_r,x_i^r\rangle)+\lambda\sum_{r=1}^{T}\langle w_r,D^{-1}w_r\rangle \quad (3.34)$$
$$\text{s.t. } D\succeq 0,\ \operatorname{tr}(D)\leq 1.$$

If (A^*,U^*) is an optimal solution of (3.33), then

$$(W^*,D^*)=\left(U^*A^*,U^*\operatorname{diag}\left(\frac{\|a^1\|_2}{\|A\|_{2,1}},\ldots,\frac{\|a^d\|_2}{\|A\|_{2,1}}\right)(U^*)^{\mathsf{T}}\right),$$

where a^i are the rows of A, is an optimal solution of problem (3.34); conversely, given an optimal solution (W^*,D^*), if U^* has as columns an orthonormal basis of eigenvectors of D^* and $A^*=(U^*)^{\mathsf{T}}W^*$, (A^*,U^*) is an optimal solution of (3.33). To obtain an optimal solution (W^*,D^*) the authors, for a better stability, use a modified problem

$$\underset{W\in\mathbb{R}^{d\times T},D\in\mathbb{R}^{d\times d}}{\arg\min}\sum_{r=1}^{T}\sum_{i=1}^{m_r}\ell(y_i^r,\langle w_r,x_i^r\rangle)+\lambda\sum_{r=1}^{T}\langle w_r,D^{-1}w_r\rangle+\mu\operatorname{tr}(D^{-1}) \quad (3.35)$$
$$\text{s.t. } D\succeq 0,\ \operatorname{tr}(D)\leq 1.$$

This problem is solved using an iterated two-step strategy. The optimization with respect to W decouples in each task and the standard Representer Theorem can be used to solve it, and the one with respect to D has a closed solution

$$D^*=(W^{\mathsf{T}}W+\mu I)^{\frac{1}{2}}/\operatorname{tr}\left((W^{\mathsf{T}}W+\mu I)^{\frac{1}{2}}\right),$$

where the identity, which improves the stability, is consequence of the addition of the term $\operatorname{tr}(D^{-1})$ in the objective function. It is interesting to observe that the regularizer of (3.34) can be expressed as $\operatorname{tr}(W^{\mathsf{T}}D^{-1}W)$ and by plugging D^* in this formula, when $\mu=0$, we obtain the squared-trace norm regularizer for W:

$$\underset{W\in\mathbb{R}^{d\times T}}{\arg\min}\sum_{r=1}^{T}\sum_{i=1}^{m_r}\ell(y_i^r,\langle w_r,x_i^r\rangle)+\lambda\|W\|_*^2. \quad (3.36)$$

Here, $\|W\|_*=\operatorname{tr}\left((W^{\mathsf{T}}W)^{\frac{1}{2}}\right)$ denotes the trace norm (also known as nuclear norm), which can be seen as the continuous envelope of the rank since the nuclear norm is $\|W\|_*=\sum_{i=1}^{\min d,T}\lambda_i$, where λ_i are the eigenvalues of W, thus favouring low-rank solutions of W. That is, the initial problem (3.33) is equivalent to a problem where the trace norm regularization for matrix W is used.

In Argyriou et al. (2007) the idea of penalizing the eigenvalues is extended to any spectral funcion $F:\mathbb{S}_{++}^d\to\mathbb{S}_{++}^d$ where $\mathbb{S}_{++}^d$ is the set of matrices $A\in\mathbb{R}^{d\times d}$ symmetric and positive definite. The definition for the spectral function $F(A)$, for a diagonalizable matrix $A=V^{\mathsf{T}}\operatorname{diag}(\lambda_1,\ldots,\lambda_d)V$ is given in terms of a scalar function f that is applied over its eigenvalues:

$$F(A)=U^{\mathsf{T}}\operatorname{diag}(f(\lambda_1),\ldots,f(\lambda_d))U .$$

Then, since the trace is invariant under cyclic permutations, a generalized regularizer for problem (3.34) can be expressed as

$$\sum_t \langle w_t, F(D)w_t \rangle = \operatorname{tr}\left(W^\mathsf{T} F(D) W\right) = \operatorname{tr}\left(F(D)WW^\mathsf{T}\right).$$

It is easy to see that problem (3.34) is a particular case where $f(\lambda) = \lambda^{-1}$. In Maurer (2009) some bounds on the excess risks are given for this Multi-Task Feature Learning method.

Another relevant extension is shown in Agarwal et al. (2010), where instead of assuming that the task parameters w_r lie in a linear subspace, i.e. $w_r = U a_r$, the authors generalize this idea by assuming that the parameter w_r lies in a manifold $\mathcal{M}$. The resultant optimization problem to train the model is

$$\underset{W, \mathcal{M}, \boldsymbol{b}}{\arg\min} \sum_{r=1}^{T} \sum_{i=1}^{m_r} \ell(y_i^r, \langle w_r, x_i^r \rangle + b_r) + \lambda \sum_{r=1}^{T} \mathcal{P}_{\mathcal{M}}(w_r), \tag{3.37}$$

where $\mathcal{P}_{\mathcal{M}}(w_r)$ represents the distance between w_r and its projection on the manifold $\mathcal{M}$. Observe that if we define w_r as $w_r = U a_r$ as before, this distance is zero because $w_r \in \operatorname{span}(u_1, \ldots, u_k)$ for all $r = 1, \ldots, T$. Again, an approximation of (3.37) is used to obtain a convex problem, and it is solved using a two-step optimization algorithm.

Other distinct, relevant approach for Feature Learning is the one described in Maurer et al. (2013), where a sparse-coding method (Maurer and Pontil, 2010) is used for MTL. Maurer *et al.* present the problem

$$\underset{D \in \mathcal{D}_k, A \in \mathbb{R}^{k \times T}}{\arg\min} \sum_{r=1}^{T} \sum_{i=1}^{m_r} \ell(y_i^r, \langle D a_r, x_i^r \rangle) + \lambda \|D\|_{2,\infty} + \mu \|A\|_{1,\infty}. \tag{3.38}$$

Here, $\mathcal{D}_k$ is the set of k-dimensional dictionaries and every $D \in \mathcal{D}_k$ is a linear map $D : \mathbb{R}^k \to \mathcal{H}$ for some RKHS $\mathcal{H}$; in the linear case, where $\mathcal{H} = \mathbb{R}^d$, the set $\mathcal{D}_k$ is the set of matrices $\mathbb{R}^{d \times k}$, such that

$$\underset{d \times T}{W} = \underset{d \times k}{D} \underset{k \times T}{A}.$$

Although (3.33) and (3.38) share a similar form, there are crucial differences. The matrix U in (3.33) is an orthogonal square matrix, while the matrix D of (3.38) is overcomplete with $k > d$ columns of bounded norm. Also, in problem (3.33) non-linear features can be used while problem (3.38) is linear. A problem very similar to (3.38) is presented in Kumar and Daumé III (2012) where the idea is the same but the regularizers are the $L_{2,2}$ (Frobenius) norm for D and the $L_{1,1}$ norm for A:

$$\underset{D \in \mathcal{D}_k, A \in \mathbb{R}^{k \times T}}{\arg\min} \sum_{r=1}^{T} \sum_{i=1}^{m_r} \ell(y_i^r, \langle D a_r, x_i^r \rangle) + \lambda \|D\|_{2,2} + \mu \|A\|_{1,1}. \tag{3.39}$$

Here, we can interpret this model as a linear sparse combination, encoded in A, of some features encoded in D: $w_r^\mathsf{T} x_i^r = \sum_{\kappa=1}^{k} a_r^\kappa (D_\kappa^\mathsf{T} x_i^r)$. Unlike the Multi-Task Feature Learning approach of Argyriou et al, this sparse coding formulation is only presented in the linear setting.

Feature Selection approaches

The feature selection approaches are also driven by learning a good set of features for all tasks; however they focus on subsets of the original features. Due to their nature, the works following this strategy are based on linear models. This is a more rigid approach than that of Feature Learning but is also more interpretable.

Most works on Multi-Task Feature Selection use an $L_{p,q}$ regularization of the weights matrix W. The first work using this idea is Obozinski et al. (2006), where the authors solve the problem

$$\underset{W}{\arg\min} \sum_{r=1}^{T} \sum_{i=1}^{m_r} \ell(y_i^r, \langle w_r, x_i^r \rangle) + \lambda \|W\|_{2,1}^2 , \qquad (3.40)$$

where the $L_{2,1}$ regularization enforces row sparsity and forces different tasks to share a subset of features by pushing some rows w^i, which are the weights corresponding to the i-th feature, towards zero. In Liu et al. (2009) the $L_{\infty,1}$ regularization is used for the same goal. Then, in Gong et al. (2012a) this idea is generalized with a capped-$L_{p,1}$ penalty of W, which is defined as $\sum_{i=1}^{d} \min(\theta, \|w^i\|_p)$. That is, the parameter θ enables a more flexible regularization: with small values of θ the smallest rows of W are pushed towards zero since the rows with norms larger than θ will not dominate the sum. As θ grows this penalty will degenerate to the standard $L_{p,1}$ norm.

In Lozano and Swirszcz (2012) a multi-level lasso selection is presented where the main idea is to decompose each w_r^i, that is the i-th feature of the r-th task, as $w_r^i = \theta^i a_r^i$ and then, using the matrix $\Theta = \text{diag}(\theta^1, \ldots, \theta^d)$, they define the problem

$$\underset{\Theta, a_1, \ldots, a_T}{\arg\min} \sum_{r=1}^{T} \sum_{i=1}^{m_r} \ell(y_i^r, \langle \Theta a_r, x_i^r \rangle) + \mu \, \text{tr} \, \Theta + \nu \|A\|_{1,1} . \qquad (3.41)$$

By doing this, the features i such that $\theta^i = 0$ are discarded for all tasks, but the rest may be shared among tasks or not depending on the values of a_r^i. Observe that this is similar to the sparse coding problem shown in (3.38) with D, the "feature building" matrix, being limited to diagonal matrices since it is acting here as a selection matrix:

$$w_r^\intercal x_i^r = \sum_{i=1}^{k} a_r^i (\theta_i x_i^r) .$$

The feature selection methods based on $L_{p,1}$ regularization are shown to be equivalent to a Bayesian approximation with a generalized Gaussian prior in Zhang et al. (2010). Moreover, this approach also allows to find the relationship among tasks and to identify outliers. In Hernández-Lobato and Hernández-Lobato (2013) a horseshoe prior is used instead to learn feature covariance; and in Hernández-Lobato et al. (2015) this prior is also used to identify outlier tasks.

3.4.2 Parameter-based MTL

The parameter-based approaches, instead of trying to find a good shared feature space, enforce the coupling through the regularization of the parameters of linear models $h_r(x_i^r) = w_r^\intercal \phi(x_i^r)$, where ϕ is a fixed, non-learnable transformation. Since we have one vector w_r for each task, we can consider the matrix $W = (w_1 \ldots w_T)$. Some approaches

rely on the assumption that the Multi-Task weight matrix W has a low-rank, others try to learn the pairwise task relations or to cluster the tasks. A different approach is the decomposition one, where the assumption is that the matrix W can be expressed as the summation of multiple matrices. We summarize each approach below.

Low-rank approaches

In the low-rank approaches the assumption is that task parameters w_r share a low-dimensional space, or, at least, are close to this subspace. This is similar to the Feature Learning approach, but it is not that rigid, since it allows for some flexibility. The idea in Ando and Zhang (2005) is that the task parameters can be decomposed as

$$w_r = u_r + \Theta^\intercal v_r,$$

where $\Theta \in \mathbb{R}^{k \times d}$ spans a shared low dimensional space, that is $\Theta\Theta^\intercal = I_k$ with $k < d$, and d is the dimension of the data. Under this consideration, to train the model it is necessary to solve the problem

$$\underset{\Theta \in \mathbb{R}^{k \times d}, \boldsymbol{u}, \boldsymbol{v}}{\arg\min} \sum_{r=1}^{T} \sum_{i=1}^{m_r} \ell(y_i^r, \langle u_r + \Theta^\intercal v_r, x_i^r \rangle) + \lambda \sum_{r=1}^{T} \|u_r\|^2 \ \text{ s.t. } \ \Theta\Theta^\intercal = I_k. \tag{3.42}$$

Observe that this problem shares some similarities with (3.33). However, this is a more flexible approach, since the vectors u_r allow for deviations of the task parameters from the shared subspace. Problem (3.42) can be reformulated as

$$\underset{\Theta \in \mathbb{R}^{k \times d}, \boldsymbol{w}, \boldsymbol{v}}{\arg\min} \sum_{r=1}^{T} \sum_{i=1}^{m_r} \ell(y_i^r, \langle w_r, x_i^r \rangle) + \lambda \sum_{r=1}^{T} \|w_r - \Theta^\intercal v_r\|^2 \ \text{ s.t. } \ \Theta\Theta^\intercal = I_k, \tag{3.43}$$

where the terms $\|w_r - \Theta^\intercal v_r\|^2$ enforce the similarity across tasks by bringing them closer to the shared subspace. Problem (3.43) is solved using a two-step optimization, iterating between minimizing in $\{\Theta, \boldsymbol{v}\}$ and minimizing in $\boldsymbol{u}$. In Chen et al. (2009) the following extension is proposed:

$$\underset{\Theta \in \mathbb{R}^{k \times d}, \boldsymbol{w}, \boldsymbol{v}}{\arg\min} \sum_{r=1}^{T} \sum_{i=1}^{m_r} \ell(y_i^r, \langle w_r, x_i^r \rangle) + \lambda \sum_{r=1}^{T} \|w_r - \Theta^\intercal v_r\|^2 + \mu \sum_{r=1}^{T} \|w_r\|^2 \ \text{ s.t. } \ \Theta\Theta^\intercal = I_k.$$
$$\tag{3.44}$$

Moreover it is shown that (3.44), when relaxing the orthogonality constraint, can be expressed as a convex minimization problem.

A different approach relies on the use of the trace norm of the matrix W. This norm penalizes $\sum_{i=1}^{d} \lambda_i(W)^2$, where $\lambda_i(W)$ are the eigenvalues of W, and thus, forcing W to be low-rank. In the work of Pong et al. (2010) new formulations for problems with this trace norm penalty and a primal-dual method for solving the problem are developed. A modification of the trace norm can be found in Han and Zhang (2016), where a capped-trace norm is defined as $\sum_{i=1}^{d} \min(\theta, \lambda_i(W)^2)$. This capped norm, like the capped-L_p norm, can enforce a lower rank matrix for small θ and also degenerates to the trace norm for large enough θ.

Task-Relation Learning approaches

In other approaches, like the feature learning approach or the low-rank approach, the assumption is that all task parameters share the same subspace, which may be detrimental when there exists a negative or neutral transfer. The task-relation learning approach aims to find the pairwise dependencies among tasks and to possibly model positive, neutral and negative transfers between tasks.

One of the first works with the goal of explicitly modelling the pairwise task-relations is Bonilla et al. (2007), where a MT Gaussian Process (GP) formulation is presented. Assuming a Gaussian noise model $y_i^r \sim N\left(f_r(x_i^r), \sigma_r^2\right)$, Bonilla *et al.* place a GP prior over the latent functions f_r to induce correlation among tasks:

$$\boldsymbol{f} \sim N\left(\mathbf{0}_{dT}, K^f \otimes K_\theta^{\mathcal{X}}\right),$$

where $\boldsymbol{f} = (f_1, \ldots, f_T)$, K^f is the inter-task covariance matrix, $K_\theta^{\mathcal{X}}$ the feature covariance matrix, and $A \otimes B$ is the Kronecker product of two matrices, so $\text{Cov}(f_r(x), f_s(x')) = K_{rs}^f k_\theta^{\mathcal{X}}(x, x')$. That is, instead of assuming a block-diagonal covariance matrix for $\boldsymbol{f}$ as in previous works of Joint Learning (Lawrence and Platt, 2004), the authors in Bonilla et al. (2007) model the covariance as a product of inter-task covariance and inter-feature covariance. The inference of this model can be done using the standard GP inference for the mean and the variance of the prediction distribution. However, the interest resides in learning the task-covariance matrix K^f, but this leads to a non-convex problem. The authors propose a low-rank approximation of K^f, which weakens its expressive power. To overcome this disadvantage, in Zhang and Yeung (2010, 2013), using the idea of the Multi-Task GP, they consider linear models $f(x_i^r) = \langle w_r, x_i^r \rangle + b_r$ and the prior on matrix $W = (w_1 \ldots, w_T)$ is defined as

$$W \sim \left(\prod_{r=1}^{T} N\left(\mathbf{0}_d, \sigma_r^2 I_d\right)\right) \mathcal{MN}\left(0_{d \times m}, I_d \otimes \Omega\right)$$

where $\mathcal{MN}\left(M, A \otimes B\right)$ denotes the matrix-variate normal distribution with mean M, row covariance matrix A and column covariance matrix B. It is shown that the problem of selecting the maximum a posteriori estimation of W and the maximum likelihood estimations of Ω and biases b_r, $r = 1, \ldots, T$, is a regularized minimization problem that, when relaxing the restrictions on Ω, can be expressed as

$$\underset{\Omega \in \mathbb{R}^{d \times T}, W, \boldsymbol{b}}{\arg\min} \sum_{r=1}^{T} \sum_{i=1}^{m_r} \ell(y_i^r, \langle w_r, x_i^r \rangle + b_r) + \lambda \sum_{r=1}^{T} \|w_r\|^2 + \mu \sum_{r,s=1}^{T} \left(\Omega^{-1}\right)_{rs} \langle w_r, w_s \rangle \tag{3.45}$$

s.t. $\Omega \succeq 0, \text{tr}\,\Omega = 1$.

Other approaches like Argyriou et al. (2013) reach a similar problem from other perspective. Argyriou *et al.* assume a representation of the structure of the tasks as a graph; then the graph Laplacian L in the optimization problem can incorporate the knowledge about the task structure, as shown in Evgeniou et al. (2005):

$$\underset{W, \boldsymbol{b}}{\arg\min} \sum_{r=1}^{T} \sum_{i=1}^{m_r} \ell(y_i^r, \langle w_r, x_i^r \rangle + b_r) + \mu \sum_{r,s=1}^{T} \left(L\right)_{rs} \langle w_r, w_s \rangle, \tag{3.46}$$

where μ is a tuning parameter. Here the regularization can be also written using the adjacency matrix A as

$$\sum_{r,s=1}^{T} (L)_{rs} \langle w_r, w_s \rangle = \sum_{r,s=1}^{T} (A)_{rs} \|w_r - w_s\|^2.$$

The goal of Argyriou et al. (2013) and Zhang and Yeung (2010) is to jointly learn the task parameters and the tasks relations. In both cases, they opt for a two-step optimization: one step to learn the task parameters and other to learn the task relations. In the first step, according to Evgeniou et al. (2005), the problem (3.46) can be solved in the dual space using a Multi-Task kernel

$$k_L(x_i^r, x_j^s) = (L + \lambda I)_{rs}^{-1} k(x_i^r, x_j^s).$$

In Zhang and Yeung (2010) the authors propose the use of the trace norm to enforce a low-rank task-covariance matrix Ω, which leads to a closed solution. In Argyriou et al. (2013) they want to learn a matrix L that is a valid graph Laplacian, so they propose the following problem in the dual space:

$$\begin{aligned} \arg\min_{\boldsymbol{\alpha}, L} \; & \boldsymbol{\alpha}^{\mathsf{T}} K_L \boldsymbol{\alpha} + \nu \boldsymbol{\alpha}^{\mathsf{T}} \boldsymbol{y} + \operatorname{tr} (L + \lambda I)^{-1} \\ \text{s.t. } & 0 \preceq L, \; (L + \lambda I)_{\text{off}} \leq 0, \; (L + \lambda I)^{-1} \mathbf{1}_n = \frac{1}{\lambda} \mathbf{1}_n, \end{aligned} \tag{3.47}$$

where A_{off} denotes the off diagonal entries of A and $\mathbf{1}_n$ is the vector of n ones. The restrictions of problem (3.47) are to ensure that L is a valid graph Laplacian: it is positive definite, with non-positive terms outside of diagonal and the rows add up to 1; and the trace term in the objective function is to force L to be low-rank so it is easier to find clusters of tasks. The objective function is not jointly convex but is convex in each $\boldsymbol{\alpha}$ and L when we fix the other. For the step to optimize the Laplacian matrix the authors develop an algorithm involving several projection steps using proximal operators. Another work focused on learning the task-relations is Dinuzzo (2013), where the approach is a learning problem in an RKHS of vector-valued functions $g : \mathcal{X} \to \mathbb{R}^T$, and the associated reproducing kernel is:

$$H(x_1, x_2) = K_{\mathcal{X}}(x_1, x_2) \cdot L$$

and L is a symmetric positive matrix called the *output* kernel. That is, the elements of such RKHS are not real-valued functions but vector-valued functions, where each element of the vector corresponds to a different task.

Task Clustering approaches

The Task Clustering approach tries to find C clusters or groups among the original set of T tasks. Usually, the goal is to learn jointly only the tasks in the same cluster, so no negative transfer takes place. The first clustering approach (Thrun and O'Sullivan, 1996) divides the optimization process in two separate steps: independently learning the task-parameters and jointly learning the clusters of tasks. Using models that involve distances among points, e.g. kernel methods, they define for each task $r = 1, \ldots, T$, the distance

$$\operatorname{dist}_{\boldsymbol{\omega}_r}(x, x') = \sqrt{\sum_{i=1}^{d} \omega_r^i (x^i - x'^i)^2}.$$

That is, $\boldsymbol{\omega}_r$ parametrizes a distance with a different weight for each feature. Then, for each task an optimal weights vector $\boldsymbol{\omega}_r^*$ is computed minimizing the distance between examples of the same class and maximizing the distance among different classes:

$$A_r(\boldsymbol{\omega}_r) = \sum_{s=1}^{T} \delta_{r,s} \sum_{i=1}^{m_r} \sum_{j=1}^{m_s} (y_i^r y_j^s) \mathrm{dist}_{\boldsymbol{\omega}_r}(x_i^r, x_j^r),$$

where $y_i^r \in \{-1, 1\}$ is the class of pattern x_i^r, and δ_{rs} is 1 if $r = s$ and -1 otherwise. After the computation of the optimal parameters $\boldsymbol{\omega}_r^*$, the empirical loss on task r of a model fitted on data of task r using a distance parametrized by $\boldsymbol{\omega}_s^*$ is defined as e_{rs}. Then, the goal is to find clusters B_κ with $\kappa = 1, \ldots, C$, minimizing

$$J(C) = \sum_{\kappa=1}^{C} \sum_{r \in B_\kappa} \frac{1}{|B_r|} \sum_{s \in B_\kappa} e_{rs},$$

where the number of clusters C with minimum $J(C)$ is selected. That is, the clusters are selected using the results of independently trained tasks but using distances whose parameters are optimal for other tasks, then the transfer is done through the parameters $\boldsymbol{\omega}_r$.

Some proposals have also been made using regularization approaches. In Jacob et al. (2008), a problem based on Evgeniou and Pontil (2004) is proposed. Considering the $T \times T$ constant matrix $U = \frac{1}{T}\mathbf{1}\mathbf{1}^\intercal$, E the $T \times C$ cluster assignment binary matrix, and defining the adjacency matrix $M = E(E^\intercal E)^{-1}E^\intercal$, the problem is

$$\arg\min_{W} \sum_{r=1}^{T} \sum_{i=1}^{m_r} \ell(y_i^r, \langle w_r, x_i^r \rangle) + \lambda(\mu_{\mathrm{m}}\Omega_{\mathrm{m}}(W) + \mu_{\mathrm{b}}\Omega_{\mathrm{b}}(W) + \mu_{\mathrm{w}}\Omega_{\mathrm{w}}(W)), \qquad (3.48)$$

where $\Omega_{\mathrm{m}} = \mathrm{tr}\,(WUW^\intercal)$ is the mean regularization, $\Omega_{\mathrm{b}} = \mathrm{tr}\,(W(M-U)W^\intercal)$ is the inter-cluster variance regularization and $\Omega_{\mathrm{w}} = \mathrm{tr}\,(W(I-M)W^\intercal)$ is the intra-cluster variance regularization. This problem cannot be solved using the results of Evgeniou et al. (2005) because the regularization used is not convex, so a convex relaxation is needed.

A similar approach is presented in Kang et al. (2011) where, using the results from Argyriou et al. (2008), they propose a trace norm regularizer of the matrices $W_\kappa = WQ_\kappa$, where Q_κ are $T \times T$ binary, diagonal matrices where the r-th element of the diagonal indicates whether task r corresponds to cluster κ. They consider the problem

$$\begin{aligned} \arg\min_{W, Q_1, \ldots, Q_T, \boldsymbol{b}} \quad & \sum_{r=1}^{T} \sum_{i=1}^{m_r} \ell(y_i^r, \langle w_r, x_i^r \rangle + b_r) + \lambda \sum_{\kappa=1}^{C} \|WQ_\kappa\|_*^2 \\ \text{s.t.} \quad & \sum_{\kappa=1}^{C} Q_\kappa = I \text{ with } Q_{\kappa t} \in \{0, 1\}. \end{aligned} \qquad (3.49)$$

Here, the trace norm acts on each cluster, so it enforces that the matrices of vectors w_r in the same cluster have low-rank. This can be seen as a clusterized version of Multi-Task Feature Learning (Argyriou et al., 2006, 2008), shown in Equation (3.33), but instead of assuming that all tasks share the same subspace, only the tasks in the same cluster do; however, the explicit learned features cannot be recovered. The optimization of matrices Q_κ on problem (3.49) is done by relaxing the binary nature of matrices Q_κ such that

$0 \le Q_{\kappa t} \le 1$, and reparametrizing them as

$$Q_\kappa[r,r] = \frac{\exp(\alpha_{\kappa r})}{\sum_{g=1}^{C} \exp(\alpha_{gr})};$$

then they perform gradient descent over the $\alpha_{\kappa r}$ of the regularizer $\|WQ_\kappa\|_*^2$.

Another approximation to clusterized MTL is provided in Crammer and Mansour (2012), where a two-step procedure is described as follows. Considering that C initial clusters are fixed containing the T tasks, then two steps are repeated. First C single task models f_κ are fitted using the pooled data from tasks in cluster κ. Secondly each task r is assigned to the cluster κ whose function f_κ obtains the lowest error in task r. The proposal of Barzilai and Crammer (2015) takes the idea of the cluster assignation step from Crammer and Mansour (2012) and is also inspired by the sparse coding work of Kumar and Daumé III (2012). In this work the weights matrix is $W = DG$, where $D \in \mathbb{R}^{d \times C}$ contains as columns the hypotheses for each cluster and $G \in \mathbb{R}^{C \times T}$ is the task assignment matrix, that is, $G_{\kappa r} \in \{0,1\}$ and $\|G_r\|^2 = 1$. The corresponding optimization problem is

$$\underset{D \in \mathbb{R}^{d \times C}, G \in \mathbb{R}^{C \times T}}{\arg\min} \sum_{r=1}^{T} \sum_{i=1}^{m_r} \ell(y_i^r, \langle Dg_r, x_i^r \rangle) + \lambda \|D\|_{2,1} \tag{3.50}$$
$$\text{s.t. } G_{\kappa r} \in [0,1], \; \|G_r\|^2 = 1,$$

where the constraints on $G_{\kappa r}$ have been relaxed to be in the $[0,1]$ interval in order to make problem (3.50) convex. Observe that (3.50) is similar to (3.39) but different restrictions are used to ensure that G can be seen as a clustering assignment matrix.

Decomposition approaches

The decomposition approaches consider that the assumption that task parameters reside in the same subspace, or that the parameter matrix W, composed by each task parameters w_r as its columns, is low-rank, is too restrictive for real world scenarios. The proposition is then to decompose the parameter matrix in the sum of two matrices, i.e. $W = U + V$, where usually U captures the shared properties of the tasks and V accounts for the information that cannot be shared among tasks. These models also receive the name of *dirty models* because they assume that the data is *dirty* and cannot be constrained to rigid subspaces. The optimization problem is

$$\underset{U,V \in \mathbb{R}^{d \times T}}{\arg\min} \sum_{r=1}^{T} \sum_{i=1}^{m_r} \ell(y_i^r, \langle u_r + v_r, x_i^r \rangle) + \lambda g(U) + \mu h(V), \tag{3.51}$$

where $g(U)$ and $h(V)$ are different regularizers for U and V, respectively. In Jalali et al. (2010) the authors use $g(U) = \|U\|_{1,\infty}$ to enforce block-sparsity and $h(V) = \|V\|_{1,1}$ to enforce element-sparsity. In Chen et al. (2010) $g(U) = \|U\|_*$ is used to enforce low-rank while maintaining $h(V) = \|V\|_{1,1}$. In Chen et al. (2011) both regularizers seek properties shared among all tasks, $g(U) = \|U\|_*$ to enforce a low-rank and $h(V) = \|V\|_{1,2}$ for row-sparsity. In Gong et al. (2012b) they propose $g(U) = \|U\|_{1,2}$ to enforce row-sparsity, i.e. the tasks share a common subspace; and $h(V) = \|V^\intercal\|_{1,2}$ which penalizes the orthogonal parts to the common subspace of task-parameter; the authors affirm that it penalizes outlier tasks.

Other approaches generalize the decomposition method by assuming that the parameter matrix can be expressed as $W = \sum_{l=1}^{L} W_l$; then the problem to solve has the form

$$\underset{W_1,\dots,W_L \in \mathbb{R}^{d \times T}}{\arg\min} \sum_{r=1}^{T} \sum_{i=1}^{m_r} \ell\left(y_i^r, \left\langle \sum_{l=1}^{L} (W_l)_r, x_i^r \right\rangle\right) + \sum_{l=1}^{L} \lambda_l r(W_l). \tag{3.52}$$

In Zweig and Weinshall (2013) the regularizer used is $r(W_l) = \|W_l\|_{2,1} + \|W_l\|_{1,1}$ to enforce the row and element-sparsity. In Han and Zhang (2015) the regularizer is $r(W_l) = \sum_{r,s=1}^{T} \|(W_l)_r - (W_l)_s\|^2$ which, alongside some constraints, allows to build a tree of task groups, where the root contains all the tasks and the leafs only correspond to one task.

3.4.3 Combination-based

The combination-based methods use a combination of task-specific part and a common part shared by all tasks. These two parts are learned simultaneously with the goal of leveraging the common and specific information.

The first proposal, which uses the SVM as the base model, is found in Evgeniou and Pontil (2004). The goal is to find a decision function for each task, defined by a vector $w_r = w + v_r$ and a bias b_r. Here w is common to all tasks and v_r is task-specific. Instead of imposing some restrictions such as low-rank or inter-task regularization the idea is to impose the coupling by directly placing a model w that is common to all tasks. The v_r part is added so each model can be adapted to the specific task.

Multiple extensions of the work of Evgeniou and Pontil (2004) have been presented: in Li et al. (2015); Xu et al. (2014) the method is extended to the Proximal SVM (Fung and Mangasarian, 2001) and Least Squares SVM (Suykens and Vandewalle, 1999), respectively. Also, in Parameswaran and Weinberger (2010) the idea is adapted for the Large Margin Nearest Neighbor model (Weinberger and Saul, 2009). However, in this work we are interested mainly in two extensions: one is the work of Evgeniou et al. (2005) and the other is developed in Liang and Cherkassky (2008) and in Cai and Cherkassky (2009); both will be described in detail in Chapter 4. The original problem presented in Evgeniou and Pontil (2004) is

$$\underset{w,V}{\arg\min} \, C \sum_{r=1}^{T} \sum_{i=1}^{m_r} \ell(y_i^r, \langle w + v_r, x_i^r \rangle) + \frac{\mu}{2} \|w\|^2 + \frac{1}{2} \sum_{r=1}^{T} \|v_r\|^2, \tag{3.53}$$

where μ is a tradeoff parameter to leverage the common and task-specific information. When the dual problem of (3.53) is obtained, the result is a problem where the kernel encodes this combination of common and shared parts; that is, the kernel function is

$$k(x_i^r, x_j^s) = \frac{1}{\mu} \langle x_i^r, x_j^s \rangle + \delta_{rs} \langle x_i^r, x_j^s \rangle,$$

where δ_{rs} is 1 only if r and s are the same task, so it encodes the specific part of the model. This is equivalent to a feature extension strategy, like the domain adaptation scheme proposed in Daumé III (2007) with only target and source domains, but that can be easily adapted to MTL.

3.4.4 Multi-Task Learning with Neural Networks

Deep learning has experimented an enormous development over the last decade, with multiple variants having great success in many fields and applications such as Convolutional Neural Networks for image recognition, Generative Adversarial Networks for generative models or Transformers for Natural Language Processing. The feature learning process, natural to neural networks, is usually the main idea behind MTL strategies, but there are also other proposals. In this subsection we explore some of these strategies and their connection with other methods used in kernel or linear models.

As proposed in Ruder (2017) we can divide the Multi-Task approaches with neural networks in hard sharing and soft sharing. The hard sharing approach is the most common one, and it consists in sharing between all tasks the hidden layers of the network while keeping some task-specific layers at the end of the network. The first example dates back to Caruana (1997) where only the output layers are task-specific. The goal of this approach is to learn a set of features that is useful for all tasks simultaneously, so the transfer of information between tasks is done in this feature building process. Although this approach has some theoretical properties to improve learning (Baxter, 2000), it is very rigid since it assumes that the learned features and the feature learning process must be the same for all tasks.

The soft sharing approach to MTL tries to tackle these problems with different strategies. In Cavallanti et al. (2010) the authors propose to learn one perceptron for each task and the update of each perceptron is determined by an interaction matrix A. Given a pair (x_i^s, y_i^s), the weights of task r are updated when a mistake is committed on task s and the update is scaled according to the position r, s of matrix A^{-1}:

$$w_r(\tau + 1) = w_r(\tau) - \nu y_i^s A_{rs}^{-1} x_i^s, \tag{3.54}$$

where w_r are the parameters of the perceptron corresponding to task r. This approach only uses perceptrons, so its expressive power is quite limited. Also, the matrix A has to be defined a priori and is not learned from the data. In Long and Wang (2015) they propose a model to overcome these limitations. They use a multi-task architecture for computer vision with multiple shared layers and multiple task-specific layers. Moreover, they use a Bayesian approach to learn the task relationships. They place a tensor prior over the network parameters such that in each specific layer l, there are T matrices W_r^l for $r = 1, \ldots, T$, and if we denote as $\mathcal{W}^l$ the vector (tensor) of such matrices, then

$$\mathcal{W}^l \sim \mathcal{TN}\left(0, \Sigma_1, \Sigma_2, \Sigma_3\right),$$

where $\mathcal{TN}$ is the tensorial normal distribution, and $\Sigma_1, \Sigma_2, \Sigma_3$ model the feature-covariance, class-covariance (in the classification layer) and task-covariance, respectively. The parameters $\mathcal{W}^l$ are learned automatically by using gradient descent and for the covariance matrices a Maximum Likelihood Estimator is used after each epoch. If we take a look at the update rule for the network parameters

$$W_r^l(\tau + 1) = W_r^l(\tau) - \nu \left(\frac{\partial \ell(f_r(x_i^r), y_i^r)}{\partial W_r^l} + \left[(\Sigma_1 \otimes \Sigma_2 \otimes \Sigma_3)^{-1} \text{vect}\left(\mathcal{W}^l\right)_{:,:,r}\right]\right),$$

where ℓ is the loss function, we can observe some similarities with the update (3.54), where the inverse of the Kronecker product of covariances models how each task affects the others. Although this approach learns the matrix relations, the architecture is still restrictive since it assumes that all tasks can be modelled using the same network

architecture. In Misra et al. (2016) they propose a strategy named "Cross-stitch" networks which uses one network for each task, but these networks are connected using a linear combination of the outputs of every layer, including the hidden ones. Consider an illustrative example with an MTL network with for tasks 1 and 2, if h_1^l, h_2^l are the output values of the l-th layers of the networks of tasks are combined as

$$\begin{bmatrix} \tilde{h}_1^l \\ \tilde{h}_2^l \end{bmatrix} = A^l \begin{bmatrix} h_1^l \\ h_2^l \end{bmatrix} = \begin{bmatrix} \alpha_{11}^l & \alpha_{12}^l \\ \alpha_{21}^l & \alpha_{22}^l \end{bmatrix} \begin{bmatrix} h_1^l \\ h_2^l \end{bmatrix}, \tag{3.55}$$

where the 2×2 matrix A^l is a "cross-stitch" unit and defines the linear combination to compute $\tilde{h}_1^l, \tilde{h}_2^l$ which will be the input values for the $(l+1)$-th layer. The network parameters and the "cross-stitch" values α_{rs}^l can both be learned using backpropagation. If $A^l = I_{2 \times 2}$ in all layers this is equivalent to two independent networks, one for each task, and using constant matrices as "cross-stitch" units results in two identical common networks. This strategy is extended in Ruder et al. (2017), where the authors include two modifications: network parameters of each network are divided in two spaces, each expecting to capture different properties of the data, and there are learnable skip-connections for each task. The first modification consists on using two spaces for each task, which implies that the number of parameters is doubled, that is, independently from the number of tasks, in each task-specific network and in each layer l there are two outputs for each task: $h_{r,a}^l, h_{r,b}^l$, each obtained with a different parameter matrix $W_{r,a}^l, W_{r,b}^l$, both with the same dimensions. Then, in the illustrative case of two tasks, as the combination shown in (3.55), A is a 4×4 matrix, because each $\alpha_{i,j}^l$ is a 2×2 matrix for $i, j \in \{1, 2\}$. The matrix A not only determines how the tasks are related but how each space of each task is related with the rest. To enforce that each space captures different properties they use the regularizers

$$\left\| (W_{r,a}^l)^\intercal W_{r,b}^l \right\|_{2,2}^2,$$

where the $L_{2,2}$ (Frobenius) norm is used, so the parameters matrices $W_{r,a}^l$ and $W_{r,b}^l$ span orthogonal spaces. Particular values of the matrices A^l can result in a combination-based approach where there is a shared common model and task-specific ones. The skip-connections are reflected in a final layer in each network that receives as input the linear combination of the activation values $h_{r,1}^l$ and $h_{r,2}^l$ for every layer l. This linear combination uses learnable parameters β_r for each task.

Other approaches consider tensor-based methods instead of the "cross-stitch" networks to learn the parameters of each task-oriented network and the degree of sharing between tasks from the data. In Yang and Hospedales (2017a), the authors propose a tensor-generative strategy to model the parameters of each layer. If W_r^l is the $d_1^l \times d_2^l$ parameter matrix for task r in layer l, in each layer we can consider the collection of such matrices as a $d_1^l \times d_2^l \times T$ tensor $\mathcal{W}^l$. We can build these tensors from other pieces in different ways; for example consider a $d_1^l \times d_2^l \times K$ tensor $\mathcal{L}$, that is, a collection of K matrices of dimension $d_1^l \times d_2^l$. Also . Now, consider the $K \times T$ matrix S, then

$$W_r^l = \mathcal{W}_{:,:,r} = \sum_{\kappa=1}^{K} \mathcal{L}_{:,:,\kappa} S_{\kappa,r}.$$

That is, all task parameter matrices W_r^l are linear combinations of the matrices collected

in $\mathcal{L}$. Observe that this is a generalization of the sparse coding scheme presented in Daumé III (2009) and shown in equation (3.38). Since all the strategies to build $\mathcal{W}$ from other pieces are based on tensor products, the whole process is differentiable and all those pieces can be learned using gradient descent. The other tensor-based proposal is presented in Yang and Hospedales (2017b) where, instead of a tensor-building approach, they consider the tensors $\mathcal{W}^l$ as the collection of matrices W_r^l and use tensor-trace norms to enforce the coupling between different tasks. Since the tensor-trace norms are not differentiable, they use the subgradient for the backpropagation during training. This approach can be categorized as low-rank, that is, a parameter-based approach according to our taxonomy.

All the previous approaches consider the same architecture for every task. Although the skip-connections can alleviate this, the sharing of information is still made in a layer-wise manner, so the same architecture has to be considered for every network. In Sun et al. (2020) they propose a single network with L layers and with skip-connections between all layers, so each task can use a specific policy. That is, task 1 can use the first, third and fourth layer while task 2 might use the second and fourth only. In this example, the first and third are specific for task 1, the second layer is specific for task 2 while the fourth is a shared layer. In general, there is a binary $L \times T$ policies matrix U that determines which layers are used for each task. The problem is then a bi-level optimization

$$\min_U \min_{W_1,\ldots,W_L} \ell(U, W_1, \ldots, W_L).$$

The optimization with respect the network parameters is done using back-propagation, while the optimization with respect to U uses a specific algorithm.

3.4.5 Multi-Task Learning with Kernel Methods

Kernel methods are central in ML, they have been successful in many areas, and its principal appeal is that the original features are implicitly transformed to a kernel space, possibly infinite-dimensional, where the problems of classification or regression for example are usually easier to solve. Unlike deep learning models, the kernel features used are not learnable, but are implicitly defined a priori by the kernel function chosen, such as Gaussian, polynomial or Matérn kernels. This fact makes it more difficult to use a feature-based MTL approach with kernel models. Instead, other proposals have been made, in this subsection we review the most relevant ones, such as graph Laplacian regularization or joint learning of common and task-specific parts.

One important approximation to MTL with kernel models is learning vector-valued functions, in which the target space is not scalar but a vector one. That is, the sample data X, Y are pairs $(x_i, \boldsymbol{y}_i)$ where $x_i \in \mathcal{X}$, e.g. $\mathcal{X} = \mathbb{R}^d$ and $\boldsymbol{y}_i \in \mathcal{Y}^T$. This approach finds its motivation in multi-output learning, where there are multiple targets for each input. Using kernel models the multi-output regularized risk is

$$\sum_{i=1}^n \ell(W^\mathsf{T}\phi(x_i) + \boldsymbol{b}, \boldsymbol{y}_i) + \mu\Omega(W), \tag{3.56}$$

where W is the matrix with T columns, one for each output and $\boldsymbol{b}$ is the vector of corresponding biases. A commonly used loss function ℓ is

$$\ell(W^\mathsf{T}\phi(x_i) + \boldsymbol{b}, \boldsymbol{y}_i) = \|W^\mathsf{T}\phi(x_i) + \boldsymbol{b} - \boldsymbol{y}_i\|_2^2$$

and Ω is a regularizer for matrix W that can enforce different behaviours; for example the trace norm regularizer is used for enforcing a low-rank matrix. Micchelli and Pontil (2004, 2005) develop the theory for operator-valued kernels, that are the kernel functions corresponding to vector-valued functions. They propose an extension of the representer theorem for operator-valued kernels when a Tikhonov regularization for vector-valued functions is used. Although these are interesting results, they do not provide explicit algorithms to learn such functions. Moreover, MTL is not that well suited for this formulation, since for each data x_i^r there is a single scalar y_i^r. The multi-task regularized risk using this formulation is

$$\sum_{i=1}^{n} \ell(A \odot W^\mathsf{T}\phi(x_i) + \boldsymbol{b}, \boldsymbol{y}_i) + \mu\Omega(W), \tag{3.57}$$

where A is a binary matrix with the same sparsity pattern as Y, and $\odot$ is the element-wise multiplication.

Other approach that seems better suited for kernel methods is a combination-based one, first shown in Evgeniou and Pontil (2004), which can be seen in (3.53), and then extended in Cai and Cherkassky (2009, 2012). The kernelized optimization problem is

$$\underset{w,V}{\arg\min} \sum_{r=1}^{T} \sum_{i=1}^{m_r} \ell(y_i^r, \langle w, \phi(x_i^r) \rangle + \langle v_r, \phi_r(x_i^r) \rangle) + \lambda(\mu \left\| w \right\|^2 + \operatorname{tr}(V^\mathsf{T}V)). \tag{3.58}$$

With this formulation, it is possible to use different implicit transformations for the common part ϕ and for each specific part ϕ_r. This can be done because the coupling between tasks is not made using some restrictions in some shared space, but using a common model for all tasks. Similary to the representer theorem, it can be shown that we have

$$w = \sum_{s=1}^{T} \sum_{i=1}^{m_s} \alpha_i^s \phi(x_i^s), \ v_r = \sum_{i=1}^{m_r} \alpha_i^r \phi_r(x_i^r),$$

where the dual coefficients α_i^r are shared, so w is a combination of all inputs from all tasks, while the specific parts v_r are only dependent on samples from the corresponding task. The fact that different spaces can be used for the common and specific parts make this formulation very flexible.

A feature-based strategy was also presented in Argyriou et al. (2008), where the linear optimization problem is given in (3.33), and the kernel extension is

$$\underset{U \in \mathcal{L}(\mathcal{H}, \mathbb{R}^d), A \in \mathbb{R}^{d \times T}}{\arg\min} \sum_{r=1}^{T} \sum_{i=1}^{m_r} \ell(y_i^r, \langle Ua_r, \phi(x_i^r) \rangle) + \lambda \left\| A \right\|_{2,1}^2 \text{ s.t. } U^\mathsf{T}U = I_d, \tag{3.59}$$

where ϕ is the implicit transformation into the kernel space $\mathcal{H}$ and U is a linear operator between $\mathcal{H}$ and $\mathbb{R}^d$. To solve this they apply the same procedure than the one used in the linear case, where they obtain a trace-norm regularized problem

$$\underset{W \in \mathbb{R}^{d \times T}}{\arg\min} \sum_{r=1}^{T} \sum_{i=1}^{m_r} \ell(y_i^r, \langle w_r, \phi(x_i^r) \rangle) + \lambda \left\| W \right\|_*^2, \tag{3.60}$$

and they propose a representer theorem for this problem, namely

$$w_r = \sum_{s=1}^{T} \sum_{i=1}^{m_s} \alpha_i^s(r) \phi(x_i^s).$$

The authors also develop an algorithm to solve problem (3.60) using this result. A clusterized extension is presented in Kang et al. (2011) whose corresponding problem is shown in (3.49).

The task relation learning is also a common approach to MTL with kernel models. Using a Gaussian Process formulation, different strategies to learn the task-covariance matrix are proposed in Bonilla et al. (2007); Lawrence and Platt (2004).

Finally, one important result for MTL with kernels is given in Evgeniou et al. (2005), where a general MTL formulation with kernel methods is presented. Given a kernelized problem

$$\sum_{r=1}^{T} \sum_{i=1}^{m} \ell(y_i^r, \langle w_r, \phi(x_i^r) \rangle) + \mu(\text{vec } W)^{\mathsf{T}}(E \otimes I)(\text{vec } W), \tag{3.61}$$

where E is a $T \times T$ matrix and we use $\text{vec } W$ for the vectorized matrix W, i.e. the vector $(W_1^{\mathsf{T}}, \ldots, W_T^{\mathsf{T}})^{\mathsf{T}}$ where $W_1, \ldots, W_T$ are the columns of W, then the solution can be expressed as

$$\text{vec } W = \sum_{r=1}^{T} \sum_{i=1}^{m} \alpha_i^r B_r \phi(x_i^r),$$

with B_r being the columns of a matrix B such that $E = (B^{\mathsf{T}}B)^{-1}$. Then, as shown in Subsection 3.3.3, this is equivalent to using a kernel

$$\hat{k}(x_i^r, x_j^s) = (E^{-1})_{rs} k(x_i^r, x_j^s).$$

This result is used in latter works, such as Zhang and Yeung (2010) and Argyriou et al. (2013), where they propose using an inter-task regularization that penalizes

$$\sum_{r,s=1}^{T} (\Theta)_{rs} \langle w_r, w_s \rangle = (\text{vec } W)^{\mathsf{T}}(\Theta \otimes I)(\text{vec } W),$$

where Θ is the inter-task relationship matrix and they propose different strategies to learn such matrix.

3.5 Multi-Task Problems

TODO: Motivar sección The concept of a Multi-Task problem is not clear because different definitions have been considered in the literature. In this thesis, only supervised problems will be considered, so we will refer to them just as problems, and we will omit the discussion about unsupervised ones. In this section, we try to shed some light on this topic and give some definitions for MTL problems.

Multiple kind of problems can be faced with a MTL approach. The most common definition of Multi-Task problem is a homogeneous setting, where all tasks are sampled from the same space. That is, we have a space $\mathcal{X} \times \mathcal{Y}$ where $\mathcal{X}$ is the feature space and $\mathcal{Y}$ is the target space, which can be $\mathbb{R}$ in the regression case or $\{0,1\}$ in the binary classification one, and each task has a possibly different distribution $P_r(x, y)$ over this

shared space. In this type of problems, all the tasks have the same number of features and the same target space, that is, every task is either a regression or classification problem, there cannot be a mix of regression and classification tasks. There are other heterogeneous definitions where each task can be sampled from a different space $\mathcal{X}_r \times \mathcal{Y}_r$, and, therefore, a mix of regression or classification tasks can be considered. However, in this work we are only interested in the homogeneous case.

Within the homogeneous MTL problems we can consider the following cases, depending on the nature of each task and the task definition procedure:

- Pure Multi-Task Problems. These are problems where each task has been sampled from a possibly different distribution, so we have possibly different samples (X_r, Y_r).

  - MT single-reg: This is the case where each task is a regression problem with a single target, e.g. $\mathcal{Y} = \mathbb{R}$.

  - MT bin-clas: This is the case where each task is a binary classification problem, e.g. $\mathcal{Y} = \{0, 1\}$.

  - MT multi-reg: This is the case where each task is a regression problem with multiple targets, e.g. $\mathcal{Y} = \mathbb{R}^m$, where m is the number of targets.

  - MT multi-clas: This is the case where each task is a multi-class classification problem, e.g. $\mathcal{Y} = \{1, \ldots, C\}$, where C is the number of classes.

- Artificial Multi-Task Problems: These are problems that can be seen as Multi-Task and be solved using MTL strategies, but it is not the only way to solve them. The main difference is that although the target samples might be different across tasks, the set of features sampled is shared by all tasks, i.e. $(X_r, Y_r) = (X, Y_r)$.

  - multi-reg: This is the case where an m-target regression problem is converted into a multi-task problem by replicating m times the features X and using one of the targets on each repetition, so we have m single target regression problems, each considered a different task.

  - multi-clas: This is the case where a multi-class classification problem with C classes is converted into a multi-task problem by replicating C times the features X and considering a one-vs-all scheme for each repetition with a different positive class in each one, so we have C binary classification problems, each considered a different task.

TODO: tabla a apéndice
In Table 3.1 we show some of the most used MTL problems, exposing its characteristics and the papers that use them.

TABLE 3.1: MTL Problems. The columns show the number of samples n, the dimension d, the number of tasks T and the nature of the problem. TODO: cambiar formato de tabla para usar siutnix

Name	n	d	T	Nature	References
school	15362	27	139	MT single-reg	Evgeniou and Pontil (2004) Evgeniou et al. (2005) Argyriou et al. (2006, 2008, 2007) Bonilla et al. (2007) Zhang and Yeung (2010) Agarwal et al. (2010) Chen et al. (2011) Zhou et al. (2011) Gong et al. (2012b) Kumar and Daumé III (2012) Zhang and Yeung (2013) Han and Zhang (2016) Jeong and Jun (2018)
20-newsgroup	18000	t.v.	20	multi-clas	Ando and Zhang (2005) Daumé III (2009)
Reuters-RCV1	800000	t.v.	103	multi-clas	Yu et al. (2005) Ando and Zhang (2005)
computer	3600	13	180	MT single-reg	Argyriou et al. (2006, 2008, 2007) Evgeniou et al. (2005) Agarwal et al. (2010) Kumar and Daumé III (2012) Jeong and Jun (2018)
landmine	14820	10	29	MT bin-clas	Xue et al. (2007) Jebara (2011) Daumé III (2009) Jawanpuria and Nath (2012)
MHC-I	32302	184	47	MT bin-clas	Jacob et al. (2008) Jawanpuria and Nath (2012)
dermatology	366	33	6	multi-clas	Jebara (2004) Argyriou et al. (2008)
sentiment	2000	473856	4	MT bin-clas	Daumé III (2009) Zhang and Yeung (2010) Crammer and Mansour (2012) Zhang and Yeung (2013)

Table 3.1 – continued from previous page

Name	n	d	T	Nature	References
					Barzilai and Crammer (2015)
sarcos	44484	21	7	multi-reg	Zhang and Yeung (2010) Chen et al. (2011) Zhou et al. (2011) Jawanpuria and Nath (2012) Zhang and Yeung (2013) Ciliberto et al. (2015)
isolet	7797	617	5	MT multi-clas	Parameswaran and Weinberger (2010) Gong et al. (2012a)
mnist	70000	400	10	multi-clas	Kang et al. (2011) Kumar and Daumé III (2012) Zweig and Weinshall (2013) Jeong and Jun (2018)
usps	9298	256	10	multi-clas	Kang et al. (2011) Kumar and Daumé III (2012) Zweig and Weinshall (2013) Jeong and Jun (2018)
adni	675	306	6	MT single-reg	Gong et al. (2012a) Gong et al. (2012b)
microarray	131	21	19	multi-reg	Lozano and Swirszcz (2012) Han and Zhang (2016)
cifar10	50000	1024	10	multi-clas	Zweig and Weinshall (2013) Han and Zhang (2016)
parkinson	5875	19	42	MT single-reg	Jawanpuria and Nath (2012) Jeong and Jun (2018)

3.6 Conclusions

In this chapter, we covered. . .

A Convex Formulation for Multi-Task Learning

4.1 Introduction

As we have seen in Chapter 3, the MTL proposals can be categorized as feature-based, parameter-based and combination-based approaches. Feature-based proposals have the strategy of finding a shared representation of the original features that is benefitial for all tasks. Parameter-based strategies typically use multi-task regularization schemes that, using specific regularizers, push together the parameters of the task-specialized models. Finally, the combination-based strategies combine a common model and a task-specific one. In this chapter we will present a convex formulation for combination-based MTL. With this formulation, a general task-specialized model is defined as

$$h_r(\cdot) = \lambda_r g(\cdot) + (1 - \lambda_r) g_r(\cdot). \tag{4.1}$$

Here, we use the hyperparameters $\lambda_r \in [0, 1]$ to combine in a convex way the common part $g(\cdot)$ and specific parts $g_r(\cdot)$. This general formulation is very flexible, because we can use any family of functions to model $g(\cdot)$ or $g_r(\cdot)$. It is also easily interpretable, since $\lambda_r = 1$ for all $r = 1, \ldots, T$ results in a common model and $\lambda_r = 0$ in independent models for each task. All the values $\lambda_r \in (0, 1)$ correspond to pure MTL models, being closer to common model when λ_r is close to 1 and more specific when λ_r is close to 0. Given an MTL sample

$$\boldsymbol{z} = \bigcup_{r=1}^{T} \{(x_1^r, y_1^r), \ldots, (x_{m_r}^r, y_{m_r}^r)\},$$

the MTL regularized risk functional that corresponds to this convex formulation is

$$R_{\boldsymbol{z}}(g, g_1, \ldots, g_T) = C \sum_{r=1}^{T} \sum_{i=1}^{m_r} \ell(\lambda_r g(x_i^r) + (1 - \lambda_r) g_r(x_i^r), y_i^r) + \Omega(g) + \sum_{r=1}^{T} \Omega_r(g_r), \tag{4.2}$$

where $\Omega(g)$ and $\Omega_r(g_r)$ are the regularizers for the common and r-th task parts, respectively. Observe that, since the regularization is made independently for each part, no multi-task specific regularization scheme is needed. The task coupling is made directly in the definition of the model, which combines common and specific parts, and the risk $R_{\boldsymbol{z}}$ is optimized jointly in all parts. This has some advantages, for example, no additional

regularizations for coupling are needed, which usually are not convex, and there is no need to develop a new optimization procedure. In this chapter these characteristics will be presented in specific models using this convex MTL formulation.

A natural alternative is to combine independently trained models. That is, given a common model $g^*(\cdot)$ trained with data from all tasks, and task-specific models $g_r(\cdot)$ trained with only the data corresponding to their task, we can minimize

$$R_{\boldsymbol{z}}(\lambda) = C \sum_{r=1}^{T} \sum_{i=1}^{m_r} \ell(\lambda_r g^*(x_i^r) + (1 - \lambda_r) g_r^*(x_i^r), y_i^r). \tag{4.3}$$

Here, the models g^* and $g_1^*, \ldots, g_T^*$ are fixed, and the goal is to select the optimal parameters λ_r^* that minimize the training risk.

More specifically, it is shown how this formulation can be applied to two of the most popular model families: kernel methods, in Section 4.2, and neural networks, in Section 4.4. Then, in Section 4.5 a real-world application of the convex MTL to the prediction of renewable energy is presented.

4.2 Convex Multi-Task Learning with Kernel Methods

As explained in the previous chapters, kernel models offer many good properties such as an implicit transformation to a possibly infinite-dimensional space and the convexity of the problems that have to be solved for the training process. With these models, a regularized risk problem is solved. A general formulation of a training problem for kernel models is

$$\hat{R}_z(w) = \sum_{i=1}^{n} \ell(\langle w, \phi(x_i) \rangle + b, y_i) + \mu \Omega(w), \tag{4.4}$$

where z is the sample $\{(x_1, y_1), \ldots, (x_n, y_n)\}$ and $\Omega(w)$ is a regularizer for w, typically the L_2 norm: $\|w\|^2$. Observe that b, the bias term, is not regularized since it does not affect the capacity of the hypothesis space. In (4.4) ϕ is a fixed transformation function such that there exists a "kernel trick", that is a kernel function k for which

$$\langle \phi(x), \phi(y) \rangle = k(x, y).$$

Here, ϕ is an element of an RKHS, where k is the reproducing kernel. The Representer Theorem (Schölkopf et al., 2001) states that the optimal solution of problem (4.4) has the following form

$$w^* = \sum_{i=1}^{n} \alpha_i \phi(x_i),$$

where $\alpha_i \in \mathbb{R}$ are some coefficients. This means that our optimal solution w^* is also an element of the same RKHS. These models embrace the Structural Risk Minimization paradigm by limiting the capacity of the space of hypothesis, which is done by penalizing the L_2 norm of w. This is equivalent to limiting our space of candidates to vectors inside a ball of some fixed radius.

Multi-Task Learning with kernel models require imposing some kind of coupling between the models for each task in the learning process. The feature learning or feature sharing approach, which is usually adopted with neural networks, is not feasible when using kernel models, since the (implicit) transformation functions ϕ used are not learned but fixed, and determined by the choice of the kernel function. Therefore, other

strategies have to be developed. One approach to MTL with kernel models was developed in Evgeniou and Pontil (2004), and later extended in Cai and Cherkassky (2009, 2012), where the models for each task are defined as:

$$w_r = w + v_r,$$

where w is a common part, shared by all models, and v_r is a task-specific part. With this approximation, the transfer of information is performed by the common part w. The regularized risk that is minimized is

$$R_{\boldsymbol{z}}(w, v_1, \ldots, v_T) = \sum_{r=1}^{T} \sum_{i=1}^{m_r} \ell(\langle w, \phi(x_i^r) \rangle + \langle v_r^{\mathsf{T}}, \phi_r(x_i^r) \rangle + b_r, y_i^r) + \mu_c \left\| w \right\|^2 + \mu_s \sum_{r=1}^{T} \left\| v_r \right\|^2,$$
(4.5)

where μ_c and μ_s are the hyperparameters to control the regularization of the common and specific parts, respectively. Here $\boldsymbol{z}$ is an MTL sample

$$\boldsymbol{z} = \bigcup_{r=1}^{T} \{(x_1^r, y_1^r), \ldots, (x_{m_r}^r, y_{m_r}^r)\}.$$

Observe also that in (4.5) different transformations are used: the transformation ϕ corresponds to the common part of the model, while ϕ_r is task-specific. This is a joint learning approach that is developed for the L1-SVM, to which Evgeniou et al. give the name of *Regularized MTL*, but we will refer to it as the additive MTL approach. Observe that as $\frac{\mu_c}{\mu_s} \to \infty$, we would have a common part w that tends to zero, which would result in independent models for each task, i.e. $w_r \approx v_r$. On the contrary, when $\frac{\mu_c}{\mu_s} \to 0$, the task-specific parts tend to zero and every model is the common part, i.e. $w_r \approx w$. There are two asymptotical behaviours: the first one tends to an Independent-Task Learning (ITL) approach, while the second one tends to a Common-Task Learning (CTL) one. The MTL formulation is a strategy that lies between those two approaches, CTL and ITL, combining them to achieve a more flexible model.

The asymptotical properties of this approach offer an interpretation to understand the influence of each hyperparameter, but they are not easily applicable in practice. Using the formulation of (4.1), we define the common part as $g(x) = w^{\mathsf{T}} \phi(x) + b$ and the task-specific parts as $g_r(x) = v_r{}^{\mathsf{T}} \phi_r(x) + b_r$, so the MTL models are

$$h_r(x_i^r) = \lambda \left\{ w^{\mathsf{T}} \phi(x_i^r) + b \right\} + (1 - \lambda) \left\{ v_r^{\mathsf{T}} \phi_r(x_i^r) + d_r \right\},$$
(4.6)

where $\lambda \in [0, 1]$ is a hyperparameter and the corresponding regularized risk is

$$R_{\boldsymbol{z}}(w, v_1, \ldots, v_T) = \sum_{r=1}^{T} \sum_{i=1}^{m_r} \ell(\lambda \left\{ w^{\mathsf{T}} \phi(x_i^r) + b \right\} + (1 - \lambda) \left\{ v_r^{\mathsf{T}} \phi_r(x_i^r) + d_r \right\}, y_i^r)$$
$$+ \mu_c \left\| w \right\|^2 + \mu_s \sum_{r=1}^{T} \left\| v_r \right\|^2.$$

We will name this approach convex, in contrast to the additive approach of the original formulation. Here, the interpretation of λ is straightforward. The model with $\lambda = 1$ is equivalent to learning a single common model for all tasks, that is $w_r = w$. When $\lambda = 0$, the models for each task are completely independent: $w_r = v_r$. The convex

formulation also defines an MTL model that is between a CTL approach and an ITL one, but it presents an advantage over the additive one: the values of λ that recover the CTL and ITL approaches are known, and these values are attainable, that is, it is not an asymptotical behaviour as in the additive formulation. Moreover, the two formulations, the additive and convex, are equivalent within an L1-SVM setting, as shown next.

4.2.1 L1-SVM and Equivalence Results

The L1-SVM (Vapnik, 2000) is the original and most popular variant of the SVMs and is also the basis of the MTL formulation in Evgeniou and Pontil (2004). We will present the development for the additive approach and for the convex one using an L1-SVM setting. Then, we will show the equivalence between the two approaches and discuss their differences.

Additive MTL L1-SVM
The additive MTL primal problem formulation, presented in Evgeniou and Pontil (2004) and extended for task-specific biases in Cai and Cherkassky (2012), is

$$\begin{aligned}
\operatorname*{arg\,min}_{w,\boldsymbol{v},b,\boldsymbol{d},\boldsymbol{\xi}} \quad & J(w,\boldsymbol{v},b,\boldsymbol{d},\boldsymbol{\xi}) = C\sum_{r=1}^{T}\sum_{i=1}^{m_r}\xi_i^r + \frac{1}{2}\sum_{r=1}^{T}\|v_r\|^2 + \frac{\mu}{2}\|w\|^2 \\
\text{s.t.} \quad & y_i^r\left(\langle w,\phi(x_i^r)\rangle + \langle v_r,\phi_r(x_i^r)\rangle + b_r\right) \geq p_i^r - \xi_i^r, \\
& \xi_i^r \geq 0;\ i=1,\ldots,m_r,\ r=1,\ldots,T.
\end{aligned} \tag{4.7}$$

Here, we are using the unified formulation of TODO: Explicar formulación unificada.
 The prediction model is then

$$h_r(\cdot) = \langle w,\phi(\cdot)\rangle + \langle v_r,\phi_r(\cdot)\rangle + b_r$$

for regression and

$$h_r(\cdot) = \operatorname{sign}\left(\langle w,\phi(\cdot)\rangle + \langle v_r,\phi_r(\cdot)\rangle + b_r\right)$$

for classification. Observe again that the transformation ϕ is used for the common part, while the transformation ϕ_r is task-specific.

In (4.7) there are two kind of hyperparameters: C and μ, which, in combination, balance the different parts of the objective function. Hyperparameter C plays the same role than in the standard L1-SVM: it balances the tradeoff between the loss incurred by the model, represented by the slack variables ξ_i^r and the complexity of the models, represented by the norms $\|w\|$ and $\|v_r\|$. Large values of C highly penalize the loss, so the resulting models are more complex because they have to adapt to the training sample distribution, but these models tend to overfit. Small values of C penalize more the norms of w and v_r, hence the resulting models are simpler but not so dependent on the training sample.

The hyperparameter μ, in combination with C, balances how common or specific the models are. Large values of μ, penalize the common part, resulting in more specific models; while small values of μ, alongside large values of C, result in a vanishing regularization of the specific parts which leads to common models. We can distinguish the following cases:

- Reduction to an ITL approach:

$$\mu \to \infty \implies h_r(\cdot) = \langle v_r,\phi_r(\cdot)\rangle + b_r.$$

That is, the models are learned independently because the common part vanishes.

- Reduction to a CTL approach (with task-specific biases):

$$C \to 0, \mu \to 0 \implies h_r(\cdot) = \langle w, \phi(\cdot) \rangle + b_r.$$

That is, the model is common for all tasks because the specific parts disappear.

- Pure MTL approach:

$$0 < \mu < \infty \implies h_r(\cdot) = (\langle w, \phi(\cdot) \rangle) + (\langle v_r, \phi_r(\cdot) \rangle) + b_r.$$

There is a range of μ, which depends on each specific problem, in which the models combine a common and task-specific parts.

Observe that (4.7) is a convex problem. As in the standard case of the L1-SVM, the corresponding dual problem is solved. To obtain the dual problem, it is necessary to express the Lagrangian of problem (4.7),

$$
\begin{aligned}
&\mathcal{L}(w, \boldsymbol{v}, \boldsymbol{b}, \boldsymbol{\xi}, \boldsymbol{\alpha}, \boldsymbol{\beta}) \\
&= C \sum_{r=1}^{T} \sum_{i=1}^{m_r} \xi_i^r + \frac{1}{2} \sum_{r=1}^{T} \|v_r\|^2 + \frac{\mu}{2} \|w\|^2 \\
&\quad - \sum_{r=1}^{T} \sum_{i=1}^{m_r} \alpha_i^r \left\{ y_i^r \left[\langle w, \phi(x_i^r) \rangle + \langle v_r, \phi_r(x_i^r) \rangle + b_r \right] - p_i^r + \xi_i^r \right\} \\
&\quad - \sum_{r=1}^{T} \sum_{i=1}^{m_r} \beta_i^r \xi_i^r,
\end{aligned}
\tag{4.8}
$$

where $\alpha_i^r, \beta_i^r \geq 0$ are the Lagrange multipliers. Here $\boldsymbol{\xi}$ represents the vector

$$(\xi_1^1, \ldots, \xi_{m_1}^1, \ldots, \xi_1^T, \ldots, \xi_{m_T}^T)^\mathsf{T}$$

and we define $\boldsymbol{\alpha}$ and $\boldsymbol{\beta}$ analogously. For compactness we also use

$$\boldsymbol{v} = (v_1^\mathsf{T}, \ldots, v_T^\mathsf{T})^\mathsf{T}, \ \boldsymbol{b} = (b_1, \ldots, b_T).$$

Recall that the dual objective function is defined as

$$
\begin{aligned}
\Theta(\boldsymbol{\alpha}, \boldsymbol{\beta}) &= \min_{w, \boldsymbol{v}, b, b_1, \ldots, b_T, \boldsymbol{\xi}} \mathcal{L}(w, \boldsymbol{v}, \boldsymbol{b}, \boldsymbol{\xi}, \boldsymbol{\alpha}) \\
&= \mathcal{L}(w^*, \boldsymbol{v}^*, \boldsymbol{b}^*, \boldsymbol{\xi}^*, \boldsymbol{\alpha}, \boldsymbol{\beta})
\end{aligned}
$$

Since $\mathcal{L}$ is convex with respect to the primal variables, it is just necessary to compute the corresponding gradients

$$\nabla_w \mathcal{L} = 0 \implies \mu w^* - \sum_{r=1}^{T} \sum_{i=1}^{m_r} \alpha_i^r \left\{ y_i^r \phi(x_i^r) \right\} = 0, \tag{4.9}$$

$$\nabla_{v_r} \mathcal{L} = 0 \implies v_r{}^* - \sum_{i=1}^{m_r} \alpha_i^r \left\{ y_i^r \phi_r(x_i^r) \right\} = 0, \tag{4.10}$$

$$\nabla_{b_r} \mathcal{L} = 0 \implies \sum_{i=1}^{m_r} \alpha_i^r y_i^r = 0, \tag{4.11}$$

$$\nabla_{\xi_i^r} \mathcal{L} = 0 \implies C - \alpha_i^r - \beta_i^r = 0. \tag{4.12}$$

Using these results and substituting back in the Lagrangian we obtain

$$\mathcal{L}(w^*, \boldsymbol{v}^*, b_1{}^*, \dots, b_T{}^*, \boldsymbol{\xi}^*, \boldsymbol{\alpha}, \boldsymbol{\beta}) =$$

$$= \frac{1}{2} \sum_{r=1}^{T} \left\| \sum_{i=1}^{m_r} \alpha_i^r \left\{ y_i^r \phi_r(x_i^r) \right\} \right\|^2 + \frac{\mu}{2} \left\| \frac{1}{\mu} \sum_{r=1}^{T} \sum_{i=1}^{m_r} \alpha_i^r \left\{ y_i^r \phi(x_i^r) \right\} \right\|^2$$

$$- \sum_{r=1}^{T} \sum_{i=1}^{m_r} \alpha_i^r \left\{ y_i^r \left[\left(\frac{1}{\mu} \sum_{s=1}^{T} \sum_{j=1}^{m_s} \alpha_j^s \left\{ y_j^s \phi(x_j^s) \right\} \right) \cdot \phi(x_i^r) \right] \right\}$$

$$- \sum_{r=1}^{T} \sum_{i=1}^{m_r} \alpha_i^r \left\{ y_i^r \left[\left(\sum_{j=1}^{m_r} \alpha_j^r \left\{ y_j^r \phi_r(x_j^r) \right\} \right) \cdot \phi_r(x_i^r) \right] \right\}$$

$$- \sum_{r=1}^{T} \sum_{i=1}^{m_r} \alpha_i^r \left\{ -p_i^r \right\}$$

$$= \frac{1}{2} \sum_{r=1}^{T} \left\langle \sum_{i=1}^{m_r} \alpha_i^r \left\{ y_i^r \phi_r(x_i^r) \right\}, \sum_{i=1}^{m_r} \alpha_i^r \left\{ y_i^r \phi_r(x_i^r) \right\} \right\rangle$$

$$+ \frac{\mu}{2} \left\langle \frac{1}{\mu} \sum_{r=1}^{T} \sum_{i=1}^{m_r} \alpha_i^r \left\{ y_i^r \phi(x_i^r) \right\}, \frac{1}{\mu} \sum_{r=1}^{T} \sum_{i=1}^{m_r} \alpha_i^r \left\{ y_i^r \phi(x_i^r) \right\} \right\rangle$$

$$- \frac{1}{\mu} \left\langle \sum_{r=1}^{T} \sum_{i=1}^{m_r} \alpha_i^r \left\{ y_i^r \phi(x_i^r) \right\}, \sum_{r=1}^{T} \sum_{i=1}^{m_r} \alpha_i^r \left\{ y_i^r \phi(x_i^r) \right\} \right\rangle$$

$$- \sum_{r=1}^{T} \left\langle \sum_{i=1}^{m_r} \alpha_i^r \left\{ y_i^r \phi_r(x_i^r) \right\}, \sum_{i=1}^{m_r} \alpha_i^r \left\{ y_i^r \phi_r(x_i^r) \right\} \right\rangle$$

$$- \sum_{r=1}^{T} \sum_{i=1}^{m_r} \alpha_i^r \left\{ -p_i^r \right\}$$

$$= -\frac{1}{2\mu} \left\langle \sum_{r=1}^{T} \sum_{i=1}^{m_r} \alpha_i^r \left\{ y_i^r \phi(x_i^r) \right\}, \sum_{r=1}^{T} \sum_{i=1}^{m_r} \alpha_i^r \left\{ y_i^r \phi(x_i^r) \right\} \right\rangle$$

$$- \frac{1}{2} \sum_{r=1}^{T} \left\langle \sum_{i=1}^{m_r} \alpha_i^r \left\{ y_i^r \phi_r(x_i^r) \right\}, \sum_{i=1}^{m_r} \alpha_i^r \left\{ y_i^r \phi_r(x_i^r) \right\} \right\rangle$$

$$+ \sum_{r=1}^{T} \sum_{i=1}^{m_r} \alpha_i^r p_i^r.$$

Observe that $\boldsymbol{\beta}$ has disappeared from the Lagrangian. Then, the dual problem can be defined as $\min_{\boldsymbol{\alpha}} \Theta(\boldsymbol{\alpha})$ where

$$\Theta(\boldsymbol{\alpha}) = -\mathcal{L}(w^*, \boldsymbol{v}^*, b^*, \boldsymbol{d}^*, \boldsymbol{\xi}^*, \boldsymbol{\alpha}, \boldsymbol{\beta}),$$

and $\boldsymbol{\alpha}$ must fulfill the KKT conditions, that is TODO: Explicar KKT . The condition (4.12), using that $\alpha_i^r, \beta_i^r \geq 0$, implies $0 \leq \alpha_i^r \leq C$. Taking into account these KKT

conditions, the dual problem can be expressed as

$$
\min_{\alpha} \quad \Theta(\alpha) = \frac{1}{2\mu} \left\langle \sum_{r=1}^{T} \sum_{i=1}^{m_r} \alpha_i^r \left\{ y_i^r \phi(x_i^r) \right\}, \sum_{r=1}^{T} \sum_{i=1}^{m_r} \alpha_i^r \left\{ y_i^r \phi(x_i^r) \right\} \right\rangle
$$
$$
+ \frac{1}{2} \sum_{r=1}^{T} \left\langle \sum_{i=1}^{m_r} \alpha_i^r \left\{ y_i^r \phi_r(x_i^r) \right\}, \sum_{i=1}^{m_r} \alpha_i^r \left\{ y_i^r \phi_r(x_i^r) \right\} \right\rangle - \sum_{r=1}^{T} \sum_{i=1}^{m_r} \alpha_i^r p_i^r, \quad (4.13)
$$
$$
\text{s.t.} \quad 0 \leq \alpha_i^r \leq C; \ i = 1, \ldots, m_r; \ r = 1, \ldots, T
$$
$$
\sum_{i=1}^{m_r} \alpha_i^r y_i^r = 0; \ r = 1, \ldots, T,
$$

where there are task-specific equality constraints that have its origin in Equation (4.11). Using the kernel trick, we can write the dual problem with a matrix formulation

$$
\min_{\alpha} \quad \Theta(\alpha) = \frac{1}{2} \boldsymbol{\alpha}^{\mathsf{T}} \left(\frac{1}{\mu} Q + K \right) \boldsymbol{\alpha} - \boldsymbol{p}\boldsymbol{\alpha}
$$
$$
\text{s.t.} \quad 0 \leq \alpha_i^r \leq C; \ i = 1, \ldots, m_r; \ r = 1, \ldots, T, \quad (4.14)
$$
$$
\sum_{i=1}^{m_r} \alpha_i^r y_i^r = 0; \ r = 1, \ldots, T.
$$

Here Q and K are the common and specific kernel matrices, respectively. The matrix Q is generated using the common kernel, defined as

$$
k(x_i^r, x_j^s) = \left\langle \phi(x_i^r), \phi(x_j^s) \right\rangle.
$$

and K is the block-diagonal matrix built using the kernel

$$
k_r(x_i^r, x_j^s) = \delta_{rs} \left\langle \phi_r(x_i^r), \phi_s(x_j^s) \right\rangle,
$$

that is,

$$
Q = \begin{pmatrix} \underbrace{Q_{1,1}}_{m_1 \times m_1} & \underbrace{Q_{1,2}}_{m_1 \times m_2} & \underbrace{Q_{1,3}}_{m_1 \times m_3} & \cdots & \underbrace{Q_{1,T}}_{m_1 \times m_T} \\ \underbrace{Q_{2,1}}_{m_2 \times m_1} & \underbrace{Q_{2,2}}_{m_2 \times m_2} & \underbrace{Q_{2,1}}_{m_2 \times m_1} & \cdots & \underbrace{Q_{2,T}}_{m_2 \times m_T} \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ \underbrace{Q_{T,1}}_{m_T \times m_1} & \underbrace{Q_{T,2}}_{m_T \times m_2} & \underbrace{Q_{T,3}}_{m_T \times m_3} & \cdots & \underbrace{Q_{T,T}}_{m_T \times m_T} \end{pmatrix},
$$

where each block $Q_{r,s}$ is defined as

$$
Q_{r,s} = \begin{pmatrix} y_1^r y_1^s k(x_1^r, x_1^s) & y_1^r y_2^s k(x_1^r, x_2^s) & \cdots & y_1^r y_{m_s}^s k(x_1^r, x_{m_s}^s) \\ y_2^r y_1^s k(x_2^r, x_1^s) & y_2^r y_2^s k(x_2^r, x_2^s) & \cdots & y_2^r y_{m_s}^s k(x_2^r, x_{m_s}^s) \\ \vdots & \vdots & \ddots & \vdots \\ y_{m_r}^r y_1^s k(x_{m_r}^r, x_1^s) & y_{m_r}^r y_2^s k(x_{m_r}^r, x_2^s) & \cdots & y_{m_r}^r y_{m_s}^s k(x_{m_r}^r, x_{m_s}^s) \end{pmatrix}; \quad (4.15)
$$

and

$$K = \begin{pmatrix} \underbrace{K_{1,1}}_{m_1 \times m_1} & \underbrace{0}_{m_1 \times m_2} & \underbrace{0}_{m_1 \times m_3} & \cdots & \underbrace{0}_{m_1 \times m_T} \\ \underbrace{0}_{m_2 \times m_1} & \underbrace{K_{2,2}}_{m_2 \times m_2} & \underbrace{0}_{m_2 \times m_1} & \cdots & \underbrace{0}_{m_2 \times m_T} \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ \underbrace{0}_{m_T \times m_1} & \underbrace{0}_{m_T \times m_2} & \underbrace{0}_{m_T \times m_3} & \cdots & \underbrace{K_{T,T}}_{m_T \times T} \end{pmatrix},$$

where each task block $K_{r,r}$ is defined as

$$K_{r,r} = \begin{pmatrix} y_1^r y_1^r k_r(x_1^r, x_1^r) & y_1^r y_2^r k_r(x_1^r, x_2^r) & \cdots & y_1^r y_{m_s}^r k_r(x_1^r, x_{m_r}^r) \\ y_2^r y_1^r k_r(x_2^r, x_1^r) & y_2^r y_2^r k_r(x_2^r, x_2^r) & \cdots & y_2^r y_{m_s}^r k_r(x_2^r, x_{m_r}^r) \\ \vdots & \vdots & \ddots & \vdots \\ y_{m_r}^r y_1^r k_r(x_{m_r}^r, x_1^r) & y_{m_r}^r y_2^r k_r(x_{m_r}^r, x_2^r) & \cdots & y_{m_r}^r y_{m_s}^r k_r(x_{m_r}^r, x_{m_r}^r) \end{pmatrix}. \quad (4.16)$$

Combined, we have a multi-task kernel matrix $\widehat{Q} = (1/\mu)Q + K$, whose corresponding multi-task kernel function can be expressed as

$$\widehat{k}(x_i^r, x_j^s) = \frac{1}{\mu} k(x_i^r, x_j^s) + \delta_{rs} k_r(x_i^r, x_j^s).$$

Here we are using two different kernels: a common one $k(x, \tilde{x})$ and a task-specific one $k_r(x, \tilde{x})$, each being the reproducing kernels for the common and task-specific transformations, respectively. That is, $k(x, \tilde{x}) = \langle \phi(x), \phi(\tilde{x}) \rangle$ and $k_r(x, \tilde{x}) = \langle \phi_r(x), \phi_r(\tilde{x}) \rangle$.

The dual problem (4.14) is very similar to the standard one TODO: poner referencia al estándar but we have two major differences: the use of the multi-task kernel matrix $\widehat{Q}$ and the multiple equality constraints. These constraints, which appear in (4.11), are consequence of the specific biases used in the primal problem (4.7). In Cai and Cherkassky (2012) the authors develop a Generalized SMO algorithm to account for these multiple equality constraints.

Analyzing the hyperparameters influence in the dual problem, note that C is an upper bound for the dual coefficients, as in the standard case, and the hyperparameter of interest for this MTL formulation, which is μ, scales the common matrix Q. As with the primal formulation, we can define three different cases:

- Reduction to an ITL approach:

$$\mu \to \infty \implies h_r(\cdot) = \sum_{i=1}^{m_r} \alpha_i^r y_i^r \{ k_r(x_i^r, \cdot) \} + b_r.$$

Thus, the matrix is block-diagonal and optimizing the dual problem is equivalent to optimizing a specific dual problem for each task.

- Reduction to a CTL approach (with task-specific biases):

$$C \to 0, \mu \to 0 \implies h_r(\cdot) = \sum_{r=1}^{T} \sum_{i=1}^{m_r} \alpha_i^r y_i^r \{ k(x_i^r, \cdot) \} + b_r.$$

This is the standard dual objective function for CTL.

- Pure MTL approach:

$$0 < \mu < \infty \implies h_r(\cdot) = \sum_{r=1}^{T} \sum_{i=1}^{m_r} \alpha_i^r y_i^r \left\{ \frac{1}{\mu} k(x_i^r, \cdot) + k_r(x_i^r, \cdot) \right\} + b_r.$$

Here the kernel matrix combines the common and specific matrices.

Convex MTL L1-SVM.
We propose a convex formulation for the MTL L1-SVM, which we have presented in Ruiz et al. (2019). This formulation offers a better interpretability because the influence of the common and task-specific parts is more clear. The primal problem is

$$
\begin{aligned}
\underset{w,\boldsymbol{v},b,\boldsymbol{d},\boldsymbol{\xi}}{\arg\min} \quad & J(w,\boldsymbol{v},b,\boldsymbol{d},\boldsymbol{\xi}) = C \sum_{r=1}^{T} \sum_{i=1}^{m_r} \xi_i^r + \frac{1}{2} \sum_{r=1}^{T} \|v_r\|^2 + \frac{1}{2} \|w\|^2 \\
\text{s.t.} \quad & y_i^r \left(\lambda_r \left\{ \langle w, \phi(x_i^r) \rangle + b \right\} + (1 - \lambda_r) \left\{ \langle v_r, \phi_r(x_i^r) \rangle + d_r \right\} \right) \geq p_i^r - \xi_i^r, \\
& \xi_i^r \geq 0; \ i = 1, \dots, m_r, \ r = 1, \dots, T.
\end{aligned}
\tag{4.17}
$$

Here, the former hyperparameter μ used in the regularization is replaced by the hyperparameters λ_r, which are used in the model definition. The prediction model is then

$$h_r(\cdot) = \lambda_r \left\{ \langle w, \phi(\cdot) \rangle + b \right\} + (1 - \lambda_r) \left\{ \langle v_r, \phi_r(\cdot) \rangle + d_r \right\}$$

for regression and

$$h_r(\cdot) = \text{sign} \left(\lambda_r \left\{ \langle w, \phi(\cdot) \rangle + b \right\} + (1 - \lambda_r) \left\{ \langle v_r, \phi_r(\cdot) \rangle + d_r \right\} \right)$$

for classification.

With this convex formulation, the roles of hyperparameters C and λ_r are independent. Hyperparameter C regulates the trade-off between the loss and the margin size of each task-specialized model h_r, while λ_r indicates how specific or common these models are in the range of $[0,1]$. With $\lambda_r = 0$ we have independent models for each task and for $\lambda_r = 1$ we have a common model for all tasks. In (4.17), depending on the value of hyperparameters $C, \lambda_1, \dots, \lambda_T$, we can highlight the following situations:

- Reduction to an ITL approach:

$$\lambda_r = 0; \ r = 1, \dots, T \implies h_r(\cdot) = \langle v_r, \phi_r(\cdot) \rangle + d_r.$$

- Reduction to a CTL approach:

$$\lambda_r = 1; \ r = 1, \dots, T \implies h_r(\cdot) = \langle w, \phi(\cdot) \rangle + b.$$

- Pure MTL approach:

$$0 < \lambda_r < 1; \ r = 1, \dots, T \implies h_r(\cdot) = \lambda_r(\langle w, \phi(\cdot) \rangle + b) + (1 - \lambda_r)(\langle v_r, \phi_r(\cdot) \rangle + d_r).$$

Observe that now the cases are not unbounded but have attainable values, 0 for ITL and 1 for CTL, while all the values in the open $(0,1)$ yield pure MTL models. Also, the parameter C no longer interferes with these cases and only the λ_r calibrate the specifity

of the models. The Lagrangian of problem (4.17) is

$$
\begin{aligned}
&\mathcal{L}(w, \boldsymbol{v}, b, \boldsymbol{d}, \boldsymbol{\xi}, \boldsymbol{\alpha}, \boldsymbol{\beta}) \\
&= C \sum_{r=1}^{T} \sum_{i=1}^{m_r} \xi_i^r + \frac{1}{2} \sum_{r=1}^{T} \|v_r\|^2 + \frac{1}{2} \|w\|^2 \\
&\quad - \sum_{r=1}^{T} \sum_{i=1}^{m_r} \alpha_i^r \left\{ y_i^r \left(\lambda_r \left\{ \langle w, \phi(x_i^r) \rangle + b \right\} + (1 - \lambda_r) \left\{ \langle v_r, \phi_r(x_i^r) \rangle + d_r \right\} \right) - p_i^r + \xi_i^r \right\} \\
&\quad - \sum_{r=1}^{T} \sum_{i=1}^{m_r} \beta_i^r \xi_i^r,
\end{aligned}
$$

(4.18)

where $\alpha_i^r, \beta_i^r \geq 0$ are the Lagrange multipliers. Again, $\boldsymbol{\xi}$ represents the vector

$$
(\xi_1^1, \ldots, \xi_{m_1}^1, \ldots, \xi_1^T, \ldots, \xi_{m_T}^T)^\intercal
$$

and analogously for $\boldsymbol{\alpha}$ and $\boldsymbol{\beta}$. We also use

$$
\boldsymbol{v} = (v_1^\intercal, \ldots, v_T^\intercal)^\intercal, \quad \boldsymbol{d} = (d_1, \ldots, d_T).
$$

The dual objective function is defined as

$$
\begin{aligned}
\Theta(\boldsymbol{\alpha}, \boldsymbol{\beta}) &= \min_{w, \boldsymbol{v}, b, \boldsymbol{d}, \boldsymbol{\xi}} \mathcal{L}(w, \boldsymbol{v}, b, \boldsymbol{d}, \boldsymbol{\xi}, \boldsymbol{\alpha}, \boldsymbol{\beta}) \\
&= \mathcal{L}(w^*, \boldsymbol{v}^*, b^*, \boldsymbol{d}^*, \boldsymbol{\xi}^*, \boldsymbol{\alpha}, \boldsymbol{\beta})
\end{aligned}
$$

The gradients with respect to the primal variables are

$$
\nabla_w \mathcal{L} = 0 \implies w^* - \sum_{r=1}^{T} \lambda_r \sum_{i=1}^{m_r} \alpha_i^r \left\{ y_i^r \phi(x_i^r) \right\} = 0,
\tag{4.19}
$$

$$
\nabla_{v_r} \mathcal{L} = 0 \implies v_r^* - (1 - \lambda_r) \sum_{i=1}^{m_r} \alpha_i^r \left\{ y_i^r \phi_r(x_i^r) \right\} = 0,
\tag{4.20}
$$

$$
\nabla_b \mathcal{L} = 0 \implies \sum_{r=1}^{T} \sum_{i=1}^{m_r} \alpha_i^r y_i^r = 0,
\tag{4.21}
$$

$$
\nabla_{d_r} \mathcal{L} = 0 \implies \sum_{i=1}^{m_r} \alpha_i^r y_i^r = 0,
\tag{4.22}
$$

$$
\nabla_{\xi_i^r} \mathcal{L} = 0 \implies C - \alpha_i^r - \beta_i^r = 0.
\tag{4.23}
$$

Using these results and substituting back in the Lagrangian we obtain

$$
\mathcal{L}(w^*, \boldsymbol{v}^*, b^*, \boldsymbol{d}^*, \boldsymbol{\xi}^*, \boldsymbol{\alpha}, \boldsymbol{\beta})
$$

$$
= \frac{1}{2} \sum_{r=1}^{T} \left\| (1-\lambda_r) \sum_{i=1}^{m_r} \alpha_i^r \{y_i^r \phi_r(x_i^r)\} \right\|^2 + \frac{1}{2} \left\| \sum_{r=1}^{T} \lambda_r \sum_{i=1}^{m_r} \alpha_i^r \{y_i^r \phi(x_i^r)\} \right\|^2
$$

$$
- \sum_{r=1}^{T} \lambda_r \sum_{i=1}^{m_r} \alpha_i^r \left\{ y_i^r \left[\left(\sum_{s=1}^{T} \lambda_r \sum_{j=1}^{m_s} \alpha_j^s \{y_j^s \phi(x_j^s)\} \right) \cdot \phi(x_i^r) \right] \right\}
$$

$$
- \sum_{r=1}^{T} (1-\lambda_r) \sum_{i=1}^{m_r} \alpha_i^r \left\{ y_i^r \left[\left((1-\lambda_r) \sum_{j=1}^{m_r} \alpha_j^r \{y_j^r \phi_r(x_j^r)\} \right) \cdot \phi_r(x_i^r) \right] \right\}
$$

$$
- \sum_{r=1}^{T} \sum_{i=1}^{m_r} \alpha_i^r \{-p_i^r\}
$$

$$
= \frac{1}{2} \sum_{r=1}^{T} \left\langle (1-\lambda_r) \sum_{i=1}^{m_r} \alpha_i^r \{y_i^r \phi_r(x_i^r)\}, (1-\lambda_r) \sum_{i=1}^{m_r} \alpha_i^r \{y_i^r \phi_r(x_i^r)\} \right\rangle
$$

$$
+ \frac{1}{2} \left\langle \sum_{r=1}^{T} \lambda_r \sum_{i=1}^{m_r} \alpha_i^r \{y_i^r \phi(x_i^r)\}, \sum_{r=1}^{T} \lambda_r \sum_{i=1}^{m_r} \alpha_i^r \{y_i^r \phi(x_i^r)\} \right\rangle
$$

$$
- \left\langle \sum_{r=1}^{T} \lambda_r \sum_{i=1}^{m_r} \alpha_i^r \{y_i^r \phi(x_i^r)\}, \sum_{r=1}^{T} \lambda_r \sum_{i=1}^{m_r} \alpha_i^r \{y_i^r \phi(x_i^r)\} \right\rangle
$$

$$
- \sum_{r=1}^{T} \left\langle (1-\lambda_r) \sum_{i=1}^{m_r} \alpha_i^r \{y_i^r \phi_r(x_i^r)\}, (1-\lambda_r) \sum_{i=1}^{m_r} \alpha_i^r \{y_i^r \phi_r(x_i^r)\} \right\rangle
$$

$$
- \sum_{r=1}^{T} \sum_{i=1}^{m_r} \alpha_i^r \{-p_i^r\}
$$

$$
= -\frac{1}{2} \left\langle \sum_{r=1}^{T} \lambda_r \sum_{i=1}^{m_r} \alpha_i^r \{y_i^r \phi(x_i^r)\}, \sum_{r=1}^{T} \lambda_r \sum_{i=1}^{m_r} \alpha_i^r \{y_i^r \phi(x_i^r)\} \right\rangle
$$

$$
- \frac{1}{2} \sum_{r=1}^{T} (1-\lambda_r) \left\langle \sum_{i=1}^{m_r} \alpha_i^r \{y_i^r \phi_r(x_i^r)\}, (1-\lambda_r) \sum_{i=1}^{m_r} \alpha_i^r \{y_i^r \phi_r(x_i^r)\} \right\rangle
$$

$$
+ \sum_{r=1}^{T} \sum_{i=1}^{m_r} \alpha_i^r p_i^r.
$$

As with the additive formulation, the dual problem can then be defined as

$$
\begin{aligned}
\min_{\alpha} \quad \Theta(\alpha) = {}& \frac{1}{2} \left\langle \sum_{r=1}^{T} \lambda_r \sum_{i=1}^{m_r} \alpha_i^r \left\{ y_i^r \phi(x_i^r) \right\}, \sum_{r=1}^{T} \lambda_r \sum_{i=1}^{m_r} \alpha_i^r \left\{ y_i^r \phi(x_i^r) \right\} \right\rangle \\
& + \frac{1}{2} \sum_{r=1}^{T} \left\langle (1 - \lambda_r) \sum_{i=1}^{m_r} \alpha_i^r \left\{ y_i^r \phi_r(x_i^r) \right\}, (1 - \lambda_r) \sum_{i=1}^{m_r} \alpha_i^r \left\{ y_i^r \phi_r(x_i^r) \right\} \right\rangle \\
& - \sum_{r=1}^{T} \sum_{i=1}^{m_r} \alpha_i^r p_i^r
\end{aligned}
\tag{4.24}
$$

$$
\begin{aligned}
\text{s.t.} \quad & 0 \le \alpha_i^r \le C; \ i = 1, \ldots, m_r; \ r = 1, \ldots, T, \\
& \sum_{i=1}^{m_r} \alpha_i^r y_i^r = 0; \ r = 1, \ldots, T.
\end{aligned}
$$

And the matrix formulation using the kernel matrices is

$$
\begin{aligned}
\min_{\alpha} \quad & \Theta(\alpha) = \frac{1}{2} \boldsymbol{\alpha}^{\mathsf{T}} \left(\Lambda Q \Lambda + (I_n - \Lambda) K (I_n - \Lambda) \right) \boldsymbol{\alpha} - \boldsymbol{p}\boldsymbol{\alpha} \\
\text{s.t.} \quad & 0 \le \alpha_i^r \le C; \ i = 1, \ldots, m_r; \ r = 1, \ldots, T, \\
& \sum_{i=1}^{m_r} \alpha_i^r y_i^r = 0; \ r = 1, \ldots, T,
\end{aligned}
\tag{4.25}
$$

where

$$
\Lambda = \operatorname{diag}(\overbrace{\lambda_1, \ldots, \lambda_1}^{m_1}, \ldots, \overbrace{\lambda_T, \ldots, \lambda_T}^{m_T})
\tag{4.26}
$$

and I_n is the $n \times n$ identity matrix; here Q and K are the common and specific kernel matrices, respectively. Combined, we have a multi-task kernel matrix

$$
\widehat{Q} = \Lambda Q \Lambda + (I_n - \Lambda) K (I_n - \Lambda),
\tag{4.27}
$$

whose corresponding multi-task kernel function can be expressed as

$$
\widehat{k}(x_i^r, x_j^s) = \lambda_r \lambda_s k(x_i^r, x_j^s) + \delta_{rs}(1 - \lambda_r)(1 - \lambda_s) k_r(x_i^r, x_j^s),
\tag{4.28}
$$

where again we are using the common kernel $k(x, \tilde{x}) = \langle \phi(x), \phi(\tilde{x}) \rangle$ and the task-specific kernels $k_r(x, \tilde{x}) = \langle \phi_r(x), \phi_r(\tilde{x}) \rangle$ for $r = 1, \ldots, T$. Then, we can define the kernel matrix as

$$
\widehat{Q} = \begin{pmatrix}
y_1^1 y_1^1 \widehat{k}(x_1^1, x_1^1) & y_1^1 y_2^1 \widehat{k}(x_1^1, x_2^1) & \ldots & y_1^1 y_{m_T}^T \widehat{k}(x_1^1, x_{m_T}^T) \\
y_2^1 y_1^1 \widehat{k}(x_2^1, x_1^1) & y_2^1 y_2^1 \widehat{k}(x_2^1, x_2^1) & \ldots & y_2^1 y_{m_T}^T \widehat{k}(x_2^1, x_{m_T}^T) \\
\vdots & \vdots & \ddots & \vdots \\
y_{m_T}^T y_1^1 \widehat{k}(x_{m_T}^T, x_1^1) & y_{m_T}^T y_2^1 \widehat{k}(x_{m_T}^T, x_2^1) & \ldots & y_{m_T}^T y_{m_T}^T \widehat{k}(x_{m_T}^T, x_{m_T}^T)
\end{pmatrix}.
$$

This dual problem is very similar to the one shown in (4.14) where there are also T equality constraints, but the multi-task kernel matrix $\widehat{Q}$ is defined differently, dropping the μ hyperparameter and incorporating the λ_r ones. Studying the influence of λ_r hyperparameters in the dual problem we can describe the following cases:

- Reduction to an ITL approach:

$$\lambda_r = 0;\ r = 1, \ldots, T \implies h_r(\hat{x}) = \sum_{r=1}^{T} \sum_{i=1}^{m_r} \alpha_i^r y_i^r \left\{ k_r(x_i^r, \hat{x}) \right\} + d_r.$$

- Reduction to a CTL approach:

$$\lambda_r = 1;\ r = 1, \ldots, T \implies h_r(\hat{x}) = \sum_{r=1}^{T} \sum_{i=1}^{m_r} \alpha_i^r y_i^r \left\{ k(x_i^r, \hat{x}) \right\} + b.$$

- Pure MTL approach:

$$0 < \lambda_r < 1;\ r = 1, \ldots, T \implies$$
$$h_r(\hat{x}) = \sum_{r=1}^{T} \sum_{i=1}^{m_r} \alpha_i^r y_i^r \left\{ \lambda_r^2 k(x_i^r, \hat{x}) + (1 - \lambda_r)^2 k_r(x_i^r, \hat{x}) \right\} + \lambda_r b + (1 - \lambda_r) d_r.$$

The properties that are found in the primal formulation are also present in the dual one. All the values of λ_r in the open interval $(0, 1)$ correspond to pure MTL approaches while the extreme values $\lambda_r = 1$ and $\lambda_r = 0$ correspond to CTL and ITL approaches, respectively.

Equivalence Results.
Se hace subsección?
 Both the additive and convex MTL SVM approaches solve a similar problem, but there is a change in the formulation to get rid of a regularization hyperparameter μ in favor of those defining convex combination of models, hyperparameters λ_r. Both approaches offer similar properties: allowing the value of their hyperparameters to go from completely common to completely independent models, and passing through pure multi-task models. However, it is not obvious what is the relation between those two approaches. In Ruiz et al. (2019) we provide two propositions to show the equivalence between additive and convex MTL SVM formulations where a common λ is used.

Proposition 1. The additive MTL-SVM primal problem with parameters C_{add} and μ (and possibly ϵ), that is,

$$
\begin{aligned}
&\underset{w, \boldsymbol{v}, b, \boldsymbol{d}, \boldsymbol{\xi}}{\arg\min} && J(w, \boldsymbol{v}, b, \boldsymbol{d}, \boldsymbol{\xi}) = C_{\text{add}} \sum_{r=1}^{T} \sum_{i=1}^{m_r} \xi_i^r + \frac{1}{2} \sum_{r=1}^{T} \|v_r\|^2 + \frac{\mu}{2} \|w\|^2 \\
&\text{s.t.} && y_i^r \left(\langle w, \phi(x_i^r) \rangle + b + \langle v_r, \phi_r(x_i^r) \rangle + b_r \right) \geq p_i^r - \xi_i^r, \\
& && \xi_i^r \geq 0;\ i = 1, \ldots, m_r,\ r = 1, \ldots, T,
\end{aligned}
\tag{4.29}
$$

and the convex MTL-SVM primal problem with parameters C_{conv} and $\lambda_1 = \ldots = \lambda_T = \lambda$ (and possibly ϵ), that is,

$$
\begin{aligned}
&\underset{u, u_r, \xi}{\arg\min} && J(u, u_r, \xi) = C_{\text{conv}} \sum_{r=1}^{T} \sum_{i=1}^{m_r} \xi_i^r + \frac{1}{2} \sum_{r=1}^{T} \|u_r\|^2 + \frac{1}{2} \|u\|^2 \\
&\text{s.t.} && y_i^r \left(\lambda \left\{ \langle u, \phi(x_i^r) \rangle + \hat{b} \right\} + (1 - \lambda) \left\{ \langle u_r, \phi_r(x_i^r) \rangle + \hat{d}_r \right\} \right) \geq p_i^r - \xi_i^r, \\
& && \xi_i^r \geq 0;\ i = 1, \ldots, m_r,\ r = 1, \ldots, T,
\end{aligned}
\tag{4.30}
$$

with $\lambda \in (0, 1)$, are equivalent when $C_{\text{add}} = (1-\lambda)^2 C_{\text{conv}}$ and $\mu = (1-\lambda)^2/\lambda^2$.

Proof. Making the change of variables $w = \lambda u$, $v_r = (1-\lambda)u_r$ and $b_r = \lambda\hat{b} + (1-\lambda)\hat{b}_r$ in the convex primal problem (4.30), we can write it as

$$\underset{w,v_r,\xi}{\arg\min} \quad J(w, v_r, \xi) = C_{\text{conv}} \sum_{t=1}^{T} \sum_{i=1}^{m_r} \xi_i^r + \frac{1}{2(1-\lambda)^2} \sum_{t=1}^{T} \|v_r\|^2 + \frac{1}{2\lambda^2}\|w\|^2$$
$$\text{s.t.} \quad y_i^r(\langle w, \phi(x_i^r)\rangle + \langle v_r, \phi_r(x_i^r)\rangle + b_r) \geq p_i^r - \xi_i^r,$$
$$\xi_i^r \geq 0,$$
$$i = 1, \ldots, m_r, \ t = 1, \ldots, T.$$

Multiplying now the objective function by $(1-\lambda)^2$ we obtain the additive MTL-SVM primal problem (4.29) with $\mu = (1-\lambda)^2/\lambda^2$ and $C_{\text{add}} = (1-\lambda)^2 C_{\text{conv}}$. Conversely, we can start at the primal additive problem and make the inverse changes to arrive now to the primal convex problem. $\square$

The previous proposition shows the equivalence between the primal problems, but this result can also be obtained from the dual ones. Consider the dual problem of the convex formulation when $\lambda_1 = \ldots = \lambda_T = \lambda$; multiplying the objective function by $\frac{1}{(1-\lambda)^2} > 0$ we get

$$\min_{\boldsymbol{\alpha}} \quad \Theta(\boldsymbol{\alpha}) = \frac{1}{2}\boldsymbol{\alpha}^{\mathsf{T}}\left(\frac{\lambda^2}{(1-\lambda)^2}Q + \frac{(1-\lambda)^2}{(1-\lambda)^2}K\right)\boldsymbol{\alpha} - \frac{1}{(1-\lambda)^2}\boldsymbol{p}\boldsymbol{\alpha}$$
$$\text{s.t.} \quad 0 \leq \alpha_i^r \leq C_{\text{conv}}; \ i = 1, \ldots, m_r; \ r = 1, \ldots, T \qquad (4.31)$$
$$\sum_{i=1}^{m_r} \alpha_i^r y_i^r = 0; \ r = 1, \ldots, T.$$

Now consider the change of variables $\boldsymbol{\beta} = (1-\lambda)^2 \boldsymbol{\alpha}$; then we obtain the problem

$$\min_{\boldsymbol{\beta}} \quad \Theta(\boldsymbol{\beta}) = \frac{1}{2}\frac{1}{(1-\lambda)^2}\boldsymbol{\beta}^{\mathsf{T}}\left(\frac{\lambda^2}{(1-\lambda)^2}Q + \frac{(1-\lambda)^2}{(1-\lambda)^2}K\right)\frac{1}{(1-\lambda)^2}\boldsymbol{\beta} - \frac{1}{(1-\lambda)^4}\boldsymbol{p}\boldsymbol{\beta}$$
$$\text{s.t.} \quad 0 \leq \beta_i^r \leq (1-\lambda)^2 C_{\text{conv}}; \ i = 1, \ldots, m_r; \ r = 1, \ldots, T,$$
$$\sum_{i=1}^{m_r} \alpha_i^r y_i^r = 0; \ r = 1, \ldots, T.$$

If we consider again $\mu = (1-\lambda)^2/\lambda^2$ and $C_{\text{add}} = (1-\lambda)^2 C_{\text{conv}}$ and multiply the objective function by $(1-\lambda)^4$ we get the equivalent problem

$$\min_{\boldsymbol{\beta}} \quad \Theta(\boldsymbol{\beta}) = \frac{1}{2}\boldsymbol{\beta}^{\mathsf{T}}\left(\frac{1}{\mu}Q + K\right)\boldsymbol{\beta} - \boldsymbol{p}\boldsymbol{\beta}$$
$$\text{s.t.} \quad 0 \leq \beta_i^r \leq C_{\text{add}}; \ i = 1, \ldots, m_r; \ r = 1, \ldots, T$$
$$\sum_{i=1}^{m_r} \alpha_i^r y_i^r = 0; \ r = 1, \ldots, T,$$

which is the dual problem of the additive formulation.

Finally, observe that the results in Proposition 1 are only valid for λ in the open set $(0, 1)$. When $\lambda = 0$, we obtain an ITL approach, where an independent model is learned

for each task. When $\lambda = 1$, the convex formulation is equivalent to a CTL approach, where a single, common model is trained using all tasks.

4.2.2 Extensions to L2 and LS-SVM

The L2-SVM (Burges, 1998) is a variant of the standard SVM where the hinge loss is replaced by the squared hinge loss in the case of a classification setting, and the epsilon-insensitive loss by the corresponding squared version in the case of regression problems. In both settings a margin where the errors are not penalized is kept, but the errors which are larger than such margin are penalized using its squared value.

The LS-SVM (Suykens and Vandewalle, 1999) is another variant of the standard SVM where the epsilon-insensitive loss is replaced by the squared loss in the case of regression problems, and for classification a regression is made for the negative and positive classes.

In this subsection, an extension of the convex MTL formulation for both L2 and LS-SVM is presented.

Convex MTL L2-SVM.
The primal problem for the convex MTL L2-SVM, presented in Ruiz et al. (2021b), is

$$\underset{w, \boldsymbol{v}, b, \boldsymbol{d}, \boldsymbol{\xi}}{\arg\min} \quad J(w, \boldsymbol{v}, b, \boldsymbol{d}, \boldsymbol{\xi}) = \frac{C}{2} \sum_{r=1}^{T} \sum_{i=1}^{m_r} \xi_i^{r\,2} + \frac{1}{2} \sum_{r=1}^{T} \|v_r\|^2 + \frac{1}{2} \|w\|^2 \tag{4.32}$$

$$\text{s.t.} \quad y_i^r \left(\lambda_r \left\{ \langle w, \phi(x_i^r) \rangle + b \right\} + (1 - \lambda_r) \left\{ \langle v_r, \phi_r(x_i^r) \rangle + d_r \right\} \right) \geq p_i^r - \xi_i^r.$$

The prediction model is again

$$h_r(\cdot) = \lambda_r \left\{ \langle w, \phi(\cdot) \rangle + b \right\} + (1 - \lambda_r) \left\{ \langle v_r, \phi_r(\cdot) \rangle + d_r \right\}$$

for regression and

$$h_r(\cdot) = \text{sign} \left(\lambda_r \left\{ \langle w, \phi(\cdot) \rangle + b \right\} + (1 - \lambda_r) \left\{ \langle v_r, \phi_r(\cdot) \rangle + d_r \right\} \right)$$

for classification.

As in the standard variant, hyperparameter C regulates the trade-off between the loss and the margin size of each task-specialized model h_r, while λ_r indicates how specific or common these models are in the range of $[0, 1]$. With $\lambda_1, \ldots, \lambda_T = 0$ we have independent models for each task and for $\lambda_1, \ldots, \lambda_T = 1$ we have a common model for all tasks.

To obtain the dual problem the Lagrangian of problem (4.32):

$$\mathcal{L}(w, \boldsymbol{v}, b, \boldsymbol{d}, \boldsymbol{\xi}, \boldsymbol{\alpha})$$

$$= \frac{C}{2} \sum_{r=1}^{T} \sum_{i=1}^{m_r} (\xi_i^r)^2 + \frac{1}{2} \sum_{r=1}^{T} \|v_r\|^2 + \frac{1}{2} \|w\|^2$$

$$- \sum_{r=1}^{T} \sum_{i=1}^{m_r} \alpha_i^r \left\{ y_i^r \left(\lambda_r \left\{ \langle w, \phi(x_i^r) \rangle + b \right\} + (1 - \lambda_r) \left\{ \langle v_r, \phi_r(x_i^r) \rangle + d_r \right\} \right) - p_i^r + \xi_i^r \right\},$$

$$\tag{4.33}$$

where $\alpha_i^r \geq 0$ are the Lagrange multipliers. Again, $\boldsymbol{\xi}$ represents the vector

$$(\xi_1^1, \ldots, \xi_{m_1}^1, \ldots, \xi_1^T, \ldots, \xi_{m_T}^T)^{\intercal}$$

and analogously for $\boldsymbol{\alpha}$. The dual objective function is defined as

$$\Theta(\boldsymbol{\alpha}) = \min_{w,\boldsymbol{v},b,\boldsymbol{d},\boldsymbol{\xi}} \mathcal{L}(w,\boldsymbol{v},b,\boldsymbol{d},\boldsymbol{\xi},\boldsymbol{\alpha})$$
$$= \mathcal{L}(w^*,\boldsymbol{v}^*,b^*,\boldsymbol{d}^*,\boldsymbol{\xi}^*,\boldsymbol{\alpha})$$

The gradients with respect to the primal variables are

$$\nabla_w \mathcal{L} = 0 \implies w^* - \sum_{r=1}^{T} \lambda_r \sum_{i=1}^{m_r} \alpha_i^r \{y_i^r \phi(x_i^r)\} = 0, \tag{4.34}$$

$$\nabla_{v_r} \mathcal{L} = 0 \implies v_r^* - (1-\lambda_r) \sum_{i=1}^{m_r} \alpha_i^r \{y_i^r \phi_r(x_i^r)\} = 0, \tag{4.35}$$

$$\nabla_b \mathcal{L} = 0 \implies \sum_{r=1}^{T} \sum_{i=1}^{m_r} \alpha_i^r y_i^r = 0, \tag{4.36}$$

$$\nabla_{d_r} \mathcal{L} = 0 \implies \sum_{i=1}^{m_r} \alpha_i^r y_i^r = 0, \tag{4.37}$$

$$\nabla_{\xi_i^r} \mathcal{L} = 0 \implies C\xi_i^r - \alpha_i^r = 0. \tag{4.38}$$

Using these results and substituting back in the Lagrangian we obtain

$$\mathcal{L}(w^*,\boldsymbol{v}^*,b^*,\boldsymbol{d}^*,\boldsymbol{\xi}^*,\boldsymbol{\alpha})$$
$$= \frac{C}{2} \frac{1}{C^2} \sum_{r=1}^{T} \sum_{i=1}^{m_r} (\alpha_i^r)^2$$
$$+ \frac{1}{2} \sum_{r=1}^{T} \left\| (1-\lambda_r) \sum_{i=1}^{m_r} \alpha_i^r - \{y_i^r \phi_r(x_i^r)\} \right\|^2 + \frac{1}{2} \left\| \sum_{r=1}^{T} \lambda_r \sum_{i=1}^{m_r} \alpha_i^r \{y_i^r \phi(x_i^r)\} \right\|^2$$
$$- \sum_{r=1}^{T} \lambda_r \sum_{i=1}^{m_r} \alpha_i^r \left\{ y_i^r \left[\left(\sum_{s=1}^{T} \lambda_r \sum_{j=1}^{m_s} \alpha_j^s \{y_j^s \phi(x_j^s)\} \right) \cdot \phi(x_i^r) \right] \right\}$$
$$- \sum_{r=1}^{T} (1-\lambda_r) \sum_{i=1}^{m_r} \alpha_i^r \left\{ y_i^r \left[\left((1-\lambda_r) \sum_{j=1}^{m_r} \alpha_j^r \{y_j^r \phi_r(x_j^r)\} \right) \cdot \phi_r(x_i^r) \right] \right\}$$
$$- \sum_{r=1}^{T} \sum_{i=1}^{m_r} \alpha_i^r \{-p_i^r\} - \frac{1}{C} \sum_{r=1}^{T} \sum_{i=1}^{m_r} (\alpha_i^r)^2,$$

which is equal to

$$
\frac{1}{2} \sum_{r=1}^{T} \left\langle (1-\lambda_r) \sum_{i=1}^{m_r} \alpha_i^r \left\{ y_i^r \phi_r(x_i^r) \right\}, (1-\lambda_r) \sum_{i=1}^{m_r} \alpha_i^r \left\{ y_i^r \phi_r(x_i^r) \right\} \right\rangle
$$

$$
+ \frac{1}{2} \left\langle \sum_{r=1}^{T} \lambda_r \sum_{i=1}^{m_r} \alpha_i^r \left\{ y_i^r \phi(x_i^r) \right\}, \sum_{r=1}^{T} \lambda_r \sum_{i=1}^{m_r} \alpha_i^r \left\{ y_i^r \phi(x_i^r) \right\} \right\rangle
$$

$$
- \left\langle \sum_{r=1}^{T} \lambda_r \sum_{i=1}^{m_r} \alpha_i^r \left\{ y_i^r \phi(x_i^r) \right\}, \sum_{r=1}^{T} \lambda_r \sum_{i=1}^{m_r} \alpha_i^r \left\{ y_i^r \phi(x_i^r) \right\} \right\rangle
$$

$$
- \sum_{r=1}^{T} \left\langle (1-\lambda_r) \sum_{i=1}^{m_r} \alpha_i^r \left\{ y_i^r \phi_r(x_i^r) \right\}, (1-\lambda_r) \sum_{i=1}^{m_r} \alpha_i^r \left\{ y_i^r \phi_r(x_i^r) \right\} \right\rangle
$$

$$
- \frac{1}{2C} \sum_{r=1}^{T} \sum_{i=1}^{m_r} (\alpha_i^r)^2 - \sum_{r=1}^{T} \sum_{i=1}^{m_r} \alpha_i^r \left\{ -p_i^r \right\}
$$

$$
= -\frac{1}{2} \left\langle \sum_{r=1}^{T} \lambda_r \sum_{i=1}^{m_r} \alpha_i^r \left\{ y_i^r \phi(x_i^r) \right\}, \sum_{r=1}^{T} \lambda_r \sum_{i=1}^{m_r} \alpha_i^r \left\{ y_i^r \phi(x_i^r) \right\} \right\rangle
$$

$$
- \frac{1}{2} \sum_{r=1}^{T} (1-\lambda_r) \left\langle \sum_{i=1}^{m_r} \alpha_i^r \left\{ y_i^r \phi_r(x_i^r) \right\}, (1-\lambda_r) \sum_{i=1}^{m_r} \alpha_i^r \left\{ y_i^r \phi_r(x_i^r) \right\} \right\rangle
$$

$$
- \frac{1}{2C} \sum_{r=1}^{T} \sum_{i=1}^{m_r} (\alpha_i^r)^2 + \sum_{r=1}^{T} \sum_{i=1}^{m_r} \alpha_i^r p_i^r.
$$

The corresponding dual problem can be then expressed as

$$
\min_{\boldsymbol{\alpha}} \quad \Theta(\boldsymbol{\alpha}) = \frac{1}{2} \frac{1}{C} \sum_{r=1}^{T} \sum_{i=1}^{m_r} (\alpha_i^r)^2
$$

$$
+ \frac{1}{2} \left\langle \sum_{r=1}^{T} \lambda_r \sum_{i=1}^{m_r} \alpha_i^r \left\{ y_i^r \phi(x_i^r) \right\}, \sum_{r=1}^{T} \lambda_r \sum_{i=1}^{m_r} \alpha_i^r \left\{ y_i^r \phi(x_i^r) \right\} \right\rangle
$$

$$
+ \frac{1}{2} \sum_{r=1}^{T} \left\langle (1-\lambda_r) \sum_{i=1}^{m_r} \alpha_i^r \left\{ y_i^r \phi_r(x_i^r) \right\}, (1-\lambda_r) \sum_{i=1}^{m_r} \alpha_i^r \left\{ y_i^r \phi_r(x_i^r) \right\} \right\rangle \tag{4.39}
$$

$$
- \sum_{r=1}^{T} \sum_{i=1}^{m_r} \alpha_i^r p_i^r
$$

$$
\text{s.t.} \qquad 0 \leq \alpha_i^r; \; i = 1, \ldots, m_r; \; r = 1, \ldots, T,
$$

$$
\sum_{i=1}^{m_r} \alpha_i^r y_i^r = 0; \; r = 1, \ldots, T.
$$

Using the kernel trick, we can write the dual problem using a matrix formulation in the following way:

$$
\begin{aligned}
\min_{\alpha} \quad & \Theta(\alpha) = \frac{1}{2}\boldsymbol{\alpha}^{\mathsf{T}} \left(\{\Lambda Q \Lambda + (I_n - \Lambda) K (I_n - \Lambda)\} + \frac{1}{C}I \right) \boldsymbol{\alpha} - \boldsymbol{p}\boldsymbol{\alpha} \\
\text{s.t.} \quad & 0 \le \alpha_i^r; \ i = 1, \ldots, m_r; \ \ r = 1, \ldots, T, \\
& \sum_{i=1}^{m_r} \alpha_i^r y_i^r = 0; \ r = 1, \ldots, T.
\end{aligned}
\tag{4.40}
$$

As in the convex MTL L1-SVM, we are using the matrix

$$
\Lambda = \text{diag}(\overbrace{\lambda_1, \ldots, \lambda_1}^{m_1}, \ldots, \overbrace{\lambda_T, \ldots, \lambda_T}^{m_T}),
\tag{4.41}
$$

and the $n \times n$ identity matrix I_n. Note that instead of having the box constraints for the dual coefficients α_i^r we add a diagonal term to the MTL kernel matrix $\widehat{Q}$, as defined in (4.27), so the matrix that is used is $\widehat{Q} + \frac{1}{C}I_n$.

The difference between the convex MTL based on the L1-SVM and this one, based on the L2-SVM, can be seen in the primal formulation, where the square of the errors is penalized, and, therefore, the variables ξ_i^r do not need to be non-negative. This is reflected in the dual problem, where there is no longer an upper bound for the dual coefficients, but a diagonal term is added to the kernel matrix, which acts as a soft constraint for the size of these dual coefficients.

Convex MTL LS-SVM.
The primal problem for the convex MTL LS-SVM, presented in Ruiz et al. (2021b), is

$$
\begin{aligned}
\underset{w, \boldsymbol{v}, b, \boldsymbol{d}, \xi}{\arg\min} \quad & J(w, \boldsymbol{v}, b, \boldsymbol{d}, \boldsymbol{\xi}) = \frac{C}{2} \sum_{r=1}^{T} \sum_{i=1}^{m_r} (\xi_i^r)^2 + \frac{1}{2} \sum_{r=1}^{T} \|v_r\|^2 + \frac{1}{2}\|w\|^2 \\
\text{s.t.} \quad & y_i^r \left(\lambda_r \{\langle w, \phi(x_i^r)\rangle + b\} + (1 - \lambda_r) \{\langle v_r, \phi_r(x_i^r)\rangle + d_r\} \right) = p_i^r - \xi_i^r.
\end{aligned}
\tag{4.42}
$$

Observe that it differs with the L1-SVM from (4.17) and L2-SVM from (4.32) The prediction model is again

$$
h_r(\cdot) = \lambda_r \{\langle w, \phi(\cdot)\rangle + b\} + (1 - \lambda_r) \{\langle v_r, \phi_r(\cdot)\rangle + d_r\}
$$

for regression and

$$
h_r(\cdot) = \text{sign} \left(\lambda_r \{\langle w, \phi(\cdot)\rangle + b\} + (1 - \lambda_r) \{\langle v_r, \phi_r(\cdot)\rangle + d_r\} \right)
$$

for classification.

As in the standard variant, the hyperparameter C regulates the trade-off between the loss and the margin size, while λ_r, in the range $[0, 1]$, indicates how specific or common the model is. With $\lambda_1, \ldots, \lambda_T = 0$, we have independent models for each task and for $\lambda_1, \ldots, \lambda_T = 1$, we have a common model for all tasks.

The Lagrangian of problem (4.42) is

$$
\begin{aligned}
&\mathcal{L}(w, \boldsymbol{v}, b, \boldsymbol{d}, \boldsymbol{\xi}, \boldsymbol{\alpha}) \\
&= \frac{C}{2} \sum_{r=1}^{T} \sum_{i=1}^{m_r} (\xi_i^r)^2 + \frac{1}{2} \sum_{r=1}^{T} \|v_r\|^2 + \frac{1}{2} \|w\|^2 \\
&\quad - \sum_{r=1}^{T} \sum_{i=1}^{m_r} \alpha_i^r \left\{ y_i^r \left(\lambda_r \left\{ \langle w, \phi(x_i^r) \rangle + b \right\} + (1 - \lambda_r) \left\{ \langle v_r, \phi_r(x_i^r) \rangle + d_r \right\} \right) - p_i^r + \xi_i^r \right\},
\end{aligned}
$$

(4.43)

where $\alpha_i^r \in \mathbb{R}$ are the Lagrange multipliers. Observe that, although the Lagrangian is identical to the one of the L2-SVM, the non-negativity condition is no longer required for the dual coefficients. Again, $\boldsymbol{\xi}$ represents the vector

$$
(\xi_1^1, \ldots, \xi_{m_1}^1, \ldots, \xi_1^T, \ldots, \xi_{m_T}^T)^\intercal,
$$

and analogously for $\boldsymbol{\alpha}$. The dual objective function is defined as

$$
\begin{aligned}
\min_{w, \boldsymbol{v}, b, \boldsymbol{d}, \boldsymbol{\xi}} \ & \mathcal{L}(w, \boldsymbol{v}, b, \boldsymbol{d}, \boldsymbol{\xi}, \boldsymbol{\alpha}) \\
&= \mathcal{L}(w^*, \boldsymbol{v}^*, b^*, \boldsymbol{d}^*, \boldsymbol{\xi}^*, \boldsymbol{\alpha})
\end{aligned}
$$

The gradients with respect to the primal variables are

$$
\nabla_w \mathcal{L} = 0 \implies w^* - \sum_{r=1}^{T} \lambda_r \sum_{i=1}^{m_r} \alpha_i^r \left\{ y_i^r \phi(x_i^r) \right\} = 0,
$$

(4.44)

$$
\nabla_{v_r} \mathcal{L} = 0 \implies v_r^* - (1 - \lambda_r) \sum_{i=1}^{m_r} \alpha_i^r \left\{ y_i^r \phi_r(x_i^r) \right\} = 0,
$$

(4.45)

$$
\nabla_b \mathcal{L} = 0 \implies \sum_{r=1}^{T} \sum_{i=1}^{m_r} \alpha_i^r y_i^r = 0,
$$

(4.46)

$$
\nabla_{d_r} \mathcal{L} = 0 \implies \sum_{i=1}^{m_r} \alpha_i^r y_i^r = 0,
$$

(4.47)

$$
\nabla_{\xi_i^r} \mathcal{L} = 0 \implies C \xi_i^r - \alpha_i^r = 0.
$$

(4.48)

Since all $\alpha_i^r \in \mathbb{R}$ are feasible, we can also write the corresponding KKT conditions as

$$
\frac{\partial \mathcal{L}}{\partial \alpha_i^r} = 0 \implies y_i^r \left(\lambda_r \left\{ \langle w, \phi(x_i^r) \rangle + b \right\} + (1 - \lambda_r) \left\{ \langle v_r, \phi_r(x_i^r) \rangle + d_r \right\} \right) = p_i^r - \xi_i^r,
$$

and, using Equations (4.44) and (4.45) to substitute w and v_r, and (4.48) to replace ξ_i^r, we can express this condition as

$$
\begin{aligned}
& y_i^r \left(\lambda_r \left\{ \left\langle \sum_{s=1}^{T} \lambda_s \sum_{j=1}^{m_s} \alpha_j^s y_j^s \phi(x_j^s), \phi(x_i^r) \right\rangle \right\} \right) \\
& + y_i^r \left((1 - \lambda_r) \left\{ \left\langle (1 - \lambda_r) \sum_{j=1}^{m_r} \alpha_j^r y_j^r \phi_r(x_j^r), \phi_r(x_i^r) \right\rangle + d_r \right\} \right) = p_i^r - \frac{1}{C} \alpha_i^r
\end{aligned}
$$

(4.49)

and hence

$$
\left(\lambda_r \left\{ \sum_{s=1}^{T} \lambda_s \sum_{j=1}^{m_s} \alpha_j^s y_i^r y_j^s k(x_j^s, x_i^r) + y_i^r b \right\} \right)
$$
$$
+ \left((1 - \lambda_r) \left\{ (1 - \lambda_r) \sum_{j=1}^{m_r} \alpha_j^r y_i^r y_j^r k_r(x_j^r, x_i^r) + y_i^r d_r \right\} \right) + \frac{1}{C} \alpha_i^r = p_i^r. \tag{4.50}
$$

These equations, alongside those corresponding to the conditions (4.46) can be expressed as the following system of equations

$$
\left[\begin{array}{c|c|c} 0 & \mathbf{0}_T^\intercal & \boldsymbol{y}^\intercal \\ \hline \mathbf{0}_T & 0_{T \times T} & A^\intercal Y \\ \hline \boldsymbol{y} & YA & \widehat{Q} + \frac{1}{C} I_n \end{array} \right] \begin{pmatrix} b \\ d_1 \\ \vdots \\ d_T \\ \boldsymbol{\alpha} \end{pmatrix} = \begin{pmatrix} 0 \\ \mathbf{0}_T \\ \boldsymbol{p} \end{pmatrix}, \tag{4.51}
$$

where $\mathbf{0}_T$ is the zero vector of length T, $0_{T \times T}$ is the $T \times T$ zero matrix, and we use the matrices

$$
\underset{T \times N}{A^\intercal} = \begin{pmatrix} \overbrace{1 \dots 1}^{m_1} & \dots & \overbrace{0 \dots 0}^{m_T} \\ \ddots & \ddots & \ddots \\ 0 \dots 0 & \dots & 1 \dots 1 \end{pmatrix}, \quad \underset{N \times N}{Y} = \begin{pmatrix} y_1^1 & 0 & \dots & 0 \\ 0 & y_2^1 & \dots & 0 \\ \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & \dots & y_{m_T}^T \end{pmatrix}.
$$

Again, $\widehat{Q}$ is the convex MTL kernel matrix defined in (4.27) and, as in the L2-SVM, a diagonal term $\frac{1}{C} I_n$ is added.

In contrast to the L1 and L2-SVM, here, in the primal problem the inequalities are replaced by equalities, which removes the bounds of the dual coefficients, thus leading to a problem that can be solved as a linear system of equations.

4.3 Optimal Convex Combination of Pre-trained Models

A natural alternative to the convex MTL formulation that we have developed is to directly combine pre-trained models in a convex manner. That is, given a model $g(\cdot)$ trained with the data from all tasks, and task-specific models $g_r(\cdot)$ that have been trained with only the data from the corresponding task, for each task $r = 1, \dots, T$ we can define the combination

$$
h_r(\cdot) = \lambda_r g(\cdot) + (1 - \lambda_r) g_r(\cdot).
$$

These are models that combine a common and task-specific models that have been trained separately. Since both $g(\cdot)$ and $g_r(\cdot)$ are fixed functions, the training phase only involves the process of learning the parameters $\lambda_r, r = 1, \dots, T$. The goal is to select the parameters λ_r that minimize the training error

$$
\mathcal{E}(\lambda_1, \dots, \lambda_T) = \sum_{r=1}^{T} \sum_{i=1}^{m_r} \ell(\lambda_r g(x_i^r) + (1 - \lambda_r) g_r(x_i^r), y_i^r),
$$

which depends on the choice of the loss function ℓ. In Ruiz et al. (2021b) we consider the training error with four popular loss functions:

- Hinge loss (classification)

$$\mathcal{E}(\lambda_1, \ldots, \lambda_T) = \sum_{r=1}^{T} \sum_{i=1}^{m_r} \left[1 - y_i^r \left\{ \lambda_r g(x_i^r) + (1 - \lambda_r) g_r(x_i^r) \right\} \right]_+. \qquad (4.52)$$

- Squared hinge loss (classification)

$$\mathcal{E}(\lambda_1, \ldots, \lambda_T) = \sum_{r=1}^{T} \sum_{i=1}^{m_r} \left[1 - y_i^r \left\{ \lambda_r g(x_i^r) + (1 - \lambda_r) g_r(x_i^r) \right\} \right]_+^2. \qquad (4.53)$$

- Absolute loss (regression)

$$\mathcal{E}(\lambda_1, \ldots, \lambda_T) = \sum_{r=1}^{T} \sum_{i=1}^{m_r} \left| y_i^r - \left\{ \lambda_r g(x_i^r) + (1 - \lambda_r) g_r(x_i^r) \right\} \right|. \qquad (4.54)$$

- Squared loss (regression)

$$\mathcal{E}(\lambda_1, \ldots, \lambda_T) = \sum_{r=1}^{T} \sum_{i=1}^{m_r} \left(y_i^r - \left\{ \lambda_r g(x_i^r) + (1 - \lambda_r) g_r(x_i^r) \right\} \right)^2. \qquad (4.55)$$

Observe that using these training errors we can cover the kernel methods that we have considered. The L1-SVM uses the hinge loss for the classification error and the absolute error can be seen as a special case of the ϵ-insensitive loss, which is the regression error of the L1-SVM, when $\epsilon = 0$. The L2-SVM used the squared hinge loss for classification and, again, the squared loss is a special case of the regression error when $\epsilon = 0$. Finally, the squared loss function is used in the LS-SVM for both regression and classification.

4.3.1   Unified Formulation

Observe that in the classification errors, with the hinge loss one in (4.52) and with the squared hinge loss one in (4.53), each term of the sum has the form

$$u \left(1 - \left\{ \lambda_r g(x_i^r) + (1 - \lambda_r) g_r(x_i^r) \right\} y_i^r \right)$$

with u being either the positive part $u(\cdot) = [\cdot]_+$, or squared positive part $u(\cdot) = [\cdot]_+^2$. With some algebra, these terms can also be expressed as

$$u \left(\lambda_r \left\{ y_i^r (g_r(x_i^r) - g(x_i^r)) \right\} + 1 - y_i^r g_r(x_i^r) \right).$$

In the case of regression errors, that is, the one with the absolute loss of (4.54) and with the squared loss in (4.55), the terms in the sum are

$$u \left(\lambda_r g(x_i^r) + (1 - \lambda_r) g_r(x_i^r) - y_i^r \right)$$

with $u(\cdot) = |\cdot|$ and $u = (\cdot)^2$ for the absolute and squared loss, respectively. Again, these terms can be expressed as

$$u\left(\lambda_r\left\{y_i^r(g_r(x_i^r) - g(x_i^r))\right\} + 1 - y_i^r g_r(x_i^r)\right)$$

Then, in both regression and classification settings, the error can be expressed as

$$\sum_{r=1}^{T}\sum_{i=1}^{m_r} u(\lambda_r c_i^r + d_i^r),$$

where for the classification setting we use

$$c_i^r = y_i^r(g_r(x_i^r) - g(x_i^r)), \ d_i^r = 1 - y_i^r g_r(x_i^r), \tag{4.56}$$

and for the regression one

$$c_i^r = g(x_i^r) - g_r(x_i^r), \ d_i^r = g_r(x_i^r) - y_i^r. \tag{4.57}$$

Recall that the functions $g(\cdot)$ and $g_r(\cdot)$ are fixed, and we want to learn the optimal parameters $\lambda_1^*, \ldots, \lambda_T^*$. Also, observe that each particular λ_r is only present in the training error terms concerning task r. That is, using the change of variables (4.56) and (4.57), we can consider the for each task $r = 1, \ldots, T$ the problem

$$\underset{\lambda_r \in [0,1]}{\arg\min} \mathcal{J}(\lambda_r) = \sum_{i=1}^{m_r} u(\lambda_r c_i^r + d_i^r),$$

where, for a simpler formulation, we remove the task index, that is

$$\underset{\lambda \in [0,1]}{\arg\min} \mathcal{J}(\lambda) = \sum_{i=1}^{m} u(\lambda c_i + d_i), \tag{4.58}$$

where u can be different functions depending on the selected loss: the positive part for the hinge loss, the squared positive part for the squared hinge loss, and the absolute and squared functions for their corresponding regression losses.

These functions, except for the squared loss, are not differentiable in their entire domain, but piece-wise differentiable, so we will use the subdifferential when necessary. In particular, let $\lambda_{(1)} < \ldots < \lambda_{(p)}$ be the sorted list of points where $\mathcal{J}$ is not differentiable, we will refer to these points as elbows. To find the optimal parameters λ_r^* we will consider the subdifferential of $\mathcal{J}(\lambda)$ in the elbows, that is, the set $\partial\mathcal{J}(\lambda) = \{c \in \mathbb{R}, \mathcal{J}(z) - \mathcal{J}(\lambda) \leq c(z - \lambda) \ \forall z\}$. Using the Fermat rule (Bauschke and Combettes, 2011), we have

$$\lambda^* = \underset{0 \leq \lambda \leq 1}{\arg\min} \mathcal{J}(\lambda) \iff (0 \in \partial\mathcal{J}(\lambda^*) \text{ and } \lambda^* \in (0,1)) \text{ or } \lambda^* = 0 \text{ or } \lambda^* = 1.$$

To compute $\partial\mathcal{J}(\lambda)$ we use that all functions $u(\lambda c_i + d_i)$ share the domain, that is, $\lambda \in [0,1]$, and therefore the subdifferential of the sum is the sum of the subdifferentials.

Moreover, a result that is common for the linear loss functions is presented in Ruiz et al. (2021b, Proposition 1).

Proposition 2. For the absolute value and hinge losses, $\partial J(\lambda)$ is single valued and constant between the elbows λ_j. Moreover, if $\partial J(\lambda) = J'(\lambda) = \gamma$ for some λ between two consecutive distinct elbows, i.e., $\lambda_{(j)} < \lambda < \lambda_{(j+1)}$, then $\gamma \in \partial J(\lambda_{(j)})$ and $\gamma \in \partial J(\lambda_{(j+1)})$.
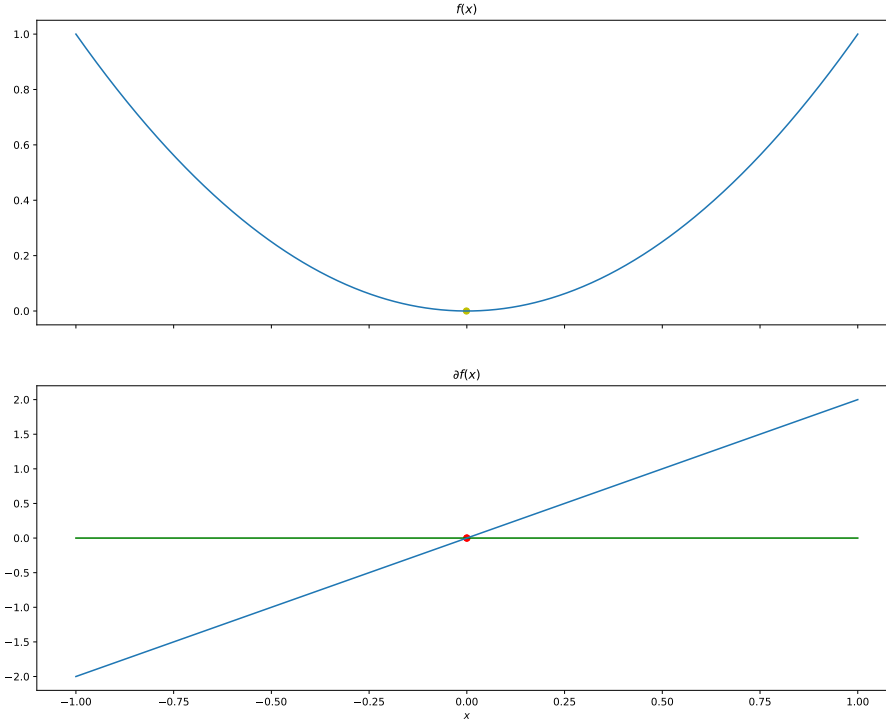
FIGURE 4.1: Squared value function (top) and its corresponding subgradient (bottom). The green line represents the 0 constant function. The yellow dot indicates the point minimizing the function.

This proposition states that in the case when the linear losses are considered, absolute and hinge losses, we find the optimal value λ^* in the elbows. In the following subsections, we show how to select the optimal parameters λ_r^*, i.e. those minimizing the training error, for each of the losses considered.

4.3.2 Optimal λ selection

Squared loss

This is the simplest case, since the squared function, $f(x) = x^2$, is differentiable, and its derivative is $\nabla f(x) = 2x$; both are shown in Figure 4.1. Using the formulation of (4.58), the training error corresponding to the squared loss is

$$\underset{\lambda \in [0,1]}{\arg\min} \mathcal{J}(\lambda) = \sum_{i=1}^{m} (\lambda c_i + d_i)^2. \tag{4.59}$$

In this case, $\mathcal{J}(\lambda)$ is a differentiable function, and its derivative is

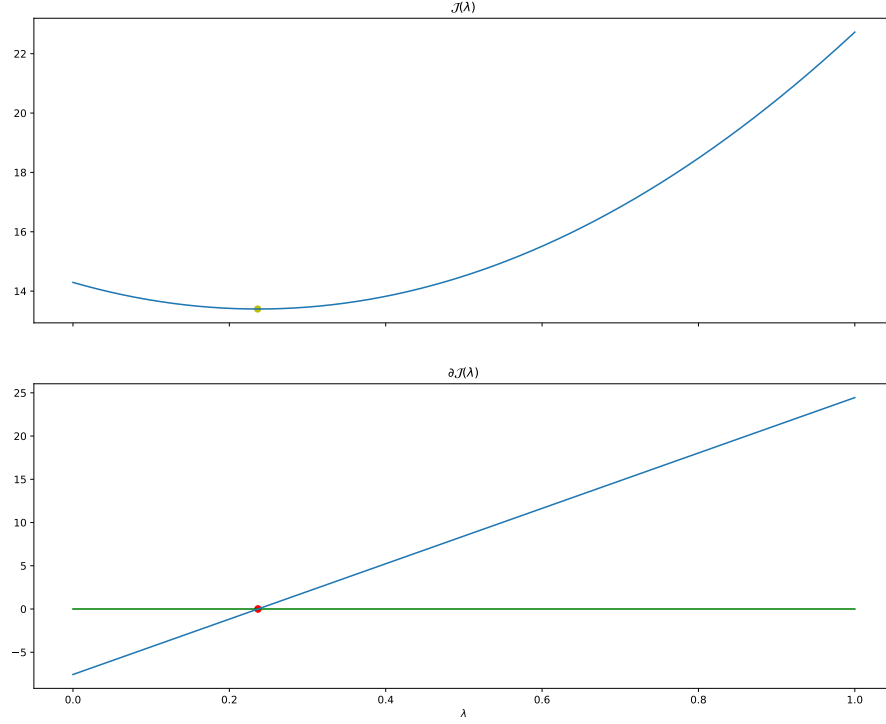$$\mathcal{J}'(\lambda) = \sum_{i=1}^{m} 2c_i(\lambda c_i + d_i).$$

FIGURE 4.2: Error using the squared loss function (top) and its corresponding subgradient (bottom). The green line represents the 0 constant function. The yellow dot is the point minimizing the error, and whose corresponding subgradient contains the value 0.

Solving $\mathcal{J}'(\lambda) = 0$ results in

$$\lambda' = -\frac{\sum_{i=1}^{m} d_i c_i}{\sum_{i=1}^{m} (c_i)^2},$$

and the optimum is hence $\lambda^* = \max(0, \min(1, \lambda'))$.

In Figure 4.2 the error function using 10 random pairs (c_i, d_i) and its corresponding differential is shown.

Absolute loss.
The absolute function

$$f(x) = \begin{cases} -x, & x \leq 0, \\ x, & x > 0 \end{cases}$$

is not differentiable at 0, but the subgradient can be expressed at any point as

$$f(x) = \begin{cases} -1, & x < 0, \\ [-1, 1], & x = 0, \\ 1, & x > 0, \end{cases}$$
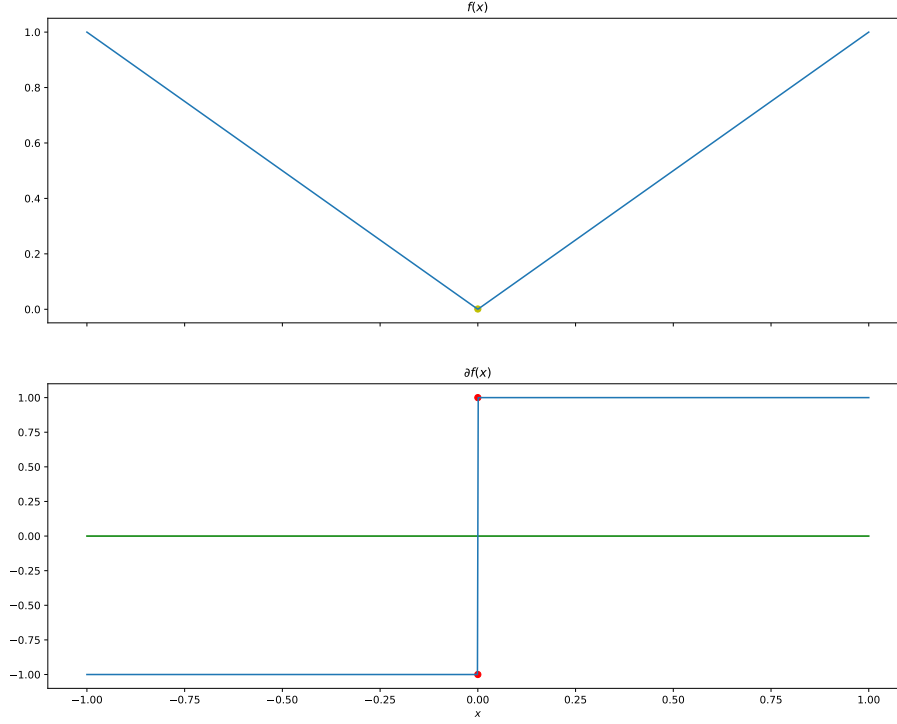
FIGURE 4.3: Absolute value function (top) and its corresponding subgradient (bottom). The green line represents the 0 constant function. The yellow dot indicates the point minimizing the function.

which is illustrated in Figure 4.3. Using the formulation of (4.58), the training error corresponding to the absolute value loss is

$$\arg\min_{\lambda \in [0,1]} \mathcal{J}(\lambda) = \sum_{i=1}^{m} |\lambda c_i + d_i|. \qquad (4.60)$$

Observe that in each term of the sum the subdifferential is

$$\partial |\lambda c_i + d_i| = \begin{cases} -|c_i|, & \lambda c_i + d_i < 0, \\ [-|c_i|, |c_i|], & \lambda c_i + d_i = 0, \\ |c_i|, & \lambda c_i + d_i > 0. \end{cases}$$

The elbows are obtained using the values $\frac{-d_i}{c_i}$, that can be clipped and sorted to get $\lambda_{(1)} < \ldots < \lambda_{(m)}$. In Ruiz et al. (2021b, Proposition 2) we present the following result to get the optimal λ^*.

Proposition 3 (Optimal λ^* with absolute value loss). In problem (4.60) $\lambda^* = 0$ is optimal *iff*

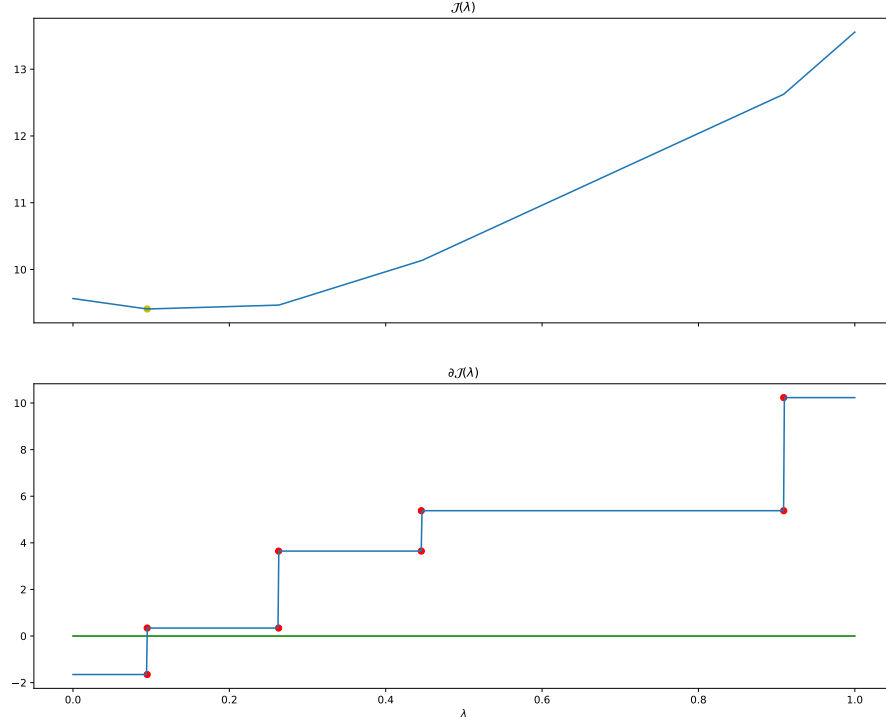$$-\sum_{j:\lambda_{(j)}<0} |c_j| + \sum_{j:\lambda_{(j)}>0} |c_j| < 0.$$

FIGURE 4.4: Error using the absolute loss function (top) and its corresponding subgradient (bottom). The green line represents the 0 constant function. Red dots mark the extremes of the subdifferential intervals of non-differentiable points. The yellow dot is the point minimizing the error, and whose corresponding subgradient contains the value 0.

If this condition does not hold, $\lambda^* \in (0,1)$ is optimal *iff* λ^* is a feasible elbow, that is, $0 < \lambda^* = \lambda_{(k)} < 1$ for some $k = 1, \ldots, m$, and

$$- \sum_{j:\lambda_{(j)} < \lambda_{(k)}} |c_j| + \sum_{j:\lambda_{(j)} > \lambda_{(k)}} |c_j| \in [-|c_k|, |c_k|]. \tag{4.61}$$

If none of the previous conditions hold, then $\lambda^* = 1$ is optimal.

The meaning of this proposition is depicted in Figure 4.4, where the error and its corresponding subgradient is computed for 10 random pairs (c_i, d_i). It can be seen that the subdifferential is constant between elbows, and the subgradient at elbow $\lambda_{(k)}$ is $[s_k + c_k, s_k - c_k]$ with

$$s_k = - \sum_{j:\lambda_{(j)} < \lambda_{(k)}} |c_j| + \sum_{j:\lambda_{(j)} > \lambda_{(k)}} |c_j|$$

. Then, the parameter λ^* is optimal when $0 \in \partial J(\lambda^*)$, that is, $0 \in [s_k + c_k, s_k - c_k]$.
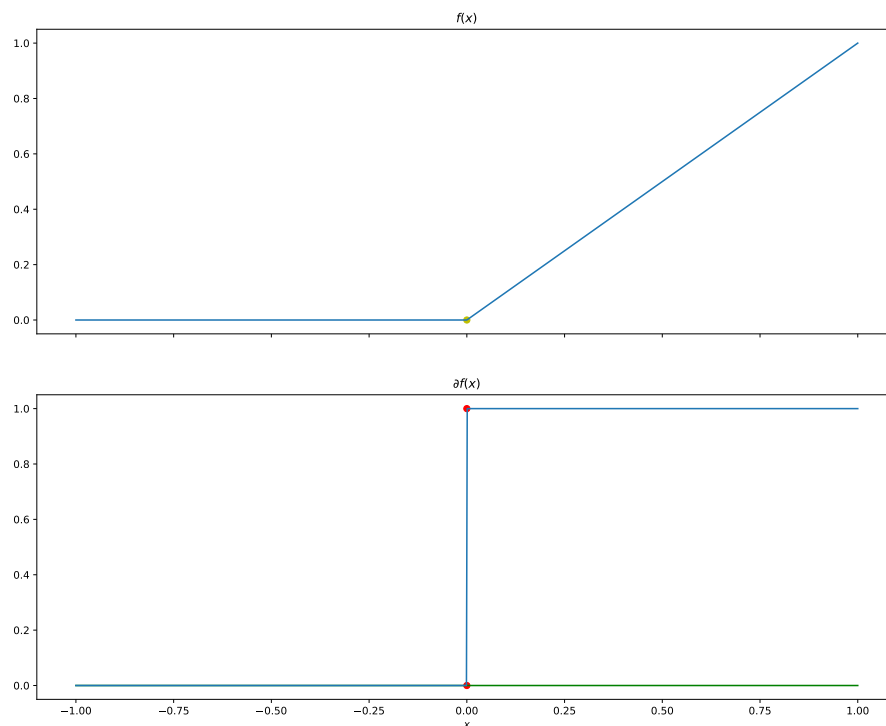
FIGURE 4.5: Positive part function (top) and its corresponding subgradient (bottom). The green line represents the 0 constant function. The yellow dot indicates the point minimizing the function.

Hinge loss.

The positive part function

$$f(x) = \begin{cases} 0, & x \le 0, \\ x, & x > 0 \end{cases}$$

is not differentiable at 0, but, again, the subgradient can be expressed at any point as

$$f(x) = \begin{cases} 0, & x < 0, \\ [0, 1], & x = 0, \\ 1, & x > 0, \end{cases}$$

which is illustrated in Figure 4.5. Using the formulation of (4.58), the training error corresponding to the hinge loss is

$$\underset{\lambda \in [0,1]}{\arg\min} \, \mathcal{J}(\lambda) = \sum_{i=1}^{m} [\lambda c_i + d_i]_+ \,. \tag{4.62}$$

Observe that in each term of the sum the subdifferential is

$$\partial \left[\lambda c_i + d_i\right]_+ = \begin{cases} 0, & \lambda c_i + d_i < 0, \\ [\min(0, c_i), \max(0, c_i)], & \lambda c_i + d_i = 0, \\ c_i, & \lambda c_i + d_i > 0. \end{cases}$$

That is, the elbows are related to the values $\frac{-d_i}{c_i}$, that can be clipped and sorted to obtain the elbows $\lambda_{(1)} < \ldots < \lambda_{(m)}$. In Ruiz et al. (2021b, Proposition 2) we present the following result to get the optimal λ^*.

Proposition 4 (Optimal λ^* with hinge loss). In (4.62), $\lambda^* = 0$ is optimal iff

$$- \sum_{j:\lambda_{(j)} < 0} \max(0, c_{(j)}) - \sum_{\lambda_{(j)} > 0} \min(0, c_{(j)}) \leq 0. \tag{4.63}$$

If this condition does not hold, a value $\lambda^* \in (0, 1)$ is optimal for problem (4.62) *iff* λ^* is a feasible elbow, that is, $0 < \lambda^* = \lambda_{(k)} < 1$ for some $k = 1, \ldots, m$, and

$$- \sum_{j:\lambda_{(j)} < \lambda_{(k)}} \max(0, c_{(j)}) - \sum_{j:\lambda_{(j)} > \lambda_{(k)}} \min(0, c_{(j)}) \in \left[\min(0, c_{(k)}), \max(0, c_{(k)})\right]. \tag{4.64}$$

If none of the previous conditions hold, then $\lambda^* = 1$ is optimal.

Again, the results of the proposition are illustrated in Figure 4.6, where 10 pairs (c_i, d_i) are randomly sampled, and the corresponding error function $\mathcal{J}(\lambda)$ and its subgradient are shown. Here, similarly to the absolute loss case, the subgradient is constant between elbows and at each elbow $\lambda_{(k)}$, the subgradient is the interval $[s - \min(0, c_{(k)}), s + \max(0, c_{(k)})]$ with

$$s = - \sum_{j:\lambda_{(j)} < \lambda_{(k)}} \max(0, c_{(j)}) - \sum_{j:\lambda_{(j)} > \lambda_{(k)}} \min(0, c_{(j)});$$

then, an elbow $\lambda_{(k)}$ is optimal if and only if $0 \in [s - \min(0, c_{(k)}), s + \max(0, c_{(k)})]$.

Squared hinge loss.
The squared positive part function is

$$f(x) = \begin{cases} 0, & x \leq 0, \\ x^2, & x > 0. \end{cases}$$

and its gradient can be expressed at any point as

$$f(x) = \begin{cases} 0, & x \leq 0, \\ 2x, & x > 0. \end{cases}$$

which is illustrated in Figure 4.5. Although the gradient can be defined at any point, the definition by parts makes it difficult to get the optimal value λ^* that minimizes the corresponding training error. Using the formulation of (4.58), the training error
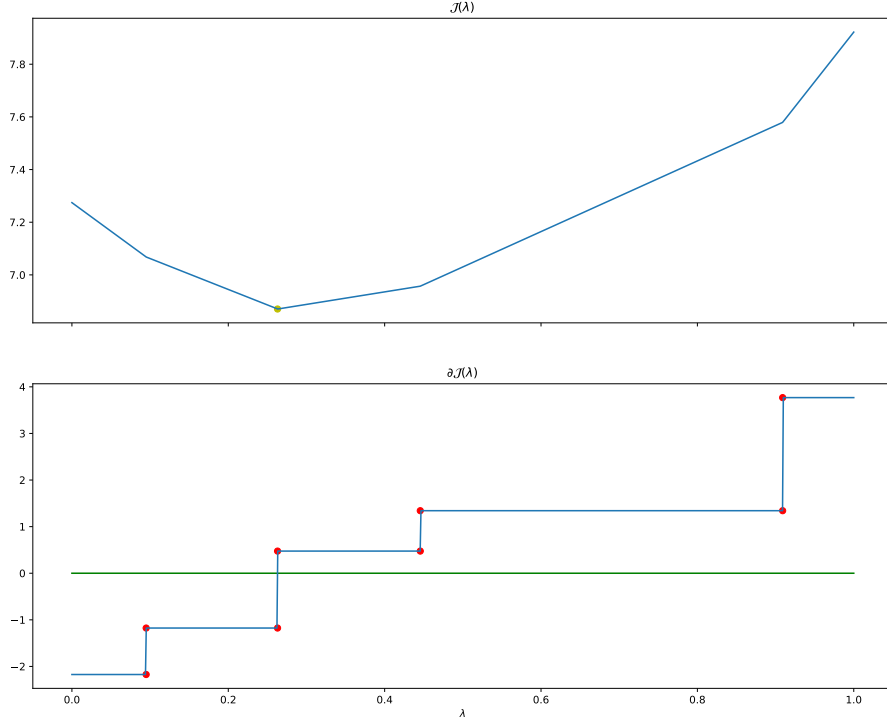
FIGURE 4.6: Error using the hinge loss function (top) and its corresponding subgradient (bottom). The green line represents the 0 constant function. Red dots mark the extremes of the subdifferential intervals of non-differentiable points. The yellow dot is the point minimizing the error, and whose corresponding subgradient contains the value 0.

corresponding to the hinge loss is

$$\arg\min_{\lambda \in [0,1]} \mathcal{J}(\lambda) = \sum_{i=1}^{m} [\lambda c_i + d_i]_+^2 . \tag{4.65}$$

In each term of the sum the subdifferential is

$$\partial [\lambda c_i + d_i]_+^2 = \begin{cases} 0 & , \lambda c_i + d_i \le 0 \\ 2c_i(\lambda c_i + d_i) & , \lambda c_i + d_i > 0 \end{cases} .$$

The elbows are again related to the values $\frac{-d_i}{c_i}$, that can be clipped and sorted to obtain the elbows $\lambda_{(1)} < \ldots < \lambda_{(m)}$. In Ruiz et al. (2021b, Proposition 2) we give the following result to get the optimal λ^* when using the squared hinge loss.

Proposition 5 (Optimal λ^* with squared hinge loss). In (4.65), $\lambda^* = 0$ is optimal iff

$$-\frac{\sum_{j:\lambda_j<0} \max(0, c_j)d_j + \sum_{j:\lambda_j>0} \min(0, c_j)d_j}{\sum_{j:\lambda_j<0} \max(0, c_j)^2 + \sum_{j:\lambda_j>0} \min(0, c_j)^2} < 0.$$
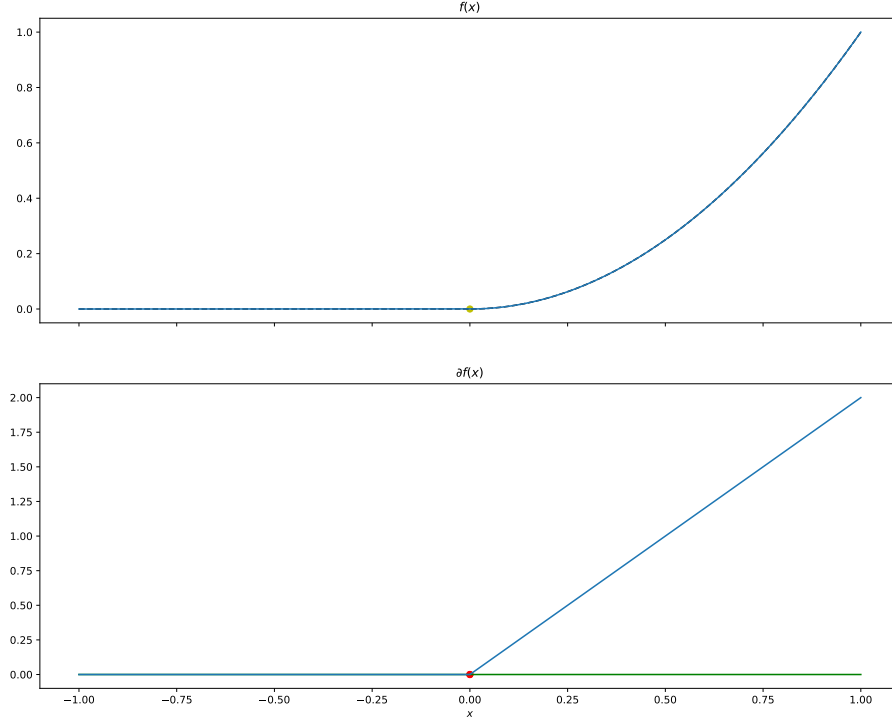
FIGURE 4.7: Positive part function (top) and its corresponding subgradient (bottom). The green line represents the 0 constant function. The yellow dot indicates one point minimizing the function.

If this condition does not hold, consider the extended sorted list of feasible elbows $\lambda_{(p-1)} = 0 \leq \lambda_{(p)} \leq \ldots, \lambda_{(q)} \leq \lambda_{(q+1)} = 1$, with $1 \leq p, q \leq m$, and define $\widehat{\lambda}_k$ for $k = p - 1, \ldots, q$ as

$$\widehat{\lambda}_k = -\frac{\sum_{j:\lambda_{(j)}<\lambda_{(k)}} \max(0, c_j)d_j + \sum_{j:\lambda_{(j)}>\lambda_{(k)}} \min(0, c_j)d_j}{\sum_{j:\lambda_{(j)}<\lambda_{(k)}} \max(0, c_j)^2 + \sum_{j:\lambda_{(j)}>\lambda_{(k)}} \min(0, c_j)^2}. \tag{4.66}$$

Then, if $\lambda_{(k)} \leq \widehat{\lambda}_k \leq \lambda_{(k+1)}$ for some $k = 0, \ldots, q$, then $\lambda^* = \widehat{\lambda}_k$ is optimal. Finally, if none of the previous conditions holds, (4.65) has a minimum at $\lambda^* = 1$.

As with the other losses, the error function and respective gradient for 10 random pairs (c_i, d_i) are represented in Figure 4.8. Now, the derivative is not constant between elbows, but a linear function. Between elbows $\lambda_{(k)}$ and $\lambda_{(k+1)}$ the derivative is

$$\lambda \left(\sum_{j:\lambda_{(j)}<\lambda_{(k)}} \max(0, c_j)^2 + \sum_{j:\lambda_{(j)}>\lambda_{(k)}} \min(0, c_j)^2 \right) + \sum_{j:\lambda_j<0} \max(0, c_j)d_j + \sum_{j:\lambda_j>0} \min(0, c_j)d_j,$$

and it is 0 for the value $\widehat{\lambda}_k$ shown in (4.65); however, that value is optimal only if $\lambda_{(k)} \leq \widehat{\lambda}_k \leq \lambda_{(k+1)}$.
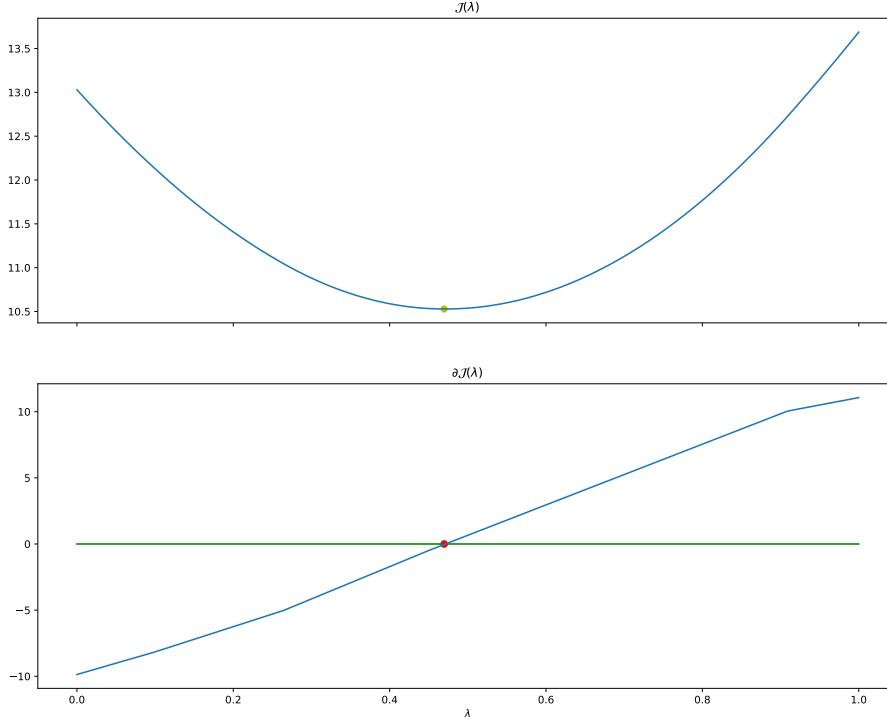
FIGURE 4.8: Error using the squared hinge loss function (top) and its corresponding subgradient (bottom). The green line represents the 0 constant function. The yellow dot is the point minimizing the error, whose corresponding derivative is 0.

4.3.3 Experiments

In this subsection we show the experiments used to test the convex MTL formulation with kernel models, which have been presented in (Ruiz et al., 2019) and (Ruiz et al., 2021b). First, following the work in (Ruiz et al., 2019), experiments comparing convex and additive MTL formulations in the L1-SVM are described. Then, the experiments of (Ruiz et al., 2021b) are exposed, where all convex MTL models using L1, L2 and LS-SVMs as well as optimal convex combination of pre-trained models are compared over multiple real problems.

Before diving into the results, some technical details have to be explained. The convex MTL formulation, as presented in Equation (4.17), uses task-specific hyperparameters λ_r, as well as common and task-specific kernels, with their corresponding hyperparameters; however, the selection of many hyperparameters is always a challenge because of the curse of dimensionality. To select the hyperparameters $p_1, \ldots, p_L$ of a learning algorithm using CV the following problem is solved:

$$p_1^*, \ldots, p_L^* = \underset{p_i \in \mathcal{S}_{p_i}, i=1,\ldots,L}{\arg\min} \sum_{j=1}^{F} \rho(\mathcal{A}(p_1, \ldots, p_L; (X_{\text{train}}^j, y_{\text{train}}^j)); (X_{\text{val}}^j, y_{\text{val}}^j)), \qquad (4.67)$$

where ρ is some measure; $\mathcal{A}$ is our choice of learning algorithm; $p_1, \ldots, p_L$ are the parameters on which $\mathcal{A}$ depends and $\mathcal{S}_{p_i}$ a feasible space for the parameter p_i; and $(X_{\text{train}}^j, y_{\text{train}}^j), (X_{\text{val}}^j, y_{\text{val}}^j)$ are the training and validation sets, respectively. We are using F folds, that is, for $j = 1, \ldots, F$ we select different training and validation sets and we train our algorithm $\mathcal{A}$ on the training set $(X_{\text{train}}^j, y_{\text{train}}^j)$; then we use ρ to measure the wellness of our model on $(X_{\text{val}}^j, y_{\text{val}}^j)$. Observe that our search space is the product space $\mathcal{S}_{p_1} \times \ldots \times \mathcal{S}_{p_L}$, so the difficulty of selecting an optimal combination of hyperparameters scales exponentially with the number of such parameters.

In a standard Gaussian kernel SVM, the definition of the classification problem depends on two hyperparameters: C and γ. For regression problems we also add a third parameter: ϵ. That is, to select the hyperparameters of a standard SVM for a single task we have to solve the problem

$$C^*, \gamma^*(, \epsilon^*) = \underset{\substack{C \in \mathcal{S}_C; \\ \gamma \in \mathcal{S}_\gamma; \\ (\epsilon \in \mathcal{S}_\epsilon);}}{\arg\min} \sum_{j=1}^{F} \rho(\mathcal{A}(C, \gamma(, \epsilon); (X_{\text{train}}^j, y_{\text{train}}^j)); (X_{\text{val}}^j, y_{\text{val}}^j)),$$

which is typically feasible by using a grid search method in the space $\mathcal{S}_C \times \mathcal{S}_\gamma (\times \mathcal{S}_\epsilon)$.

However, using the convex MTL formulation, we have the following hyperparameters:

- the regularization parameter C,

- the common kernel width γ and task-specific ones $\gamma_1, \ldots, \gamma_T$,

- the convex combination parameters $\lambda_1, \ldots, \lambda_T$,

- and possibly the ϵ parameter.

That is, there are at least $2T + 2$ hyperparameters that have to be selected. Even with $T = 2$, a search space of dimension 6 is computationally unfeasible to cover with grid search.

To solve this difficulty and select the optimal parameters for the convex MTL formulation we use the next strategy. For the kernel widths, we proceed as follows: we first hyperparametrize the corresponding CTL and ITL kernel models, which have a common and task-specific kernel widths, respectively. This step would give also optimal C and ϵ values, but we discard them. Observe that in the CTL approach, where a single virtual task is solved, and in the ITL approach, where we solve the tasks independently, we have 2 or 3 hyperparameters. We solve the corresponding problems, which are represented in Equation (4.67), using grid search, and we obtain optimal common C^*, γ^* and task-specific ones C_r^*, γ_r^* for $r = 1, \ldots, T$ (also the respective ϵ^* values in the regression setting). Then, we reutilize the optimal widths for these models, and fix them in the MTL one. Moreover, for the convex combination parameters we use a single λ for all tasks, that is, $\lambda_1 = \ldots = \lambda_T = \lambda$. With these considerations, the problem to select the remaining hyperparameters is

$$C^*, \lambda^*(, \epsilon^*) = \underset{\substack{C \in \mathcal{S}_C; \\ \lambda \in \mathcal{S}_\lambda; \\ (\epsilon \in \mathcal{S}_\epsilon);}}{\arg\min} \sum_{j=1}^{F} \rho(\mathcal{A}(C, \lambda, \gamma^*, \gamma_1^*, \ldots, \gamma_T^*(, \epsilon); (X_{\text{train}}^j, y_{\text{train}}^j)); (X_{\text{val}}^j, y_{\text{val}}^j)),$$

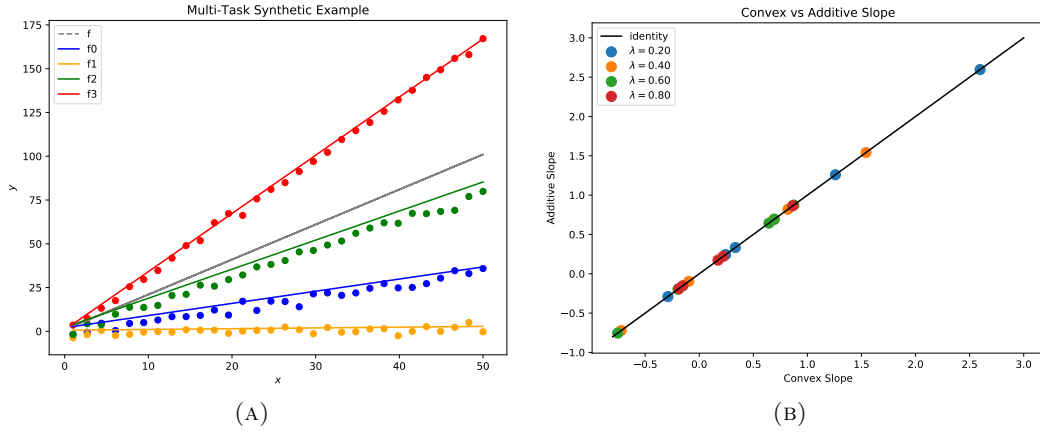where we have a maximum of 3 hyperparameters.

FIGURE 4.9: (a): Synthetic example dataset, where the data of each task (corresponding to a different function f_i) are represented with a different color. (b): Comparison of the weights obtained by the convex and additive approaches.
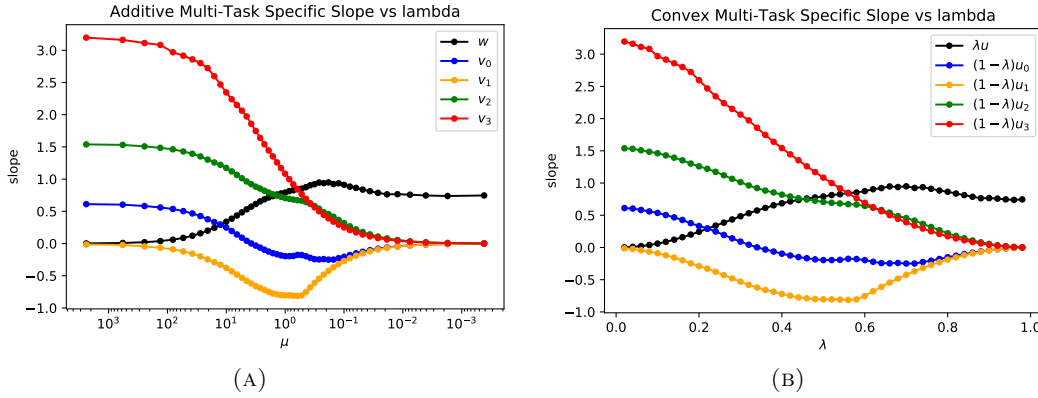
FIGURE 4.10: Convex (b) and additive (a) MTLSVR slope estimates weights as a function of λ. We represent the common part of the models, w for the additive and λu for the convex, as well as each task specific part v_t and $(1 - \lambda)u_t$.

In summary, these are the two adjustments that we make to carry out the experiments shown in this subsection. The first adjustment concerning kernel widths does not change the model definition, but it assumes that kernel widths that are good for CTL or ITL models are good for the MTL one. Consider the Gaussian kernel

$$k(x, y) = \frac{1}{\sqrt{2\pi}\sigma} \exp\left(-\frac{1}{2\sigma^2}(x - y)^2\right),$$

where σ is the kernel width and it regulates "how wide is the influence of each point"; then, kernel widths are not that descriptive of the model but of the data used, which makes the reutilization of kernel widths a sensible decision. The combined data from all tasks, which is used in CTL approach, is well "characterized" by a width σ^* so we expect that this width is also useful for the common part of the MTL model. The same reasoning applies to task-specific data and ITL approach. The second adjustment does change the models definition, using a single λ that determines the specifity of all models. However, we expect that the possible difference in specifity among tasks can be corrected in the training process by selecting larger task-specific weights v_r if necessary.

Comparison of Convex and Additive Formulations

In (Ruiz et al., 2019), we illustrate the results of equivalence between the additive and convex formulations in an empirical way. To do this we generate a synthetic problem, shown in Figure 4.9, left, as a four linear regression tasks problem. We use four different functions f_0, f_1, f_2, f_3 by considering a base function $f(x) = 2x - 1$ with slope $m = 2$ and bias $n = 1$, then we sample $z_m^r, z_n^r \sim N(0, 1)$ for each task $r = 0, \ldots, 3$, and, create the r-th slope and bias by adding these Gaussian samples, i.e., $m_r = m + z_m^r$ and $n_r = n + z_n^r$. Also, we sample the noise σ_r for each task uniformly in $(0, 5)$. Then, the r-th task consists on estimating m_r and n_r from data. To do this, for each task we uniformly sample 30 points $x_i^r \in [1, 50)$, and the target values are defined as $y_i^r = f_r(x_i^r) + \epsilon_r^i$ where $\epsilon_r^i \sim N(0, \sigma_r)$. Combining all tasks, there are 120 data points, 30 for each task, that we split randomly in a task-stratified way: two thirds, i.e. 80 points are used for training, and the rest are used for testing purposes; in this division, the task size proportions are kept constant, that is $1/4$ for each task, in both the train and test sets. With this synthetic problem, we train four convex MTL models corresponding to values of $\lambda \in \{0.2, 0.4, 0.6, 0.8\}$; and we also train the corresponding equivalent additive MTL models, in which we set $\mu = (1 - \lambda)/\lambda^2$ and $C_{\text{add}} = (1 - \lambda)^2 C_{\text{conv}}$, as shown in Proposition 1. Our goal is to compare the influence of μ with that of λ in the final models that are obtained. To do that, we need C to be small enough so the regularization and, therefore, μ are relevant; the value $C_{\text{conv}} = 10^{-2}$ is found to be useful. We also consider linear kernels, so we can obtain the primal coefficients, that is, the slopes, and compare those obtained using the additive and convex formulations. In Figure 4.9, right, we show the estimated slopes for each of the λ values considered with a different color. In the x axis we represent the slopes estimated using the convex approach and in the y axis those estimated with the additive one. There are four dots of each color, corresponding to each of the tasks considered in our synthetic problem. We can observe that all dots lie in the diagonal line corresponding to the identity function, as we expected from the equivalence result from Proposition 1.

To further compare the two formulations, we visualize how the change on the hyperparameters imposes a change on the final models. Using the already described synthetic problem of Figure 4.9, left, we select values of λ ranging from 0 to 1, and their corresponding values $\mu = (1 - \lambda)^2/\lambda^2$, and fit convex and additive MTL linear SVMs using these values. In Figure 4.10 we show the coefficients obtained for the common and task-specific parts with the additive formulation (left) and convex one (right). We can observe how both graphics show a similar behaviour, with the common model starting in 0 when μ is large or $\lambda = 0$, and, as μ decreases and λ grows, the task-specific parts go to zero and the common model reaches the optimal value for CTL.

However, two facts are noticeable. The first one is the range of hyperparameters needed for each formulation; while the convex formulation always uses $[0, 1]$, with the additive formulation it seems that $(10^{-3}, 10^3)$ is useful in this problem, but we cannot extrapolate to other problems; the second one is the smoother transition of the convex formulation, where the changes are steadily made, while with the additive one the changes look more abrupt.

Performance of Convex MTL and Optimal Convex Combination.

To test the performance of the convex MTL formulation, in (Ruiz et al., 2019) we also conduct experiments with real problems, which we later extend in (Ruiz et al., 2021b). To test our proposal we apply it to three SVM variants: L1, L2 and LS-SVM. For each

variant, we compare the CTL, ITL with our convex MTL formulation. To do this, we use fourteen different problems, six regression problems and eight classification ones.

Models. Considering that LX can stand for L1, LS or LS, we use the following approaches:

- Common Task Learning LX-SVM (CTL-LX): A single LX-SVM fitted using data from all the tasks and does not use task information.

- Independent Task Learning LX-SVM (ITL-LX): Multiple task-specific LX-SVMs, each of which is fitted with the data from its own task.

- Direct Convex Combination of LX-SVMs (cvxCMB-LX): A combination of the best CTL-LX and ITL-LX as described in Subsection 4.3.

- Convex Multi-Task Learning LX-SVM (cvxMTL-LX): The Convex MTL formulations shown in Subsection 4.2.1 for the L1-SVM and in Subsection 4.2.2 for the L2- and LS-SVM.

Problems. To compare these approaches we will use several regression and classification problems. We use a total of six regression problems:

- majorca: The goal is to predict the photovoltaic energy production in Mallorca. The tasks are defined as the energy prediction at each of the 14 hours with sunlight (we remove the night hours).

- tenerife: The goal is to predict the photovoltaic energy production in Tenerife. The tasks are defined as the energy prediction at each of the 14 hours with sunlight (we remove the night hours).

- boston: This is the housing problem in Boston, where the goal is to predict house prices, and the tasks are defined as the predictions in different areas of the city. In Boston we have the houses that are next to the river and those that are not.

- california: This is the housing problem in California, where the goal is also to predict house prices. Here the tasks correspond to 4 different areas of California, related to their distance to the sea.

- abalone: The goal is to predict the number of rings of a specie of marine molluscs. The tasks are male, female or infant specimens.

- crime: The goal is to predict the number of crimes per 100.000 habitants in the U.S., the tasks are the prediction of the crime rate in different states, and we consider nine states.

For the classification setting, we consider eight problems, six of which are generated by applying different task definitions to two different problems.

- landmine: This is a binary classification problem in which the goal is to detect landmines. Detection of different types of landmines determines a different task each.

- binding: This is a binary classification problem where the goal is to determine if a given molecule will bind with peptides. Different molecules define different tasks.

TABLE 4.1: Sample sizes, dimensions and number of tasks of the datasets used.

Dataset	Size	No. feat.	No. tasks	Avg. task size	Min. t. s.	Max. t. s.
majorca	15 330	765	14	1095	1095	1095
tenerife	15 330	765	14	1095	1095	1095
california	19 269	9	5	3853	5	8468
boston	506	12	2	253	35	471
abalone	4177	8	3	1392	1307	1527
crime	1195	127	9	132	60	278
binding	32 302	184	47	687	59	3089
landmine	14 820	10	28	511	445	690
adult_(G)	48 842	106	2	24 421	16 192	32 650
adult_(R)	48 842	103	5	9768	406	41 762
adult_(G, R)	48 842	101	10	4884	155	28 735
compas_(G)	3987	11	2	1993	840	3147
compas_(R)	3987	9	4	997	255	1918
compas_(G, R)	3987	7	8	498	50	1525

TABLE 4.2: Hyperparameters, grids used to select them (when appropriate) and hyperparameter selection method for each model.

	Grid	CTL-L1,2	ITL-L1,2	cvxMTL-L1,2	CTL-LS	ITL-L,S	cvxMTL-LS
C	$\{4^k : -2 \leq k \leq 6\}$	CV	CV	CV	CV	CV	CV
ϵ	$\{\frac{\sigma}{4^k} : 1 \leq k \leq 6\}$	CV	CV	CV	-	-	-
γ_c	$\{\frac{4^k}{d} : -2 \leq k \leq 3\}$	CV	-	CTL-L1,2	CV	-	CTL-LS
γ_s^r	$\{\frac{4^k}{d} : -2 \leq k \leq 3\}$	-	CV	ITL-L1,2	-	CV	ITL-LS
λ	$\{0.1k : 0 \leq k \leq 10\}$	-	-	CV	-	-	CV

- adult: The goal is to predict whether the yearly salary of a particular person is greater than 50K based on sociocultural data. We can define different tasks dividing the population by either gender or race, so we have the problems:

    - ad_{G}: dividing by gender. We consider 2 tasks.
    - ad_{R}: dividing by race. We consider 5 tasks.
    - ad_{G, R}: dividing by both gender and race, so we define a total of 10 tasks.

- compas: The goal is to predict whether the COMPAS algorithm will assign "low" or "high" scores of recidivism to a particular subject. We can also divide the sample by either race or gender, so we obtain:

    - comp_{G}: dividing by gender. We consider 2 tasks.
    - comp_{R}: dividing by race. We consider 4 tasks
    - comp_{G, R}: dividing by both gender and race, so we define a total of 8 tasks.

We give in Table 4.1 a description of the characteristics of each problem.

Experimental Procedure. To obtain the experimental results, we have to select the optimal hyperparameters for each model in a training-validation set and measure their performance on a test set. To do this, we follow the adjustments and experimental procedure described at the beginning of this subsection, reusing the kernel widths from CTL and ITL approaches and limiting the convex MTL formulation to a single parameter λ. In all models considered we use Gaussian kernels, so all features have been scaled to the $[0, 1]$ interval. As previously explained, the hyperparameters for the CTL and ITL approaches in the classification setting are $\{C, \gamma\}$ for all variants L1, L2 and LS-SVM. In the regression setting we have $\{C, \gamma, \epsilon\}$ for the L1 and L2 variants, and $\{C, \gamma\}$ for the LS-SVM. After selecting the optimal kernel widths in the CTL approach σ^*, and the

task-specific widths σ_r^* in the ITL approach, we use these values to fix them in the convex MTL formulation. Then, the hyperparameters that we are considering for CV search in the convex MTL approaches in the classification settings are $\{C, \lambda\}$ for all variants, while in the regression problems they are $\{C, \lambda, \epsilon\}$ for the L1 and L2 variants and $\{C, \lambda\}$ for the LS-SVM. The optimal combination approaches do not have proper hyperparameters, since λ is computed. In all problems, the hyperparameters considered are selected with a grid search using a task-stratified 3-fold CV. That is, given a training-validation set, we divide it in three different folds where the task proportions are kept constant, and we use two of these folds for training the model and evaluate its performance in the remaining one. In the regression problems, we will measure the validation performance using the MAE, see Table 4.3, and MSE, see Table 4.4. Observe that the objective function of L1-SVM based models is more aligned with minimizing the MAE, while those of L2 and LS-SVM based models are more related to minimizing the MSE. In the classification problems, see Table 4.5, we will consider the F1 score to deal with the class-imbalance ratio that we find, for example, in the landmine dataset where we have 200 negative examples for each 13 positive ones. In Table 4.2 we show for each hyperparameter of the considered approaches whether they are selected using a CV procedure or recycling them from other approach, as well as the grids used for the CV search.

We have explained how we select the hyperparameters given a training-validation set, but it is necessary also to describe how we get the final test results that we show on tables. In every problem, except for majorca and tenerife, we will consider three external folds, each with the internal three folds for cross-validation. That is, the whole dataset of each problem is first divided in three task-stratified external folds: F_1, F_2, F_3; then, two folds will form the training-validation set and the third one will be used as the test set. All folds have the same task proportions. There are three different combinations to do this division: $\{F_1, F_2; F_3\}, \{F_1, F_3; F_2\}$ and $\{F_2, F_3; F_1\}$, where the first two folds form the train-validation set and the other one is the test set. In each train-validation set we follow the procedure described above to select the optimal hyperparameters, and the performance model with the optimal hyperparameters will be tested in the remaining fold, i.e., the test set.

The problems of majorca and tenerife have a temporal dependency, so it is not possible to use training data from a time that is ahead of that of the test or validation data. Therefore, we use data from years 2013, 2014 and 2015, each corresponding to train, validation and test sets, respectively. We consider different metrics to measure the test performance. Therefore, in every problem, except majorca and tenerife, for each metric we obtain three different scores, each corresponding to a different test set, so we will show the mean and standard deviation of such scores. In majorca and tenerife we obtain a single test score corresponding to data from the year 2015. For the regression problems, in Tables 4.3 and 4.4, we show both the MAE and R2 score, closely related to MSE, obtained in the test set, and for classification, in Table 4.5, we show the F1 and the accuracy scores.

Results. The tables with the numerical results have three blocks, one for each variant L1, L2 or LS-SVMs, and, in each block, for each problem we show in bold the model with best test results. We also provide a statistical significance ranking based on the Wilcoxon test. The Wilcoxon test is a pairwise test that checks whether the difference of two samples has 0 median, i.e., the distribution is symmetric around 0. Instead of showing every Wilcoxon test result between all pairs of models, which would result in a 12×12 matrix difficult to interpret, we take the following approach. We first sort the

TABLE 4.3: Test MAE (top) and R2 score (bottom) and Wilcoxon-based ranking for the models selected using the MAE for hyperparametrization. The best models are shown in bold.

	maj.	ten.	boston	california	abalone	crime
				MAE		
ITL-L1	5.087 (6)	5.743 (3)	2.341±0.229 **(1)**	36883.582±418.435 (2)	1.481±0.051 (3)	0.078±0.001 (2)
CTL-L1	5.175 (7)	5.891 (5)	**2.192±0.244 (1)**	41754.337±270.908 (6)	1.482±0.050 (3)	0.078±0.001 (2)
cvxCMB-L1	**5.047** (5)	**5.340 (1)**	2.239±0.255 **(1)**	36880.238±420.417 **(1)**	1.470±0.052 (2)	0.077±0.002 (2)
cvxMTL-L1	5.050 (5)	5.535 (2)	2.206±0.292 **(1)**	**36711.383±343.333 (1)**	**1.454±0.048 (1)**	**0.074±0.002 (1)**
ITL-L2	4.952 (3)	**5.629** (3)	2.356±0.300 **(1)**	37374.618±433.511 (5)	1.498±0.054 (4)	0.079±0.002 (2)
CTL-L2	5.193 (7)	6.107 (8)	**2.083±0.136 (1)**	42335.612±163.773 (8)	1.503±0.047 (5)	0.080±0.002 (2)
cvxCMB-L2	4.869 (3)	5.963 (6)	2.089±0.128 **(1)**	37374.618±433.511 (4)	1.494±0.050 (4)	0.077±0.003 (2)
cvxMTL-L2	**4.854** (2)	5.784 (4)	2.089±0.134 **(1)**	**37202.603±419.166** (3)	**1.482±0.049** (3)	**0.077±0.002** (2)
ITL-LS	4.937 (3)	5.649 (3)	2.204±0.116 **(1)**	37348.347±441.240 (4)	1.496±0.051 (4)	0.079±0.002 (2)
CTL-LS	5.193 (7)	6.005 (7)	**2.072±0.143 (1)**	42259.492±146.825 (7)	1.502±0.052 (5)	0.079±0.002 (2)
cvxCMB-LS	4.977 (4)	**5.593** (3)	2.081±0.146 **(1)**	37339.179±430.288 (4)	1.486±0.049 (4)	0.079±0.002 (2)
cvxMTL-LS	**4.824 (1)**	5.754 (4)	2.077±0.152 **(1)**	**37231.043±420.992** (4)	**1.478±0.050** (3)	**0.076±0.002** (2)
				R2		
ITL-L1	0.845 (6)	0.901 (7)	0.821±0.041 (2)	0.699±0.009 (7)	0.543±0.022 (8)	0.732±0.021 (3)
CTL-L1	0.837 (9)	0.901 (6)	0.854±0.036 **(1)**	0.639±0.006 (10)	0.559±0.014 (6)	0.740±0.027 (3)
cvxCMB-L1	0.844 (6)	0.905 (4)	0.845±0.053 **(1)**	0.699±0.009 (6)	0.555±0.018 (7)	0.741±0.029 (3)
cvxMTL-L1	**0.846** (4)	**0.908** (2)	**0.858±0.057 (1)**	**0.703±0.007** (6)	**0.568±0.012** (5)	**0.760±0.024** (2)
ITL-L2	0.846 (5)	0.906 (3)	0.836±0.045 (2)	0.707±0.009 (5)	0.565±0.025 (6)	0.743±0.017 (3)
CTL-L2	0.840 (8)	0.901 (8)	**0.889±0.017 (1)**	0.645±0.005 (9)	0.574±0.013 (4)	0.744±0.028 (3)
cvxCMB-L2	0.850 (3)	0.900 (9)	0.885±0.013 **(1)**	0.707±0.009 (4)	0.571±0.018 (4)	0.755±0.024 (3)
cvxMTL-L2	**0.863** (2)	**0.908 (1)**	0.888±0.015 **(1)**	**0.709±0.008 (1)**	**0.580±0.014** (3)	**0.762±0.028 (1)**
ITL-LS	0.849 (3)	0.907 (3)	0.856±0.008 **(1)**	0.707±0.009 (3)	0.573±0.015 (4)	0.743±0.022 (3)
CTL-LS	0.838 (9)	0.904 (5)	**0.894±0.015 (1)**	0.646±0.005 (8)	0.576±0.016 (4)	0.746±0.032 (3)
cvxCMB-LS	0.843 (7)	0.907 (2)	0.886±0.024 **(1)**	0.707±0.009 (2)	0.581±0.012 (2)	0.746±0.021 (3)
cvxMTL-LS	**0.863 (1)**	**0.910 (1)**	0.890±0.016 **(1)**	**0.709±0.008** (2)	**0.581±0.015 (1)**	**0.763±0.028 (1)**

models, according to some criterion, and test the significance between one model and the next in the sorting order. Then we create a new significance ranking, the ranking increases only if this difference is significant. For example, if there are no significant differences between the first and second model, according the sorting order, they both obtain a ranking of 1.

In the regression tables, apart from the test scores, we provide a Wilcoxon-based ranking for each problem. To apply the Wilcoxon test, in each problem we sort the models (using all blocks) according to their mean score. Then, we test if the difference between each model and the next one in the ranking is significant. To do this, we take the list of errors committed by each model in the test set, that is, $e_1 = y - \hat{y_1}$ and $e_2 = y - \hat{y_2}$. Then, we use the Wilcoxon test to check whether $e_1 - e_2$ is centered around 0. If the null hypothesis is rejected at a 5% level, then we say that the difference is significant. The results for regression problems are shown in Table 4.3, where the best parameters are selected according to the MAE validation score, and in Table 4.4, where the MSE is used as the validation metric. Although we cannot pick a single overall winner, we can still draw some conclusions from the tables. Notice that the convex MTL approaches usually perform better, obtaining the best result in 11 out of 18 MAE blocks and 16 out of 18 R2 blocks of Table 4.3; in Table 4.4 it obtains 12 out of 18 MAE blocks and 15 out of 18 R2 blocks. Also, a convex MTL approach obtains the single best overall model in four problems, while ties for the first place in boston and, only in tenerife ends up as second, after the cvxCMB-L1 model.

The classification results, in Table 4.5 show a similar behavior. In this table, the ranking is not computed for each problem, but in general, computing the mean of scores across all problems. This mean score is used to rank the models, which is the second left-most column, and, using the Wilcoxon-based procedure we produce a statistical

TABLE 4.4: Test MAE (top) and R2 score (bottom) and Wilcoxon-based ranking for the models selected using the MSE for hyperparametrization. The best models are shown in bold.

	maj.	ten.	boston	california	abalone	crime
				MAE		
ITL-L1	5.087 (7)	5.743 (3)	2.437±0.281 (3)	36941.516±450.767 (1)	1.480±0.058 (3)	0.079±0.002 (3)
CTL-L1	5.175 (8)	5.891 (7)	2.315±0.192 (2)	41857.602±235.021 (6)	1.479±0.047 (3)	0.078±0.000 (2)
cvxCMB-L1	**4.920** (4)	5.743 (4)	2.315±0.192 (3)	**36941.476±450.711 (1)**	1.471±0.057 (2)	0.079±0.002 (2)
cvxMTL-L1	5.050 (6)	**5.535 (1)**	**2.244±0.150** (1)	36999.003±360.445 (2)	**1.455±0.046 (1)**	**0.074±0.001 (1)**
ITL-L2	4.924 (5)	5.752 (5)	2.437±0.324 (3)	37407.929±461.878 (5)	1.497±0.050 (5)	0.079±0.002 (2)
CTL-L2	5.193 (8)	6.107 (9)	2.096±0.112 (1)	42335.612±163.773 (7)	1.504±0.048 (6)	0.079±0.002 (2)
cvxCMB-L2	**4.813 (1)**	**5.623** (3)	2.116±0.131 (1)	37398.940±449.498 (5)	1.495±0.051 (5)	0.078±0.003 (2)
cvxMTL-L2	4.854 (4)	5.784 (6)	**2.082±0.130** (1)	**37356.599±390.629** (4)	1.481±0.041 (4)	**0.076±0.000** (2)
ITL-LS	4.937 (5)	5.649 (3)	2.326±0.231 (3)	37385.244±403.331 (4)	1.495±0.045 (5)	0.079±0.002 (2)
CTL-LS	5.193 (8)	6.005 (8)	**2.072±0.143 (1)**	42339.063±156.624 (7)	1.504±0.043 (6)	0.078±0.002 (2)
cvxCMB-LS	**4.820** (2)	5.578 (2)	2.136±0.106 (1)	37377.005±391.694 (4)	1.491±0.048 (5)	0.078±0.002 (2)
cvxMTL-LS	4.824 (3)	**5.754** (6)	2.090±0.090 (1)	**37232.918±397.866** (3)	1.478±0.042 (3)	**0.076±0.000** (2)
				R2		
ITL-L1	0.845 (6)	0.901 (9)	0.800±0.050 (3)	0.703±0.009 (8)	0.534±0.053 (10)	0.732±0.017 (4)
CTL-L1	0.837 (7)	0.901 (8)	0.860±0.026 (2)	0.642±0.006 (10)	0.564±0.011 (8)	0.748±0.017 (3)
cvxCMB-L1	**0.852** (4)	0.901 (10)	0.860±0.026 (3)	0.703±0.009 (7)	0.550±0.036 (9)	0.733±0.018 (3)
cvxMTL-L1	0.846 (5)	**0.908** (5)	**0.871±0.019** (1)	**0.705±0.008** (6)	**0.573±0.011** (7)	**0.764±0.019** (1)
ITL-L2	0.850 (6)	0.906 (6)	0.819±0.053 (3)	0.707±0.009 (4)	0.573±0.020 (6)	0.744±0.018 (3)
CTL-L2	0.840 (6)	0.901 (11)	0.886±0.014 (1)	0.645±0.005 (9)	0.574±0.013 (6)	0.747±0.025 (3)
cvxCMB-L2	0.857 (3)	**0.910 (1)**	0.883±0.016 (1)	0.707±0.009 (2)	0.574±0.021 (5)	0.751±0.029 (3)
cvxMTL-L2	**0.863** (2)	0.908 (4)	**0.887±0.015** (1)	**0.708±0.007** (2)	**0.581±0.011** (2)	**0.768±0.020 (1)**
ITL-LS	0.849 (4)	0.907 (5)	0.841±0.028 (3)	0.707±0.009 (5)	0.577±0.012 (4)	0.743±0.021 (3)
CTL-LS	0.838 (7)	0.904 (7)	**0.894±0.015 (1)**	0.645±0.005 (9)	0.575±0.012 (4)	0.754±0.022 (3)
cvxCMB-LS	0.856 (3)	0.909 (3)	0.877±0.009 (1)	0.707±0.009 (3)	0.580±0.013 (3)	0.750±0.024 (3)
cvxMTL-LS	**0.863 (1)**	**0.910** (2)	0.890±0.014 (1)	**0.710±0.008** (1)	**0.582±0.011 (1)**	**0.763±0.019** (2)

significance ranking shown in the last column. Now, the Wilcoxon tests are done with samples of size 8, the number of classification problems, so it is more difficult to find significant differences. In any case, the Wilcoxon test here uses a very small sample size and is given only for illustration purposes. The convex MTL approaches gets the best 18 F1 scores out of 24 and the 22 best accuracy scores out of 26. The best overall model is the cvxMTL-L2, while the other convex MTL models are tied with it in the significance ranking.

4.4   Convex Multi-Task Learning with Neural Networks

The convex MTL formulation is easily applicable and interpretable, so it has good properties for kernel methods, but also for a broader class of learning models. However, although kernel methods have some very useful properties, they present some important limitations. The main one is the computation effort necessary to solve the optimization problems related with these methods, which grows in a superquadratic way with the number of patterns. This make it unfeasible to use these models for very large datasets. Moreover, it is difficult to develop specific kernel methods for data where there is a spatial or temporal dependency, such as images. Neural networks, however, are models that are well suited for this kind of challenges Deep NNs have had a massive success in multiple applications. Moreover, they are very flexible models whose architecture can be adapted to fulfill different goals. In this section we show how to use our convex formulation for MTL with neural networks.

In Chapter 3 we have reviewed the taxonomy for MTL methods, and the approaches can be broadly grouped in three categories: feature-based, parameter-based and combination-based. We also give in Subsection 3.4.4 some of the most relevant

TABLE 4.5: Test F1 (top) and accuracy (bottom) scores, global and block-wise Wilcoxon-based rankings for classification problems. The best models in each block are shown in bold.

	comp_(G)	comp_(R)	comp_(G,R)	ad_(G)	ad_(R)	ad_(G,R)	landmine	binding	mean	rank	Wil.
F1											
ITL-L1	0.625	**0.639**	0.630	**0.659**	0.653	0.657	0.231	0.867	0.620	10	1
CTL-L1	0.623	0.638	0.638	0.657	0.650	0.653	0.255	0.901	0.627	7	1
cvxCMB-L1	0.616	0.638	0.638	0.658	0.650	0.653	**0.270**	0.901	**0.628**	6	1
cvxMTL-L1	**0.627**	0.636	**0.640**	**0.659**	**0.655**	**0.659**	0.242	**0.907**	**0.628**	5	1
ITL-L2	0.636	0.623	0.607	**0.668**	**0.666**	**0.668**	0.256	0.867	0.624	8	3
CTL-L2	**0.640**	0.647	**0.651**	0.665	0.661	0.659	**0.270**	0.903	0.637	2	2
cvxCMB-L2	0.629	0.640	0.645	0.666	0.662	0.661	**0.270**	0.903	0.634	3	2
cvxMTL-L2	0.634	**0.651**	0.650	**0.668**	**0.666**	**0.668**	0.263	**0.909**	**0.639**	1	1
ITL-LS	**0.631**	0.622	0.608	0.659	0.659	0.660	0.243	0.867	0.619	12	2
CTL-LS	0.628	**0.644**	0.649	0.650	0.653	0.647	0.230	0.853	0.619	11	2
cvxCMB-LS	0.630	0.635	0.642	0.657	0.658	0.654	0.238	0.873	0.623	9	2
cvxMTL-LS	0.630	0.641	0.648	**0.659**	**0.659**	0.659	**0.257**	**0.906**	**0.632**	4	1
Accuracy											
ITL-L1	0.750	0.749	0.746	0.852	0.851	**0.853**	**0.941**	0.790	0.817	11	3
CTL-L1	**0.757**	0.759	**0.763**	0.852	0.847	0.849	0.938	0.850	0.827	6	2
cvxCMB-L1	0.754	0.759	**0.763**	0.852	0.847	0.849	0.935	0.850	0.826	7	2
cvxMTL-L1	0.753	**0.760**	**0.763**	**0.853**	**0.852**	**0.853**	0.933	**0.861**	**0.829**	5	1
ITL-L2	0.754	0.762	0.751	**0.856**	**0.855**	**0.856**	**0.942**	0.791	0.821	8	2
CTL-L2	**0.762**	0.765	**0.767**	0.854	0.853	0.851	0.933	0.853	0.830	3	1
cvxCMB-L2	0.757	0.764	0.766	0.854	0.853	0.853	0.934	0.853	0.829	4	1
cvxMTL-L2	0.753	**0.766**	0.766	**0.856**	**0.855**	**0.856**	0.933	**0.864**	**0.831**	1	1
ITL-LS	0.754	0.761	0.750	**0.851**	0.850	**0.851**	0.943	0.791	0.819	9	2
CTL-LS	**0.757**	**0.764**	0.766	0.845	0.847	0.842	0.914	0.750	0.811	12	3
cvxCMB-LS	0.754	**0.764**	0.765	0.849	**0.850**	0.848	0.925	0.793	0.818	10	3
cvxMTL-LS	**0.757**	**0.764**	**0.767**	**0.851**	**0.850**	**0.851**	**0.944**	**0.858**	**0.830**	2	1
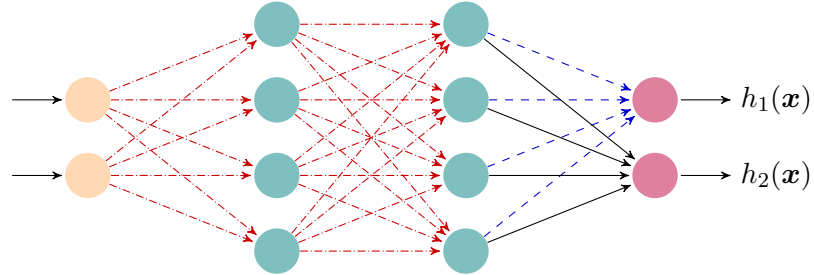
FIGURE 4.11: *Hard Sharing* Neural Network for two tasks and a two-dimensional input. Assuming a sample belonging to task 1 is used, the updated shared weights are represented in red, and in blue the updated specific weights. The input neurons are shown in yellow, the hidden ones in cyan and the output ones in magenta.

approaches to MTL with neural networks. Most of these approaches can fit in the feature-based category, where the shared layers are fully or partially shared to obtain a latent representation that is useful for all tasks, as shown in Figure 4.11; see for example Caruana (1997); Misra et al. (2016); Ruder et al. (2017). Some approaches rely also on a parameter-based view, where the parameters of each task-specific network are regularized together so that they are close in some sense, see Long and Wang (2015); Yang and Hospedales (2017b). However, to the best of our knowledge, we presented the first purely combination-based approach to MTL with neural networks in Ruiz et al. (2022), where we use a convex combination of neural networks.

4.4.1 Model Definition

Using the formulation of (4.1), we use neural networks to the model the common part

$$g(x_i^r; w, \Theta) = w^\intercal f(x_i^r; \Theta) + b,$$

and task-specific parts

$$g_r(x_i^r; w_r, \Theta_r) = w_r^\intercal f_r(x_i^r; \Theta_r) + d_r.$$

Here Θ and Θ_r are the sets of hidden weights, w, w_r are the output weights of the common and specific networks, respectively, and b and b_r the output biases. Observe that the feature transformations $f(x_i^r; \Theta)$ and $f_r(x_i^r; \Theta_r)$ are not fixed like the transformations $\phi(x_i^r)$ and $\phi_r(x_i^r)$ in the kernel methods; instead, here, they are automatically learned in the training process. The full MTL models are then

$$h_r(x_i^r) = \lambda\{w^\intercal f(x_i^r; \Theta) + b\} + (1 - \lambda)\{w_r^\intercal f_r(x_i^r; \Theta_r) + d_r\}. \tag{4.68}$$

This formulation offers multiple combinations since we can model each common or independent function using different architectures for $f(\cdot; \Theta)$ or $f_r(\cdot; \Theta_r)$. For example, we can use a network with a larger number of parameters for the common part, since it will be fed with more data, and simpler networks for the task-specific parts. Even different types of neural networks, such as fully connected and convolutional, can be combined depending on the characteristics of each task. This combination of neural networks can also be interpreted as an implementation of the LUPI paradigm (Vapnik and Izmailov, 2015) presented in Subsection 3.2.4, i.e., the common network captures the privileged information for each of the tasks, since it can learn from more sources.

4.4.2 Training Procedure

The regularized risk corresponding to the convex MTL neural networks is

$$R_{\boldsymbol{z}} = \sum_{r=1}^{T} \sum_{i=1}^{m} \ell(h_r(x_i^r), y_i^r) + \frac{\mu}{2} \left(\|w\|^2 + \sum_{r=1}^{T} \|w_r\|^2 + \Omega(\Theta) + \Omega(\Theta_r) \right). \tag{4.69}$$

Here, h_r is defined as in equation (4.68), and $\Omega(\Theta)$ and $\Omega(\Theta_r)$ represents the L_2 regularization of the set of hidden weights of the common and specific networks, respectively. Given a loss function $\ell(\hat{y}, y)$ and a pair (x_i^t, y_i^t) from task t, we use the chain rule to compute the gradient of the loss function with respect to some parameters $\mathcal{P}$:

$$\nabla_{\mathcal{P}} \ell(h_t(x_i^t), y_i^t) = \frac{\partial}{\partial \hat{y}_i^t} \ell(\hat{y}_i^t, y_i^t)|_{\hat{y}_i^t = h_t(x_i^t)} \nabla_{\mathcal{P}} h_t(x_i^t). \tag{4.70}$$

Recall that we are using the formulation

$$h_t(x_i^t) = \lambda\{w^\intercal f(x_i^t; \Theta) + b\} + (1 - \lambda)\{w_t^\intercal f_t(x_i^t; \Theta_t) + b_t\},$$

Input: $X_{\text{mb}}, t_{\text{mb}}$ // Minibatch data and task labels
Output: f // Forward pass for the minibatch
Data: λ // Parameter of convex combination
Data: $g; g_1, \ldots, g_T$ // Modules of the common and specific networks
for $x_i, t_i \in (X_{mb}, t_{mb})$ **do**
   | $f_i \leftarrow \lambda g(x_i) + (1 - \lambda) g_{t_i}(x_i)$ // Convex combination
end

Algorithm 1: Forward pass for Convex MTL neural network.

where we make a distinction between output weights w, w_t and hidden parameters Θ, Θ_t. Then, the corresponding gradients of h_t needed to compute the loss gradients are

$$
\begin{aligned}
\nabla_w h_t(x_i^t) &= \lambda\{f(x_i^t, \Theta)\}, & \nabla_\Theta h_t(x_i^t) &= \lambda\{w^{\intercal} \nabla_\Theta f(x_i^t, \Theta)\}; \\
\nabla_{w_t} h_t(x_i^t) &= (1 - \lambda)\{f_t(x_i^t, \Theta)\}, & \nabla_{\Theta_t} h_t(x_i^t) &= (1 - \lambda)\{w^{\intercal} \nabla_{\Theta_t} f_t(x_i^t, \Theta_t)\}; \\
\nabla_{w_r} h_t(x_i^t) &= 0, & \nabla_{\Theta_r} h_t(x_i^t) &= 0, \text{ for } r \neq t.
\end{aligned}
\tag{4.71}
$$

Putting all together, the gradient of the loss with respect to w, for example, is

$$
\nabla_w \ell(h_t(x_i^t), y_i^t) = \ell(\hat{y}_i^t, y_i^t)|_{\hat{y}_i^t = h_t(x_i^t)} \lambda\{f(x_i^t, \Theta)\}
$$

and the same for the rest of parameters. Observe that the convex combination information is transferred in the back-propagation as the loss gradients with respect to the common parameters are scaled by λ, while those of the task-specific parameters are scaled by $(1 - \lambda)$. Moreover, the regularization of each set of parameters, i.e., $\{w\}, \Theta$ and $\{w_r\}, \Theta_r$, is done independently, so their gradients can be computed in the standard way. During the back propagation procedure, we only update the parameters that have been used in the forward pass, with possibly different learning rates for each network. That is, given an example (x_i^t, y_i^t), when using vanilla Stochastic Gradient Descent (SGD) the update rules for the common network parameters would be

$$
\begin{aligned}
w^{\tau+1} &\leftarrow w^\tau + \eta \left[\frac{\partial}{\partial \hat{y}_i^t} \ell(\hat{y}_i^t, y_i^t)|_{\hat{y}_i^t = h_t(x_i^t)} \lambda\{f(x_i^t, \Theta)\} + \mu w^\tau \right], \\
\Theta^{\tau+1} &\leftarrow \Theta^\tau + \eta \left[\frac{\partial}{\partial \hat{y}_i^t} \ell(\hat{y}_i^t, y_i^t)|_{\hat{y}_i^t = h_t(x_i^t)} \lambda\{w^{\intercal} \nabla_\Theta f(x_i^t, \Theta)\} + \mu\{\nabla_\Theta \Omega(\Theta)\} \right],
\end{aligned}
\tag{4.72}
$$

while the update rules for t-th task network parameters would be

$$
\begin{aligned}
w_t^{\tau+1} &\leftarrow w_t^\tau + \eta_t \left[\frac{\partial}{\partial \hat{y}_i^t} \ell(\hat{y}_i^t, y_i^t)|_{\hat{y}_i^t = h_t(x_i^t)} (1 - \lambda)\{f(x_i^t, \Theta)\} + \mu w_t^\tau \right], \\
\Theta_t^{\tau+1} &\leftarrow \Theta_t^\tau + \eta_t \left[\frac{\partial}{\partial \hat{y}_i^t} \ell(\hat{y}_i^t, y_i^t)|_{\hat{y}_i^t = h_t(x_i^t)} (1 - \lambda)\{w^{\intercal} \nabla_{\Theta_t} f(x_i^t, \Theta_t)\} + \mu\{\nabla_{\Theta_t} \Omega(\Theta_t)\} \right],
\end{aligned}
\tag{4.73}
$$

and the parameters from the rest of task-specific network are not updated. That is, no specific algorithm has to be developed for training the convex MTL NN. In (4.72) and (4.73) we have shown the update rules for vanilla SGD, but any other algorithm, e.g., Adam, can be used scaling properly the loss gradients.
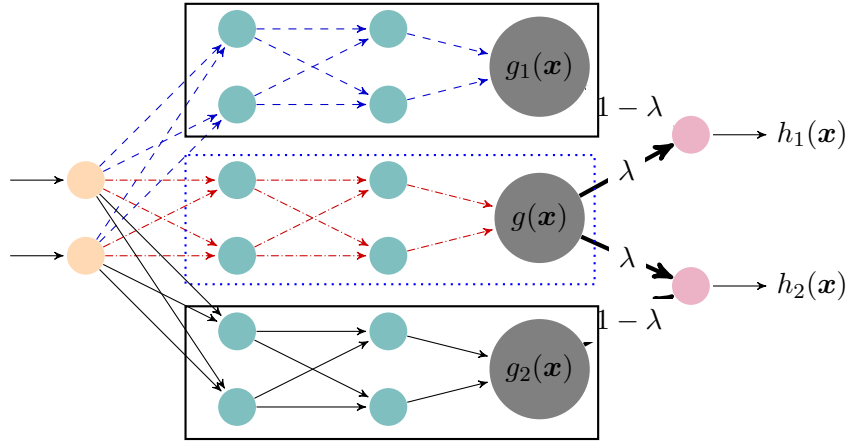
FIGURE 4.12: Convex MTL neural network for two tasks and a two-dimensional input. Assuming a sample belonging to task 1 is used, the updated shared weights are represented in red, and in blue the updated specific weights. Specific networks are framed in black boxes and the common one in a blue box. The input neurons are shown in yellow, the hidden ones in cyan (except those in grey), and the output ones in magenta. We use the grey color for hidden neurons containing the intermediate functions that will be combined for the final output: $g_1(\boldsymbol{x})$, $g_2(\boldsymbol{x})$ and $g(\boldsymbol{x})$. The thick lines are the hyperparameters λ and $1 - \lambda$ of the convex combination.

4.4.3 Implementation Details

Our implementation of the convex MTL neural network is based on `PyTorch` (Paszke et al., 2019). Although we include the gradients expressions in equation (4.71), the `PyTorch` package implements automatic differentiation, so the gradients are not explicitly implemented. Instead, we implement each network, common or specific, using (possibly different) `PyTorch` modules. In the forward pass of the network, the output for an example x_i^r from task r is computed using a pass of the common module and the corresponding specific module, and combining both passes with the convex formulation to obtain the final output $h_r(x_i^r)$. In the training phase, in which minibatches are used, the full minibatch is passed through the common model, but the minibatch is task-partitioned, where each partition is passed through its corresponding specific module. By doing this, when using examples from the r-th task only the parameters corresponding to the common module and its corresponding specific one are updated. Moreover, as mentioned above, with the adequate forward pass, the `PyTorch` package automatically computes the scaled gradients in the training phase. Our implementation is a general framework to combine in a convex manner any `Pytorch` modules and select which ones should be used and updated in each training iteration, as well as using the corresponding modules for the prediction stage. TODO: ponerlo en github

In Algorithm 1 we show the pseudo-code of the forward pass of the convex MTL neural network. Here, g and $g_1, \ldots, g_T$ are the common and task-specific modules, whose outputs are combined. We do not show the backward pass because we rely on PyTorch automatic differentiation.

4.4.4 Experiments

Problem Description.
To test the Convex MTL neural networks we use four Multi-Task image datasets: var-MNIST, rot-MNIST, based on MNIST (LeCun et al., 1998), and var-FMNIST, rot-FMNIST,
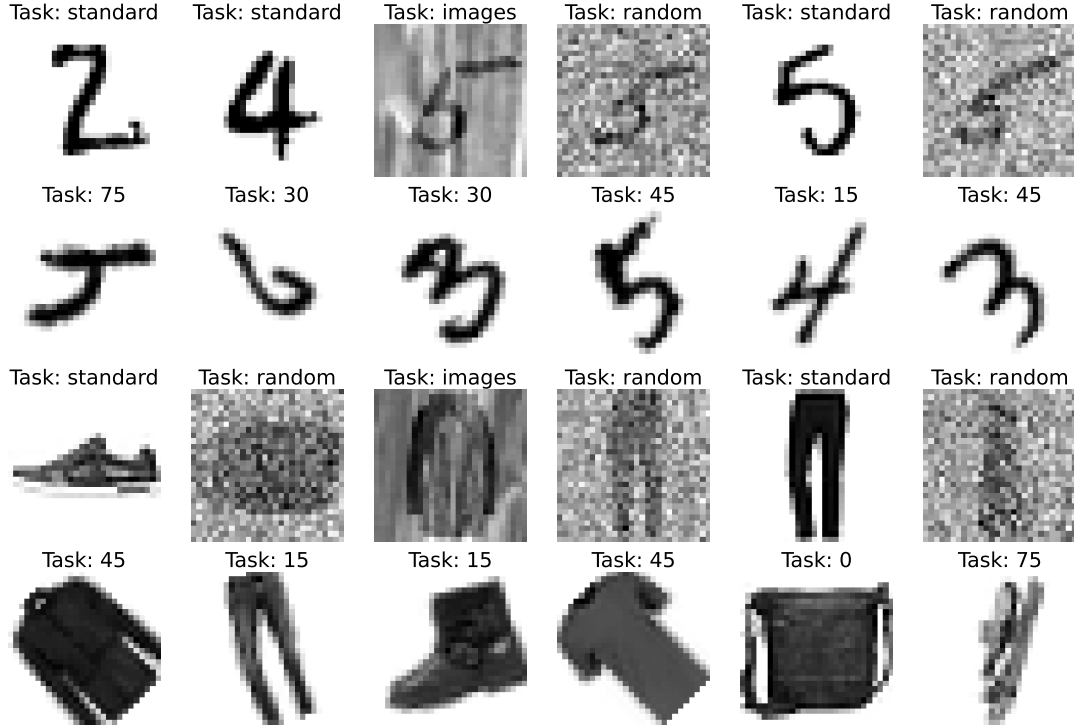
FIGURE 4.13: Images of the four classification problems used. Each image has a title indicating the corresponding task. The rows correspond to var-MNIST, rot-MNIST, var-FMNIST and rot-FMNIST (from top to bottom).

based on fashion-MNIST (Xiao et al., 2017). The MNIST and fashion-MNIST datasets are both composed of 70 000 examples of 28×28 grey-scale images. The MNIST dataset contains images of handwritten numbers, while the fashion-MNIST has images of clothes and fashion-related objects. Both are used as classification problems, where the images have to be classified in one of 10 possible classes. These classes are balanced in both datasets. To generate the Multi-Task image datasets that we use, we take the images from MNIST or fashion-MNIST and use either the *variations* procedure or the *rotation* one. For the var-MNIST and var-FMNIST we use the *variations* procedure. Inspired by the work of Bergstra and Bengio (2012), we consider three transformations:

- *random*: adding random noise to the original image.

- *image*: adding a random patch of another image to the original image.

- *standard*: no transformations are applied to the original image.

Then, we use a random split to divide the original datasets in three groups. To each group we apply one of the transformations defined, so we get three tasks: two with 23 333 examples and the third one with 23 334.

For the rot-MNIST and rot-FMNIST we use the *rotations* procedure. Using the definitions of Ghifary et al. (2015), we consider six transformations:

- *0*: rotating $0°$ the original image.

- *15*: rotating $15°$ the original image.

- *30*: rotating $30°$ the original image.

- *45*: rotating 45° the original image.

- *60*: rotating 60° the original image.

- *75*: rotating 75° the original image.

Again, we use a random split to divide the original datasets in six groups and each group is applied one of the transformations defined, so we get six tasks: four with $11\,667$ examples and two with $11\,666$.

In Figure 4.13 we show examples of the four MTL image problems that we generate, with the corresponding task annotation for each one.

Experimental Procedure.
For testing the performance of our proposal we consider the following models:

- ctlNN: a CTL-based neural network, that is, a single network for all tasks.

- itlNN: an ITL-based neural network, that is, an independent network for each task.

- hsNN: a MTL-based neural network using the hard sharing strategy. That is, a single neural network is used for all tasks, where the first layers are shared among tasks, but a task-specific output layer is used for each task.

- cvxmtlNN: a MTL-based neural network using the convex formulation we propose.

All of these models are based on a convolutional network, which we will name convNet, whose architecture is taken from the Spatial Transformer Network (Jaderberg et al., 2015) implementation given in Pytorch[1]. The architecture of convNet consists, in this order, on two convolutional layers of kernel size 5, with 10 and 20 output channels each; then a dropout layer, followed by a max pooling layer, and two fully connected hidden layers with 320 and 50 neurons. After this, the output layers follow.

Since we use one-hot encoding for the multiple classes, the architecture of the ctlNN consists on a single network with the convNet architecture with 10 output neurons, one for each class. In the itlNN, an independent network, with the convNet architecture and 10 output neurons, is used for each task. For the hsNN, a single network with the convNet architecture is used, but we have a group of 10 output neurons for each task in the problem. For example, if we are using of the *rotation*-based problems, we would have 60 output neurons, but only the group of 10 corresponding to each task is used with each example. In the cvxmtlNN we use a network with the convNet architecture and 10 output neurons to model the common network and each of the task-specific networks. That is, given an example from task r, the 10 common output neurons and the r-th task-specific ones are combined to obtain the final output.

We use the AdamW algorithm (Loshchilov and Hutter, 2019) to train all the models considered, and the weight decay parameter μ for each model is selected using a CV-based search over the values $\{10^{-4}, 10^{-3}, 10^{-2}, 10^{-1}, 10^{0}\}$. The rest of the parameters, which are part of the architecture, are fixed and set to the default values: the dropout rate is 0.5 and we use a 2×2 max pooling layer with a stride of 2. The cvxmtlNN model also has λ as a hyperparameter, and it is selected, alongside μ, using a CV grid search, where the grid for λ is $\{0, 0.2, 0.4, 0.6, 0.8, 1\}$.

The train and test sets are generated using a task-stratified split of 70% and 30%, respectively. The CV grid searches are carried out using the training set, where we

---

[1] www.pytorch.org/tutorials/intermediate/spatial_transformer_tutorial.html

TABLE 4.6: Test Accuracy with Majority Voting.

	var-MNIST	rot-MNIST	var-FMNIST	rot-FMNIST
ctlNN	0.964	0.973	0.784	0.834
itlNN	0.968	0.981	0.795	0.873
hsNN	0.971	0.980	0.770	0.852
cvxmtlNN	**0.974** $(\lambda^* = 0.6)$	**0.984** $(\lambda^* = 0.8)$	**0.812** $(\lambda^* = 0.6)$	**0.880** $(\lambda^* = 0.6)$

TABLE 4.7: Test Mean Categorical Cross Entropy.

	var-MNIST	rot-MNIST	var-FMNIST	rot-FMNIST
ctlNN	1.274 ± 0.143	1.145 ± 0.039	2.369 ± 0.183	1.757 ± 0.075
itlNN	1.072 ± 0.029	0.873 ± 0.058	2.356 ± 0.130	1.598 ± 0.042
hsNN	1.087 ± 0.253	0.898 ± 0.073	3.067 ± 0.888	1.888 ± 0.075
cvxmtlNN	$\mathbf{0.924 \pm 0.024}$ $(\lambda^* = 0.6)$	$\mathbf{0.831 \pm 0.029}$ $(\lambda^* = 0.8)$	$\mathbf{2.147 \pm 0.090}$ $(\lambda^* = 0.6)$	$\mathbf{1.482 \pm 0.063}$ $(\lambda^* = 0.6)$

use a 5-fold CV scheme. These folds are task-stratified, that is, all have the same task proportions. Also, since the problems are class-balanced and the sample size is reasonably large, the folds are expected to be class-balanced as well.

Results.
To get more accurate results, less sensitive to randomness, we train each model 5 different times. To do this, once the optimal hyperparameters have been selected using the CV in the training set, we refit the model with these hyperparameters using the entire training set. That is, CV is done only once for each model in each problem, but then we repeat 5 times the procedure of training the network over the whole training set.

Although maximizing the accuracy is ultimately the goal in classification problems, it is not a differentiable measure, so we use the categorical cross entropy instead as the loss function to train the networks. Both measures are broadly correlated, but they do not represent exactly the same behavior. We will show the results using both for completeness.

Since we have 5 instances of each model, a typical strategy to combine their predictions is majority voting. We perform this majority voting directly using the logits, in the output neurons, of each instance and averaging them, so we have 10 values, one for each class. In Table 4.6 we show the test accuracy, using the logits average already described, for each of the approaches considered. Other approach to visualize these results is to compute the categorical cross entropy directly on the averaged logits, which we show in Table 4.7. In both tables we also show the optimal values for λ selected in the CV.

Our proposal, the cvxmtlNN model, obtains the best results in all the problems, either in terms of accuracy or cross entropy. The ITL approach comes second in all problems, except for the accuracy score in var-MNIST, where the hsNN is second and itlNN goes third. The hsNN model goes third in the rest of problems, while the ctlNN gets the worst results consistently in all problems, sometimes by a large margin. By looking at the tables, it seems that a CTL approach is not able to capture the different properties of each task simultaneously. On the other hand, the ITL approach obtains good results, because it is specialized in each task. Looking at the optimal values for λ, there is, however, common information shared among the tasks. These values fall far from the $0, 1$

margins, so neither the CTL nor the ITL are optimal approaches. This is reflected in the Tables 4.6 and 4.7, where the cvxMTLNN outperforms consistently ctlNN and itlNN. We can assume, then, that the information learned by the common network and the task-specific ones is complementary, because their combination leads to better results. The hard sharing approach, although better than a CTL one, seems to be too rigid to effectively capture the differences among different tasks, so it is frequently surpassed by the ITL network.

4.5 Application to Renewable Energy Prediction

A transition towards renewable energies is taking place, with particular interest for solar and wind generation, which implies a demand for accurate energy production forecasts to be made for the transmission system operators, wind and solar farm managers and market agents. These forecasts can be made at different time horizons: very short (up to one hour), short (up to a few hours), or medium-long (one or more days ahead). In this application of our convex MTL techniques to renewable energies forecast, we will focus wit the latter, in particular, the hourly, day-ahead prediction, that is, the prediction of tomorrow, at each hour, is predicted today.

Machine Learning (ML), like in other forecasting problems, has an increasing presence in energy prediction approaches. The usage of ML models requires choosing the predictive features that will be used, which depends on the time horizon of interest. For short-term forecast, past values of energy production and real time meteorological data can be used; however, for longer horizons, the most common features are numerical weather predictions (NWP), that can be provided by entities such as the European Centre for Medium-Range Weather Forecasts, which is the one used in this work. For the hourly day-ahead predictions of interest here, we use the NWP forecasts of the ECMWF run at 00 UTC in a given day to predict the hourly energy productions the day after. That is, using the NWP of 00 UTC, the energy generation predictions are given for each hour from 24 to 47h.

After the selection of predictive features, the ML method better suited for the problem at hand has to be selected. Also, each method has a set of hyperparameters that influence its behaviour and have to be adjusted in each occasion. In the case of interest here, there are two possibilities: using local models for single installations or global models for multiple installations within a geographic area.

Anyway, any ML approach has to deal with the changing behaviour of a wind or solar farm, which can be altered substantially according to different conditions or time. In the PV energy production this is obvious, since different times of the day, from sunrise to sunset, have very different behaviours; but there are also seasonal effects that affect the energy generation in the solar farms. For wind energy, it is more difficult to define the variables that determine different scenarios. The wind velocity forecast is the most relevant variable for energy generation, but it is important to take into account the power curve of wind turbines, which has three different response zones: one for low speed and near zero production, an intermediate one with power growing with wind velocity and one with maximum constant power up to the cut-off speed. The angle in which this wind incides is also important, since the turbines of a farm are set for a specific direction. Finally, the assymetrical wind velocities between the day and night period can also affect the energy production.

One way to deal with these different behaviour scenarios is to apply a MTL model, where the models built are specialized in each scenario but all scenarios are used in the

TABLE 4.8: Hyperparameters, grids used to find them (when appropriate), and hyperparameter selection method for each model. Here, d is the number of dimensions of the data and σ is the standard deviation of the target.

Par.	Grid	ctlSVR	itlSVR	cvxMTL
C	$\left\{10^k : -1 \leq k \leq 6\right\}$	CV	CV	CV
ϵ	$\left\{\frac{\sigma}{2^k} : 1 \leq k \leq 6\right\}$	CV	CV	CV
γ	$\left\{\frac{4^k}{d} : -2 \leq k \leq 3\right\}$	CV	-	ctlSVR
γ_r	$\left\{\frac{4^k}{d} : -2 \leq k \leq 3\right\}$	-	CV	itlSVR
λ	$\left\{10^{-1}k : 0 \leq k \leq 10\right\}$	-	-	CV

learning process. In this Section a convex MTL kernel-based approach will be used for wind and solar energy forecasting. To do this, it is necessary to define the tasks of interest on each case, which are described in the following subsections. In the first subsection the experimental methodology is described, showing how the models are chosen and the hyperparameters are selected. Then, the next two subsection presents the approach and the detailed task definitions used for solar and wind energy, respectively, as well as the results to measure the resultant performance.

4.5.1 Experimental Methodology.

Here we describe the methodology that we have followed to conduct the experiments of renewable energy prediction. For the solar and wind energy we use the same procedure. In each problem, we have a train, validation and test sets, each corresponding to one year of data. In the solar energy problems, where we have data from photovoltaic parks in Mallorca and Tenerife, and we use 2013, 2014 and 2015 as train, validation and test sets, respectively; while for wind energy problems, where we have data from the Sotavento park, in Spain, we use the years 2016, 2017 and 2018. For both solar and wind problems we use different definition of tasks. For the tasks definition we use only the training data for establishing rules to partition the data in different tasks; then, with these tasks' definition, we apply them to get the tasks of the validation and test examples. For example, if we use the wind velocity to define three tasks, we study the velocities of the training examples to set the boundaries that define each task: low, medium and high velocity; then, we use these definitions on the validation and test sets. We will represent the task definition applied with the nomenclature: (taskDef)_modelName, where taskDef is a name for the task definition and modelName is the name of the model.

We consider three different models based on the standard Gaussian kernel SVR:

- ctlSVR: a CTL model, that is, a single SVR for all tasks. Its set of hyperparameters is $\{C, \gamma, \epsilon\}$, where C is the regularization trade-off parameter, γ is the kernel width, and ϵ the width of the error insensitive area.

- itlSVR: an ITL model, that is, an independent SVR for each task. For each task, we have standard SVR with its hyperparameters: $\{C_r, \gamma_r, \epsilon_r\}$ for $r = 1, \ldots, T$.

- mtlSVR: a convex MTL model, as shown in Subsection 4.2.1, with its corresponding set of hyperparameters is $\{C, \epsilon, \gamma, \gamma_1, \ldots, \gamma_T, \lambda_1, \ldots, \lambda_T\}$, where γ is the kernel

width of the common part and γ_r the one of the r-th specific part; also, λ_r is the convex combination parameter corresponding to the r-th task.

Although the CTL approach does not use the tasks information, the ITL and MTL models depend on the task definition that we use. For example, prediction at different hours can define different tasks, where the possible values are, for example, (hour=14) or (hour=12). The models using this task definition will be named (hour)_itlSVR and (hour)_mtlSVR. Since each task definition partitions the data, we can also use multiple task definitions, combining them and creating finer partitions. We will name (taskDef1,...,taskDefM) the combination of task definitions taskDef1, ..., taskDefM. For example, consider the (hour) definition for solar energy, in which we consider 14 different hours, hence, 14 tasks; and consider the season definition, which considers the prediction in each season as a different task, hence, four tasks. The combined definition (hour, season) generates 14×4 possible tasks, whose values can be, for example, (hour = 12, season = summer). The models using this combination will be named (hour, season)_itlSVR or (hour, season)_mtlSVR.

As explained in Subsection 4.2.1, the cost of methods to select the optimal hyperparameters scales exponentially with the dimension, so it is not feasible to use a CV grid search, for example, if we have more than 3 hyperparameters. In the ctlSVR and itlSVR it is not a problem, since we have 3 hyperparameters, for the common, single SVR, and for the task-independent ones, respectively. We use then a CV grid search, using as train and validation sets described above. However, to find the hyperparameters of mtlSVR we have to make some adjustments, as discussed in Subsection 4.3.3. First, we use a convex MTL formulation with a single λ parameter, common to all tasks. Second, we use the optimal kernel widths selected in validation for the CTL and ITL approaches as the widths in the MTL approach. That is, we get the optimal common γ^* and task-specific $\gamma_1^*, \ldots, \gamma_T^*$, and fix them in the mtlSVR model, not including them in the grid search procedure. Then, we use a CV grid search to find the optimal values of the remaining hyperparameters, that is, $\{C, \epsilon, \lambda\}$. In Table 4.8 we show the method to obtain each hyperparameter, as well as the grids used in the CV grid search procedures. We use the MAE as the validation metric, because is the most natural to the ϵ-insensitive loss that is used in the SVRs, and the most commonly used in energy forecasting.

The whole procedure to get the final scores is:

1. **Scale the target and normalizing the features.** We scale the target values to $[0, 1]$ and we normalize each feature, so it has 0 mean and a standard deviation of 1. This is done using the training data only. For the target scale, we select the target minimum and maximum values of the training set, that is, $y_{\min} = 0$, when no energy is produced, and $y_{\max}$ is the maximum capacity of the park.

2. **Use a CV grid search to select the optimal hyperparameters.** This is done using the train and validation sets, with the grids and adjustements already explained. We use the MAE as our validation metric.

3. **Predict on the test set and rescale to the original scale.** That is, we use the corresponding model $f(\cdot)$ to compute the prediction of the normalized i-th test example from task r, $\tilde{x}_i^r$, as $f(\tilde{x}_i^r)$. Then, we rescale it back to obtain the final prediction $\hat{y}_i^r = f(x_i^r) \times (y_{\max} - y_{\min}) + y_{\min}$

4. **Compute the test score.** Using the target values y_i^r and their corresponding predictions, $\hat{y}_i^r$, we measure the performance of our model using both MAE and MSE.

The entire process is carried out using a `Pipeline` object, where we make use of class `StandardScaler` to normalize the data, and `TransformedTargetRegressor` class to scale the targets. All these classes are part of the *scikit-learn* library.

To put our results in perspective, we also show the errors of simple persistence models and of multilayer perceptrons. For the perceptron we use the `MLPRegressor` class of *scikit-learn*. The architecture for both problems consists on fully connected networks with two hidden layers, with 100 and 50 neurons. We train these networks using the L-BFGS solver with a maximum of 800 iterations and a tolerance of 10^{-10}. The regularization hyperparameter is selected using a CV grid search, as those described above, where the grid is $\left\{ 4^k : -2 \leq j \leq 3 \right\}$.

4.5.2 Solar Energy

The goal is to predict the hourly energy production in the islands of Mallorca and Tenerife, and we name the corresponding problems as majorca and tenerife, respectively. In this subsection the experiments with these solar problems are presented, using the experimental procedure already described. First a description of the problems and the data used is given; then the results are given and analyzed.

Data and Tasks.
For both problems, majorca and tenerife, the same variables, extracted from the Numerical Weather Prediction (NWP), are used:

- Surface net solar radiation (SSR).

- Surface solar radiation downwards (SSRD).

- Total Cloud Cover (TCC).

- Temperature at 2 meters above surface (T2M).

- Module of the speed of wind at 10 meters above surface (v10).

The radiation variables SSR, solar radiation, and SSRD, the diffuse radiation scattered by the atmosphere plus solar radiation, as well as the TCC, have all a direct impact on PV production. The T2M and v10 features are also considered because they influence the conversion of photon energy into electrical one, and also the overall performance of PV stations.

To collect these features, geographical grids with a $0.125°$ spatial resolution are considered. For majorca, the grid has its northeast coordinates at $(2°, 40°)$, and its southwest coordinates at $(4°, 39°)$. For tenerife, the coordinates are $(-17.5°, 28.75°)$ for the northwest corner, and $(-15.5°, 27.75°)$ for the southeast one. That is, both grids have a longitude width of 2 degrees and a latitude height of 1 degree. With the spatial resolution considered, this results in a total number of $17 \times 9 = 153$ grid points; since we use five variables at every point, the total dimension of our data is thus $5 \times 153 = 765$.

Observe that we obtain large dimensional patterns, where the features might be highly correlated. That is, a feature, SSR for example, measured in one point of the grid and another close point might be very correlated, since the grids are squares with sides of about $12\,\text{km}$. This correlation will affect those models based on matrix-vector computations, such as linear models, which are the most obvious, but, also, to some extent, neural networks. Although Ridge or Tikhonov regularization can alleviate this issue for these models, the kernel-methods, such as the kernel SVMs, seems better suited for these

TABLE 4.9: Test MAEs (left), test MSEs (center), with the corresponding rankings, and optimal mixing λ^* (right) of the solar energy models considered in `majorca`. Base units are either MWh or percentages (%). The best model errors are shown in bold.

	MAE			**MSE**			$\boldsymbol{\lambda^*}$
	MWh	%	rank	MWh	‰₀₀	Rank	
ctlSVR	5.265	7.265	(6)	59.322	112.985	(6)	-
(season)_itlSVR	5.305	7.384	(7)	59.591	113.498	(7)	-
(season)_mtlSVR	**4.884**	**6.740**	**(1)**	53.222	101.366	(2)	0.4
(hour)_itlSVR	5.083	7.015	(4)	54.540	103.877	(3)	-
(hour)_mtlSVR	4.957	6.840	(2)	**52.614**	**100.208**	**(1)**	0.3
(hour, season)_itlSVR	5.250	7.251	(5)	57.927	110.328	(5)	-
(hour, season)_mtlSVR	5.038	6.952	(3)	54.601	103.992	(4)	0.3

kind of problems. When we use kernels, such as the Gaussian kernel $\exp -\gamma \|x - y\|^2$, the algorithm learns using the distances among patterns $\|x - y\|$, instead of its features. These distances scale linearly with the dimension of data, that is, consider the features scaled to $[0, 1]$; then a rough estimate of the distance between two patterns would be d. In the extreme case where all features are equal, i.e. $x_j = x_1$ for all $j = 1, \ldots, d$; then, $\|x - x'\| = d |x_1 - x'_1|$. However, this influence of the dimension can be easily controlled by γ, and, if selected properly, should not affect the performance of a Gaussian SVR.

Recall that we use data from years 2013, 2014 and 2015 as train, validation and test sets, respectively. We show the errors in both total MWh and as percentages, in the range $[0, 100]$ of the total install PV power in each island, 72.46 MW in `majorca` and 107.68 MW in `tenerife`. We remove night data for obvious reasons and make predictions between 06 UTC and 19 UTC for `majorca` and between 07 UTC and 20 UTC for `tenerife`. The hour of the day has a direct influence on the solar radiation, and, therefore, on energy production. Also, the season of the year has a similar impact on the production. This leads to two obvious task definitions:

- `hour`: The prediction at each hour is defined as a different task; there are thus 14 tasks in `majorca` (from 06 to 19 UTC) and `tenerife` (from 07 to 20 UTC).

- `season`: The prediction at each season is defined as different task. With a slight abuse of language, the following definitions for each season are used: Spring, from 16 February to 15 May; Summer, from 16 May to 15 August; Autumn, from 16 August to 15 November; and Winter, from 16 November to 15 February.

In Subfigures 4.14a and 4.14b the hourly averages of the PV energy in MWh are shown for `majorca` and `tenerife`. Also, in Subfigures 4.14c and 4.14d we show the monthly averages, colored by season.

Experimental Results.
In Tables 4.9 and 4.10 we show the numerical results for `majorca` and `tenerife`, respectively. We give the test MAE, which is the most natural metric for SVRs, as well as the MSE. In the case of percentages, for MAE, we give the percentage corresponding to the total installed power, and for the MSE, the permyriad, that is per 10 000, of the installed power. We also show the rankings in terms of MAE or MSE, and the optimal hyperparameter λ^* selected by the CV for the MTL models.
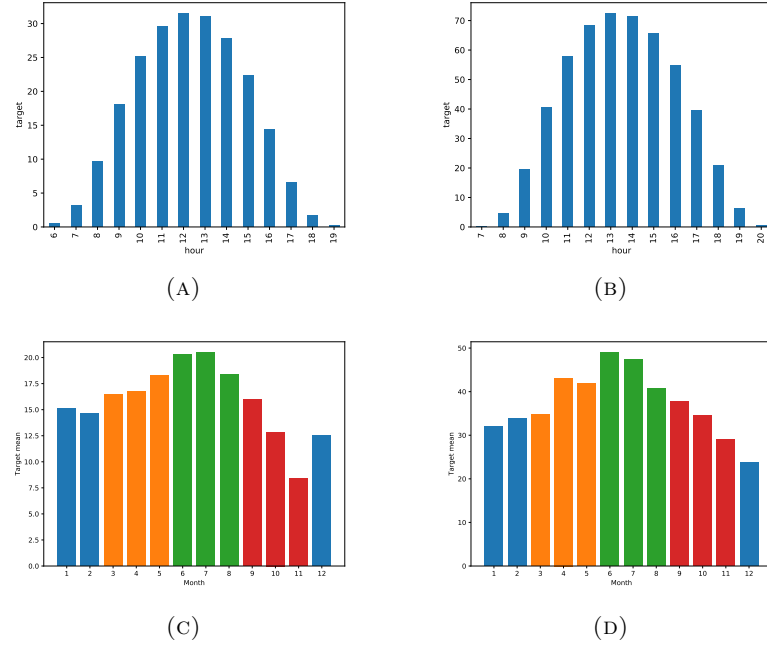
FIGURE 4.14: Hourly photovoltaic energy mean in `majorca`(a) and `tenerife`(b) measured in MWh. Photovoltaic energy monthly averages for `majorca`(c) and `tenerife`(d) , colored using the tasks defined using the season and measured in MWh. All the histograms have been computed using data from year 2016.

TABLE 4.10: Test MAEs (left), test MSEs (center), with the corresponding rankings, and optimal mixing λ^* (right) of the solar energy models considered in `tenerife`. Base units are either MWh or percentages (%). The best model errors are shown in bold. The positions in bold correspond to the model ranked first in terms of MAE or MSE, as indicated by its column.

	MAE			**MSE**			$\boldsymbol{\lambda^*}$
	MWh	%	Rank	MWh	‰₀₀	Rank	
ctlSVR	5.786	5.373	(5)	88.323	76.174	(5)	-
(season)_itlSVR	5.930	5.545	(6)	97.454	84.611	(6)	-
(season)_mtlSVR	5.579	5.181	(4)	86.227	74.366	(3)	0.8
(hour)_itlSVR	5.403	5.018	(2)	86.686	74.762	(4)	-
(hour)_mtlSVR	**5.376**	**4.993**	**(1)**	**84.207**	**72.624**	**(1)**	0.7
(hour, season)_itlSVR	6.025	5.554	(7)	104.536	90.297	(7)	-
(hour, season)_mtlSVR	5.494	5.102	(3)	85.440	73.687	(2)	0.7

To assess statistical significance the Wilcoxon test is used, but instead of testing every pair of models, the rankings of Tables 4.9 and 4.10 are used. The Wilcoxon test is applied between each model and the next in ranking at the 0.05 level, to determine if the difference is significant. The test is applied over the list of errors commited by each model in the patterns of the test set. When the null hypothesis of the Wilcoxon test is refused, it implies that the distribution of the difference between the errors does not have its median at 0. In Table 4.11, we show the *p*-values of the pairwise tests. If the *p*-value is smaller than the level considered level of 0.05, then the different between models is considered significant. With these procedure, a new significance ranking, shown

TABLE 4.11: Wilcoxon *p*-values for absolute (left) and quadratic (right) errors.

	MAE		MSE	
	majorca	tenerife	majorca	tenerife
ctlSVR	0.014 (4)	0.000 (5)	0.081 (4)	0.000 (5)
(season)_itlSVR	0.008 (5)	0.636 (5)	0.215 (4)	0.354 (5)
(season)_mtlSVR	—— (1)	0.000 (4)	0.036 (2)	0.000 (3)
(hour)_itlSVR	0.693 (2)	0.006 (2)	0.000 (3)	0.000 (4)
(hour)_mtlSVR	0.067 (1)	—— (1)	—— (1)	—— (1)
(hour, season)_itlSVR	0.000 (3)	0.000 (6)	0.000 (4)	0.098 (5)
(hour, season)_mtlSVR	0.000 (2)	0.000 (3)	0.745 (3)	0.000 (2)

in Table 4.11, is generated: starting from the best model, we compare each model with the next best one, and we increase the ranking only if the difference is significant. For example, in Table 4.9, in terms of MAE, the first model, (season)_mtlSVR, is tested against the second one, (hour)_mtlSVR. In Table 4.11, we show the *p*-value corresponding to that test, which is 0.067, and since it is larger than the level 0.05, the ranking is not increased and we give to both models the same significance ranking.

Looking at the tables, it is easy to see that the MTL approaches obtain the best results in both problems, while ctlSVR has the worst performance in `tenerife` and second worst in `majorca`. ITL models are more difficult to interpret: although they are always behind their corresponding MTL approaches, they can obtain good results, see the (hour)_itlSVR in `majorca` which is second; but they can also have bad performances, as the (season)_itlSVR in `tenerife`.

The λ^* values can help to understand this behaviour. In both problems, the selected values lie far from the extremes 0 or 1, which distances the MTL approaches from the CTL or ITL ones. If these values are optimal, then the CTL or ITL equivalent models, with $\lambda = 1$ and $\lambda = 0$, respectively, obtain a worse result in validation. This is reflected also in the test set, as shown in the tables. Also, it is noticeable that the optimal values for `majorca` are all smaller than 0.5, which can be intepreted as models with a stronger common part, while in `tenerife`, the optimal values are larger than 0.5 which reflects stronger independent parts. Although the MTL approaches get the best results with any task definition, the (hour) definition seems to work best than the (season) one. The (hour)_mtlSVR gets a second best result, which is not significantly worse than the best one in `majorca`, and is the single best result in `tenerife`.

For completeness the scores of persistence models and a neural network are given here. The persistence forecasts are obtained by predicting at each hour the target value 24 hours prior. By doing this, the MAE scores for `majorca` are 5.776 MWh and 7.766 MWh, which scaled to $[0, 100]$ correspond to 7.97% and 7.21%; which are an 18% and 44% error increase of the best MTL models. The neural network errors are 5.140 MWh and 5.763 MWh, that is, 7.09% and 5.35% of the total PV installed. While this results are still competitive, this performance is worse than those of the convex MTL models proposed.

To get a better understanding of the results we plot the predictions against the target values of the CTL model and the best ITL and MTL ones. In the case of `majorca`, we plot the predictions ctlSVR, (hour)_itlSVR and (season)_mtlSVR in Figure 4.15. From these scatter plots it seems that the CTL approach has a larger deviation in its predictions, while the ITL one has a bias, that is, it systematically underestimates the prediction corresponding to larger values of energy production. The MTL approach seems to correct,
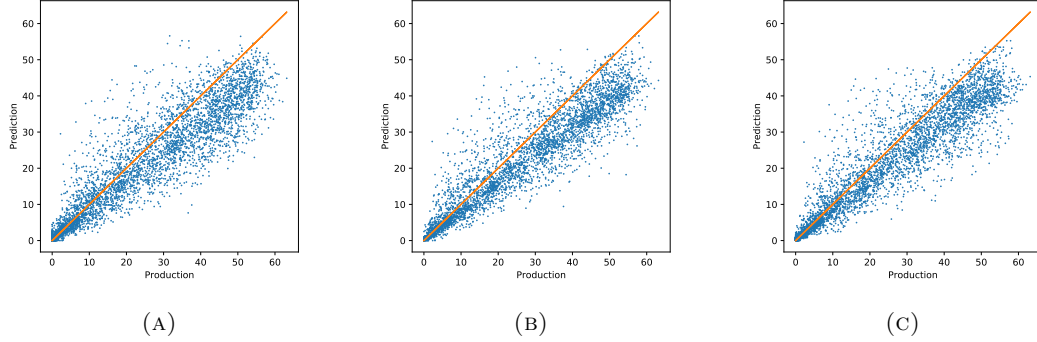
FIGURE 4.15: Real energy production against prediction made by the best CTL(a), ITL(b), and MTL(c) models in `majorca` in terms of MAE; the perfect prediction line is shown in orange. The units of the axis are MWh.
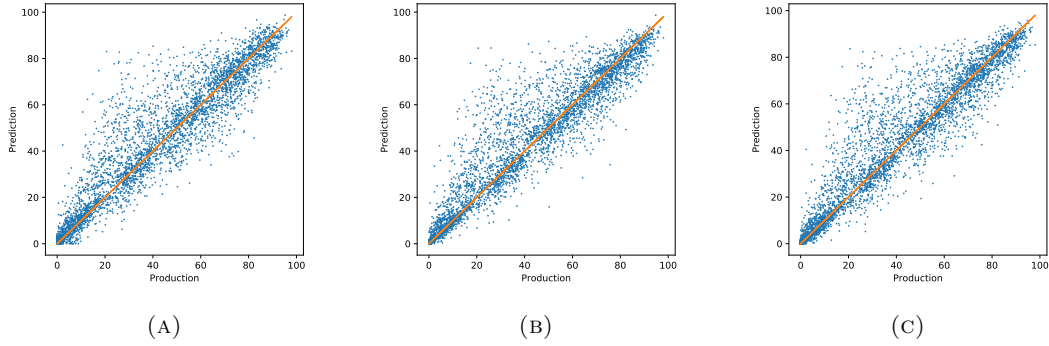

FIGURE 4.16: Real energy production against prediction made by the best CTL (a), ITL (b), and MTL (c) models in `tenerife` in terms of MAE; the perfect prediction line is shown in orange. The units of the axis are MWh.

to a certain degree, this bias of the ITL model, while preserving a smaller variance than the CTL one. For `tenerife` we plot the predictions of ctlSVR, (hour)_itlSVR and (hour)_mtlSVR, which are shown in Figure 4.16. It is more difficult to interpret the plots in this case, although it is possible to highlight that the MTL approach seems less prone to overestimate the production at lower values than either the CTL or ITL models.

4.5.3   Wind Energy

We want to predict the wind energy production at the Sotavento wind park located in Galicia, Spain. Wind energy forecasting is more challenging than the photovoltaic one, due to the unstable nature of the wind, whose behavior is more chaotic than solar radiation. Detecting scenarios where the wind energy production behaves differently but is stable in each one seems crucial, but it is a difficult task. Looking at different characteristics of wind energy production we establish some task definitions and use them to test our proposal.

Data and Tasks.
The variables from the NWP that we consider as predictors are:
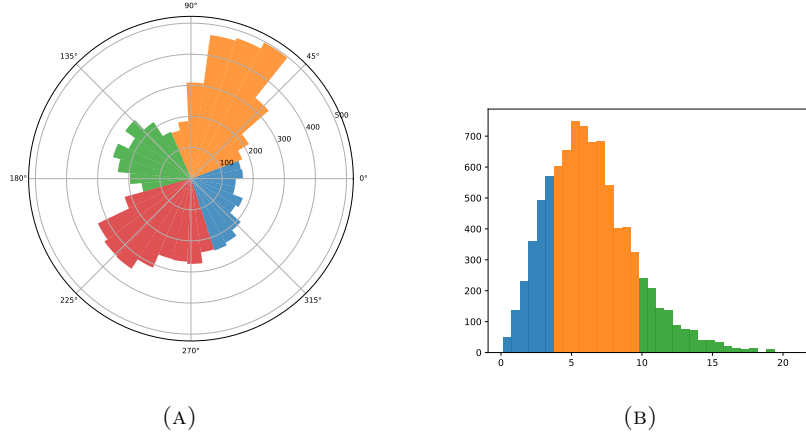
• Eastward component of the wind at 10 m (U10).

FIGURE 4.17: Histograms of wind. (a) Histogram of wind angles derived from NWP data for the year 2016 in Sotavento colored by task angle. (b) Histogram of wind velocity derived from NWP data for the year 2016 and measured in m/s in Sotavento colored by task velocity.
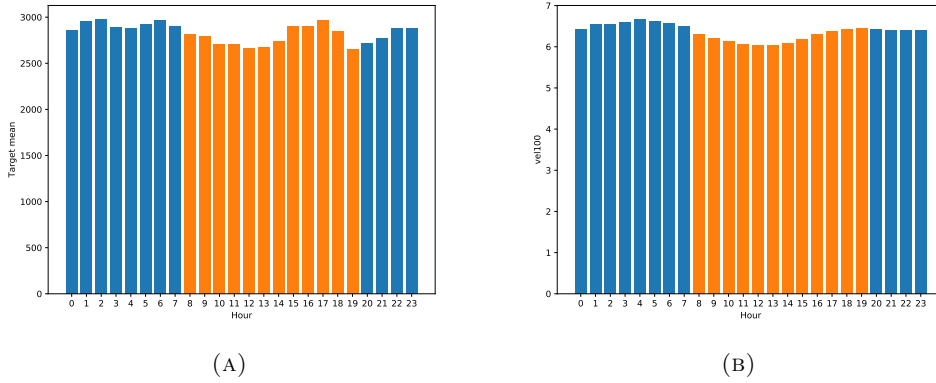

FIGURE 4.18: Histograms of wind. (a) Hourly mean measured in kWh of generated energy during the year 2016 in Sotavento colored by task timeOfDay. (b) Hourly mean of velocity in Sotavento derived from NWP data for the year 2016 and measured in m/s at 100 m colored by task timeOfDay.

- Northward component of the wind at 10 m (V10).

- Module of velocity of the wind at 10 m.

- Eastward component of the wind at 100 m (U100).

- Northward component of the wind at 100 m (V100).

- Module of velocity of the wind at 100 m.

- Surface Pressure (sp).

- 2 meter temperature (2t)

These variables are collected in a grid, which is approximately centered at the farm, with northwest and southeast coordinates at $(-9.5°, 44°)$ and $(-6°, 42.25°)$, respectively,
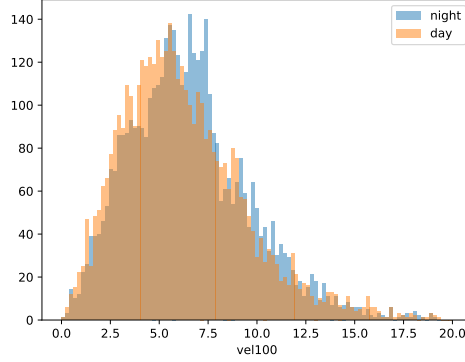
FIGURE 4.19: Histograms of the velocity of wind at 100m derived from NWP data for the year 2016 and measured in m/s during the day and night in Sotavento.

and a spatial resolution of $0.125°$. This results in a grid with 435 points, that, with the 8 variables considered at each point, gives a total number of 3480 predictive features. We scale the energy productions values to $[0, 100]$ using the maximum power installed $(17.56\,\text{MW})$, which corresponds to the value 100. Recall that we have three years of data, 2016, 2017 and 2018, which are used as train, validation and test sets. Although there are no obvious task definitions, we consider three different criteria:

- angle: Here the wind angle at a height of $100m$ is considered, which is obtained from the U100 and V100 variables. First, the most frequent angle is estimated, which is $56°$, and then, we use it as the center of the first quadrant. That is, the patterns that correspond to the first task as those whose wind angle lies between 11 and $101°$ The other three quadrants, corresponding to a task each, are defined by the remaining sectors of $90°$. The histogram of wind angles, and the defined quadrants in different colors, are shown in Figure 4.17a

- velocity: Here the wind velocity at a height of $100m$ is considered, which is again obtained from the U100 and V100 variables. The speed boundaries selected to define three tasks are 4 and 10m/s, which, for an ideal generator, are approximately the starting point of wind energy generation and its maximum power plateau, before cut-off speed. In Figure 4.17b the histrogram of velocities is shown with the task regions colored.

- timeOfDay: Here the 24 hours of a day are divided in two 12 hours periods: a day period between 08 and 19 UTC, and a night one between 20 to 07 UTC. In Figures 4.18a and 4.18b the hourly average energy production and wind speed are shown, with the hours colored according to the two tasks defined. In Figure 4.19 the histograms of wind velocity in the night and day periods are shown.

For these task definitions, it is necessary to perform an analysis of the data, which is done only using the train set, corresponding to 2016. The tasks in this case are not as clear as those defined for the solar energy. For the angle and velocity definitions some differences across tasks in the histograms can be found, but not definitive ones. For the timeOfDay definition, the two histograms of Figure 4.19 look very similar. Moreover, the boundaries of the tasks are set in a way that is partially arbitrary, and a bad selection of tasks could lead to poor results.

TABLE 4.12: Test MAEs (left), MSEs scores (center), with the corresponding rankings, and optimal mixing λ^* (right) of the Sotavento wind energy models considered. The best model errors are shown in bold.

	MAE	MSE	λ^*
ctlSVR	**6.132** (1)	90.228 (2)	-
(velocity)_itlSVR	6.211 (7)	93.363 (7)	-
(velocity)_mtlSVR	6.208 (6)	93.199 (6)	0
(timeOfDay)_itlSVR	6.283 (9)	93.594 (9)	-
(timeOfDay)_mtlSVR	**6.132** (1)	90.228 (2)	1
(timeOfDay, velocity)_itlSVR	6.341 (11)	97.250 (11)	-
(timeOfDay, velocity)_mtlSVR	6.312 (10)	94.774 (10)	0.4
(timeOfDay, angle)_itlSVR	6.266 (8)	93.517 (8)	-
(timeOfDay, angle)_mtlSVR	**6.132** (1)	90.228 (2)	1
(timeOfDay, angle, velocity)_itlSVR	6.410 (12)	102.031 (12)	-
(timeOfDay, angle, velocity)_mtlSVR	**6.132** (1)	90.228 (2)	1
(angle)_itlSVR	6.170 (4)	91.586 (4)	-
(angle)_mtlSVR	6.135 (2)	**90.026** (1)	0.9
(angle, velocity)_itlSVR	6.173 (5)	92.529 (5)	-
(angle, velocity)_mtlSVR	6.168 (3)	90.990 (3)	0.7

TABLE 4.13: Wilcoxon p-values and corresponding rankings for absolute (right) and square (left) wind energy errors in Sotavento. The positions in bold correspond to the model ranked first in terms of MAE or MSE, as indicated by its column.

	MAE	MSE
ctlSVR	—— (1)	—— (2)
(velocity)_itlSVR	0.570 (3)	0.150 (3)
(velocity)_mtlSVR	0.356 (3)	0.466 (3)
(timeOfDay)_itlSVR	0.195 (4)	0.258 (4)
(timeOfDay)_mtlSVR	—— (1)	—— (2)
(timeOfDay, velocity)_itlSVR	0.941 (4)	0.021 (5)
(timeOfDay, velocity)_mtlSVR	0.428 (4)	0.650 (4)
(timeOfDay, angle)_itlSVR	0.000 (4)	0.015 (4)
(timeOfDay, angle)_mtlSVR	—— (1)	—— (2)
(timeOfDay, angle, velocity)_itlSVR	0.090 (4)	0.024 (6)
(timeOfDay, angle, velocity)_mtlSVR	—— (1)	—— (2)
(angle)_itlSVR	0.855 (3)	0.644 (3)
(angle)_mtlSVR	0.035 (2)	—— (1)
(angle, velocity)_itlSVR	0.253 (3)	0.465 (3)
(angle, velocity)_mtlSVR	0.018 (3)	0.001 (3)

Experimental Results.

In Table 4.12 the MAE and MSE scores are shown, and also the ranking is given in parentheses. Unlike the solar energy case, here the CTL approach seems to better suited than using an ITL one. This is also reflected in the selection of λ^* values in the models that get best results, which are close to 1, the CTL equivalent case. That is MTL models such as (timeOfDay)_mtlSVR, (timeOfDay, angle)_mtlSVR and (timeOfDay, angle,
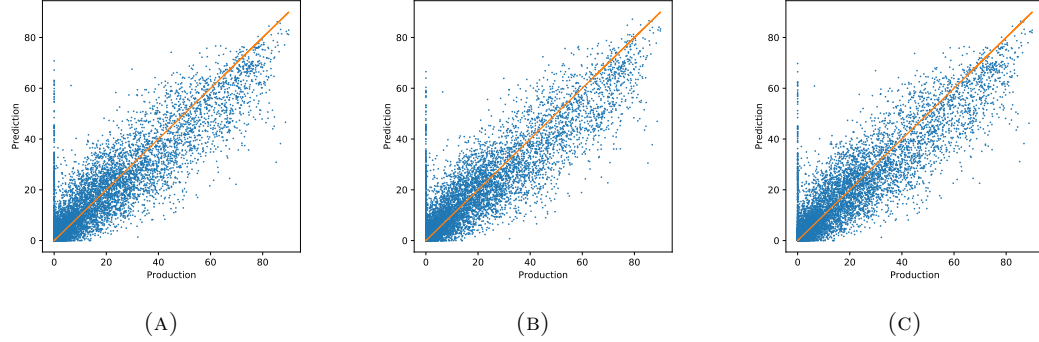
FIGURE 4.20: Real energy production against prediction made by the best CTL a, ITL b, and MTL c models for Sotavento; the perfect prediction line is shown in orange. The units of the axis are percentages points of the total PV energy installed.

velocity)_mtlSVR, where $\lambda^* = 1$, obtain the best results and are equivalent to ctlSVR. Also note the cases of (angle)_mtlSVR and (angle, velocity)_mtlSVR, that are second and third, and where the selected values are $\lambda^* = 0.9$ and $\lambda^* = 0.7$. Those models who put the emphasis on the independent parts, like (timeOfDay, velocity)_mtlSVR, and, of course, the ITL approaches, get worse results. As with the solar energy problems, the statistical significance of the results is tested using the Wilcoxon test. As before, using the ranking of Table 4.12, the significance of the difference between one model and its immediate succesor is tested. In Table 4.13 the p-values of these Wilcoxon tests are given, and the statistical significance ranking is shown. Recall that two models have different significance ranking only if the Wilcoxon test hypothesis is refused. The CTL approach obtains the best results, being the best model in terms of MAE and second best in terms of MSE. Nevertheless, the equivalent MTL approaches that use $\lambda^* = 1$ trivially tie at first place; these are (timeOfDay)_mtlSVR, (timeOfDay, angle)_mtlSVR and (timeOfDay, angle, velocity)_mtlSVR, while the (angle)_mtlSVR gets the second best MAE score. When the MSE scores are analyzed, the roles are reversed, the (angle)_mtlSVR gets the best result, and the ctlSVR and the equivalents MTL approaches are second.

This advantage of the CTL approach can find its roots on poorly defined tasks, which do not have a strong relation with the energy production. Also, the definition of the tasks is made using the train set, data from year 2016, which may not be useful to the validation or test years. Nevertheless, the MTL approaches, having the possibility of blending to either CTL or ITL approaches get the best results too.

In this wind energy problem, the persistence forecasts, which again predict the energy production the previous day at the same hour, obtain an error of 15.64%, which is quite large and represent a 150% increase on the lowest error using SVRs. This is not unusual, since the wind velocity or angle between two different days are not necessarily correlated. With the neural network regressor, the error is a 6.66%, which is also greater than any of the models considered.

Again, we plot the predictions agains the target values of the CTL model and best ITL, (angle)_itlSVR, and pure MTL, (angle)_mtlSVR approaches. The presence of points with zero production is noticeable, but this is relatively frequent in wind energy. It can be caused either by energy curtailments or by maintenance periods of the wind farm. Also, it is appreciable the frequency of small production values, below 20%, which is due to the approximate Weibull distribution of wind speeds, where small speed values have

higher frequencies. With these considerations, it is difficult to compare the plots and find significant differences in model performance.

4.6 Conclusions

In this chapter, we have. . .

Adaptive Graph Laplacian for Multi-Task Learning

5.1 Introduction

In Chapter 3, the MTL strategies are divided into feature-based, parameter-based and joint learning approaches. The feature-based approaches, which try to find a representation shared by all tasks to obtain leverage in the learning process, rely on the assumption that all tasks can indeed share a common latent representation. In the case of joint learning, which combines a common part and task-specific parts in the models, a similar belief is held: although this approach has many good properties, it relies on the assumption that all tasks can share the same common information, which is captured by the common part of the model. However, this might not be the case in some MTL scenarios, where there can exist groups of tasks that share some information, but are unrelated to the rest. The parameter-based approaches are usually reliant on enforcing low-rank matrices, thus, assuming that all tasks parameters belong to the same subspace; however, the graph Laplacian based approaches offer a different perspective.

In the graph Laplacian strategies, the idea is based on penalizing the distance between the parameters of different tasks. It is more natural for linear or kernel approaches, where the models for each task $r = 1, \ldots, T$ are defined as

$$f_r(x) = w_r \cdot \phi(x) + b_r$$

and $\phi(x)$ is a transformation, which can be the identity $\phi(x) = x$ in linear models, or an implicit transformation to an RKHS in kernel models. Here, the models are determined by the parameter w_r, so pushing together these parameters enforces the models to be similar. The idea is to assume that the relationship between tasks can be modelled using a graph, then the adjacency matrix A of such graph is used to define the regularization

$$\sum_{r=1}^{T} \sum_{s=1}^{T} A_{rs} \left\| w_r - w_s \right\|^2, \tag{5.1}$$

where A_{rs} are positive scalars that weight the pairwise distances. Although (5.1) is easy to compute for the linear case, it is not that direct in the case of kernel models where, as shown in the Representer Theorem, the optimal parameters w_r^* are elements of an RKHS. In this Chapter, a framework to use the graph Laplacian regularization with

kernel models is presented. This is implemented with L1, L2 and LS-SVMs. Moreover, the graph Laplacian strategy is combined with the convex joint learning one presented in Chapter 4.

Besides, the definition of the weights A_{rs} is not trivial, and it is crucial for the good performance of this strategy. First, the values A_{rs} have to be bounded, otherwise its interpretability is lost and, moreover, one term can dominate the sum, so only two tasks models would be enforced to be similar. Even with bounded weights, in absence of expert knowledge to select them, it is necessary to find a procedure that finds a set of weights A_{rs} that reflects the real relations between tasks. In this chapter, a data-driven procedure to select the weights for a Laplacian regularization is presented.

5.2 Graph Laplacian Multi-Task Learning with Kernel Methods

5.2.1 Linear Case and Kernel Extension

Linear Case.
Observe that we can express the Laplacian regularization as

$$\Omega(w_1,\ldots,w_T) = \sum_{r=1}^{T}\sum_{s=1}^{T} A_{rs}\left\|w_r - w_s\right\|^2 = \sum_{r=1}^{T}\sum_{s=1}^{T} A_{rs}\left\{\left\|w_r\right\|^2 + \left\|w_s\right\|^2 - 2\langle w_r, w_s\rangle\right\}.$$

Here, only the distance between model parameters is penalized, and in the extreme case where all the tasks can use the same model, i.e. $w_r = w$, there would be no regularization for the complexity of such model w. This problem can be solved by adding the individual model regularization aside from the Laplacian one. To illustrate this, we first use a linear L1-SVM, whose primal problem is

$$\begin{aligned}
\underset{\substack{w_1,\ldots,w_T;\\ b_1,\ldots,b_T;\\ \boldsymbol{\xi};}}{\arg\min} \quad & C\sum_{r=1}^{T}\sum_{i=1}^{m}\xi_i^r + \frac{\nu}{4}\sum_{r=1}^{T}\sum_{s=1}^{T}A_{rs}\left\|w_r - w_s\right\|^2 + \frac{1}{2}\sum_{r}\left\|w_r\right\|^2 \\
\text{s.t.} \quad & y_i^r(w_r \cdot x_i^r + b_r) \ge p_i^r - \xi_i^r,\ i=1,\ldots,m_r;\ r=1,\ldots,T, \\
& \xi_i^r \ge 0,\ i=1,\ldots,m_r;\ r=1,\ldots,T\ .
\end{aligned} \tag{5.2}$$

The corresponding Lagrangian is

$$\begin{aligned}
\mathcal{L}(w_1,\ldots,w_T, b_r, \boldsymbol{\alpha}, \boldsymbol{\beta}) = {} & C\sum_{r=1}^{T}\sum_{i=1}^{m_r}\xi_i^r + \frac{\nu}{2}\sum_{r=1}^{T}\sum_{s=1}^{T}A_{rs}\left\|w_r - w_s\right\|^2 + \frac{1}{2}\sum_{r}\left\|w_r\right\|^2 \\
& - \sum_{r=1}^{T}\sum_{i=1}^{m_r}\alpha_i^r[y_i^r(w_r \cdot x_i^r + b_r) - p_i^r + \xi_i^r] - \sum_{r=1}^{T}\sum_{i=1}^{m_r}\beta_i^r\xi_i^r\ .
\end{aligned} \tag{5.3}$$

And the derivatives of the Lagrangian with respect the primal variables are

$$\frac{\partial \mathcal{L}}{\partial w_r} = 0 \implies w_r^* + \frac{\nu}{2}\sum_{s=1}^{T}(A_{rs} + A_{sr})(w_r^* - w_s^*) = \sum_{i=1}^{m_r}\alpha_i^r y_i^r x_i^r \ , \tag{5.4}$$

$$\frac{\partial \mathcal{L}}{\partial b_r} = 0 \implies \sum_{i=1}^{m_r}\alpha_i^r y_i^r = 0 \ , \tag{5.5}$$

$$\frac{\partial \mathcal{L}}{\partial \xi_i^r} = 0 \implies C_r - \alpha_i^r - \beta_i^r = 0 \ . \tag{5.6}$$

Consider the following definitions of vectors

$$\underset{1\times Td}{\boldsymbol{w}^{\intercal}} = (w_1^{\intercal}\ldots w_T^{\intercal}), \quad \underset{1\times(\sum_r m_r)}{\boldsymbol{\alpha}^T} = (\boldsymbol{\alpha}_1^{\intercal}\ldots\boldsymbol{\alpha}_T^{\intercal}), \quad \underset{1\times m_r}{\boldsymbol{\alpha}_r^{\intercal}} = (\alpha_1^r\ldots\alpha_{m_r}^r);$$

and consider also the matrices

$$\underset{Td\times Td}{E_{\otimes}} = \{I_T + \nu L\}\otimes I_d, \quad \underset{T\times T}{L} = \begin{bmatrix} \sum_{s\neq 1}A_{1s} & -A_{12} & \ldots & -A_{1T} \\ \vdots & \vdots & \ddots & \vdots \\ A_{T1} & A_{T2} & \ldots & \sum_{s\neq T}A_{Ts} \end{bmatrix}$$

and

$$\underset{(\sum_r m_r)\times Td}{\Phi} = \begin{bmatrix} X_1 & 0 & \ldots & 0 \\ 0 & X_2 & \ldots & 0 \\ \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & \ldots & X_T \end{bmatrix}.$$

where X_r is the matrix with examples from task r. Then, we can write (5.4) with a matrix formulation as

$$E\boldsymbol{w} = \Phi^{\intercal}\boldsymbol{\alpha} \implies \boldsymbol{w} = E^{-1}\Phi^{\intercal}\boldsymbol{\alpha}$$

With these definitions and using the derivatives to substitute in the Lagrangian, the result is

$$\begin{aligned} \mathcal{L}(\boldsymbol{w}, \boldsymbol{\alpha}) &= \frac{1}{2}\boldsymbol{w}^{\intercal}E\boldsymbol{w} - \boldsymbol{\alpha}^{\intercal}\Phi\boldsymbol{w} + p^{\intercal}\boldsymbol{\alpha} \\ &= \frac{1}{2}(E^{-1}\Phi^{\intercal}\boldsymbol{\alpha})^{\intercal}EE^{-1}\Phi^{\intercal}\boldsymbol{\alpha} - \boldsymbol{\alpha}^{\intercal}\Phi E^{-1}\Phi^{\intercal}\boldsymbol{\alpha} + p^{\intercal}\boldsymbol{\alpha} \\ &= \frac{1}{2}\boldsymbol{\alpha}^{\intercal}\Phi(E^{\intercal})^{-1}\Phi^{\intercal}\boldsymbol{\alpha} - \boldsymbol{\alpha}^{\intercal}\Phi E^{-1}\Phi^{\intercal}\boldsymbol{\alpha} + p^{\intercal}\boldsymbol{\alpha} \\ &= -\frac{1}{2}\boldsymbol{\alpha}^{\intercal}\Phi E^{-1}\Phi^{\intercal}\boldsymbol{\alpha} + p^{\intercal}\boldsymbol{\alpha}. \end{aligned}$$

Here, the block structure of Φ and E allow to write the following result

$$\Phi E^{-1}\Phi^{\intercal} = \begin{bmatrix} E_{11}^{-1}X_1X_1^{\intercal} & E_{12}^{-1}X_1X_2^{\intercal} & \ldots & E_{1T}^{-1}X_1X_T^{\intercal} \\ E_{21}^{-1}X_2X_1^{\intercal} & E_{22}^{-1}X_2X_2^{\intercal} & \ldots & E_{2T}^{-1}X_2X_T^{\intercal} \\ \vdots & \vdots & \ddots & \vdots \\ E_{T1}^{-1}X_TX_1^{\intercal} & E_{T2}^{-1}X_TX_2^{\intercal} & \ldots & E_{TT}^{-1}X_TX_T^{\intercal} \end{bmatrix}. \tag{5.7}$$

That is, $\Phi E^{-1} \Phi^{\mathsf{T}} = \widetilde{Q}$, where $\widetilde{Q}$ is the kernel matrix corresponding to the kernel function

$$\widetilde{k}(x_i^r, y_j^s) = \left((I_T + \nu L)^{-1}\right)_{rs} \left\langle x_i^r, x_j^s \right\rangle . \tag{5.8}$$

Here, the individual regularization of the primal problem leads to the diagonal term added, which ensures the invertibility of the matrix $(I_T + \nu L)$. Thus, this additional regularization ultimately makes the solution of the problem more numerically stable. Using this result, the corresponding dual problem is

$$
\begin{aligned}
\arg\min_{\boldsymbol{\alpha}} \quad & \Theta(\boldsymbol{\alpha}) = \frac{1}{2}\boldsymbol{\alpha}^t \widetilde{Q}\boldsymbol{\alpha} - p\boldsymbol{\alpha} \\
\text{s.t.} \quad & 0 \leq \alpha_i^r \leq C, \ i = 1, \ldots, m_r; r = 1, \ldots, T \text{ (box constraints)}, \\
& \sum_{i=1}^{n_r} \alpha_i^r y_i^r = 0, \ r = 1, \ldots, T \text{ (equality constraints)} .
\end{aligned}
\tag{5.9}
$$

Non-Linear Case.
When we use an implicit kernel transformation, the result (5.7) is not direct. However, we can use the result from Lemma 3.24. Observe that

$$\boldsymbol{v}^{\mathsf{T}}(I_T \otimes I_d)\boldsymbol{v} = \sum_{r=1}^{T} \|v_r\|^2 ,$$

$$\boldsymbol{v}^{\mathsf{T}}(L \otimes I_d)\boldsymbol{v} = \frac{1}{2}\sum_{r=1}^{T}\sum_{s=1}^{T} A_{rs} \|v_r - v_s\|^2 .$$

To check the second equation, see

$$
\begin{aligned}
\boldsymbol{v}(L \otimes I_d)\boldsymbol{v} &= \boldsymbol{v}(D \otimes I_d)\boldsymbol{v} - \boldsymbol{v}(A \otimes I_d)\boldsymbol{v} \\
&= \sum_{r=1}^{T}\sum_{s=1}^{T} D_{rs} v_r^{\mathsf{T}} v_s - \sum_{r=1}^{T}\sum_{s=1}^{T} A_{rs} v_r^{\mathsf{T}} v_s \\
&= \sum_{r=1}^{T} D_{rr} v_r^{\mathsf{T}} v_r - \sum_{r=1}^{T}\sum_{s=1}^{T} A_{rs} v_r^{\mathsf{T}} v_s \\
&= \sum_{r=1}^{T}\sum_{s=1}^{T} A_{rs} v_r^{\mathsf{T}} v_r - \sum_{r=1}^{T}\sum_{s=1}^{T} A_{rs} v_r^{\mathsf{T}} v_s \\
&= \sum_{r=1}^{T}\sum_{s=1}^{T} A_{rs}(v_r^{\mathsf{T}} v_r - v_r^{\mathsf{T}} v_s) \\
&= \sum_{r=1}^{T}\sum_{s=r}^{T} \{A_{rs}(v_r^{\mathsf{T}} v_r - v_r^{\mathsf{T}} v_s) + A_{sr}(v_s^{\mathsf{T}} v_s - v_s^{\mathsf{T}} v_r)\} \\
&= \sum_{r=1}^{T}\sum_{s=r}^{T} \{(A_{rs} + A_{sr})(v_r^{\mathsf{T}} v_r + v_s^{\mathsf{T}} v_s - 2 v_r^{\mathsf{T}} v_s)\} \\
&= \sum_{r=1}^{T}\sum_{s=r}^{T} \{(A_{rs} + A_{sr}) \|v_r - v_s\|^2\} \\
&= \frac{1}{2}\sum_{r=1}^{T}\sum_{s=1}^{T} \{(A_{rs} + A_{sr}) \|v_r - v_s\|^2\} \\
&= \sum_{r=1}^{T}\sum_{s=1}^{T} \{A_{rs} \|v_r - v_s\|^2\}.
\end{aligned}
$$

where

$$
\boldsymbol{v} = \sum_{t=1}^{T} e_r \otimes v_r,
$$

and $e_1, \ldots, e_T$ are the canonical base of $\mathbb{R}^T$. Then, we can write a general primal problem with Laplacian regularization using kernels as

$$
R(\boldsymbol{v}) = C \sum_{r=1}^{T}\sum_{i=1}^{m} \ell(y_i^r, \langle \boldsymbol{v}, e_r \otimes \phi(x_i^r)\rangle) + (\boldsymbol{v}^{\mathsf{T}}((\nu L + I_T) \otimes I)\boldsymbol{v}), \tag{5.10}
$$

As shown in Lemma 3.24, this is equivalent to solving a dual problem where the kernel function is

$$
\widetilde{k}(x_i^r, x_j^s) = (\nu L + I_T)_{rs}^{-1} k(x_i^r, x_j^s)
$$

where $k(\cdot, \cdot)$ is the reproducing kernel induced by the implicit transformation $\phi(\cdot)$. One example, using the L1-SVM would be the primal problem shown in (5.2) where an implicit transformation $\phi(\cdot)$ is used. The corresponding dual problem is, therefore, the problem shown in (5.9), but where the kernel matrix $\widetilde{Q}$ is defined using the kernel function (5.8).

5.2.2 Convex Graph Laplacian MTL

In Ruiz et al. (2021a) we proposed a convex formulation of the Graph Laplacian MTL SVM which includes a convex combination of a common part and task-specific parts that are coupled through a Laplacian regularization. That is, the models for each task are

$$h_r(x) = \lambda_r w \cdot \phi(x) + \lambda_r v_r \cdot \psi(x),$$

where $\phi(\cdot)$ and $\psi(\cdot)$ are the implicit transformations for the common part and task-specific parts, and can be possibly distinct, trying to capture different properties of the data. That is, $\phi : \mathcal{X} \to \mathcal{H}_\phi$ and $\psi : \mathcal{X} \to \mathcal{H}_\psi$, where $\mathcal{H}_\phi$ and $\mathcal{H}_\psi$ are RKHS's with reproducing kernels $k_\phi(\cdot, \cdot)$ and $k_\psi(\cdot, \cdot)$, respectively. Unlike the model definition for Convex MTL in (4.6), where each task-specific part can use a different Hilbert space, here all tasks must use the same two transformations. The common $\phi(\cdot)$ must be obviously equal for all tasks, but also the specific one $\psi(\cdot)$ must be the same for all tasks because to impose a Laplacian regularization, all the parameters v_r have to be elements from the same space. Again, as with the Convex MTL approach, parameters $\lambda_r \in [0, 1]$ define how relevant the common part for each task. When $\lambda_r = 1$, only the common part is present, while $\lambda_r = 0$ results in task-specific models that, now, are coupled through the Laplacian regularization.

A general result for Convex Graph Laplacian MTL.
In general, we can use the Representer Theorem with the tensor kernels defined in Chapter 3 to find the result of kernel problems using this approach. Using the same approach used in Section 3.3, with $e_1, \ldots, e_T$ being an orthonormal basis of $\mathbb{R}^T$, we define

$$\boldsymbol{v} = \sum_{t=1}^{T} e_r \otimes v_r \in \mathbb{R}^T \otimes \mathcal{H}_\psi, \tag{5.11}$$

such that $\langle \boldsymbol{v}, e_r \otimes \psi(x_i^r) \rangle = \langle v_r, \psi(x_i^r) \rangle$. If we consider then the product space $\mathcal{H}_\phi \times (\mathbb{R}^T \otimes \mathcal{H}_\psi)$, where the norm of $(w, \boldsymbol{v})$ is

$$\langle (w, \boldsymbol{v}), (w, \boldsymbol{v}) \rangle = \langle w, w \rangle + \langle \boldsymbol{v}, \boldsymbol{v} \rangle = \|w\|^2 + \left\|\boldsymbol{v}^2\right\|,$$

then the general kernelized problem for convex graph Laplacian MTL can be expressed as

$$
\underset{(w,\boldsymbol{v}) \in \mathcal{H}_\phi \times (\mathbb{R}^T \otimes \mathcal{H}_\psi)}{\arg\min} R((w, \boldsymbol{v})) = \sum_{r=1}^{T} \sum_{i=1}^{m_r} \ell(y_i^r, (\langle (w, \boldsymbol{v}), (\lambda_r \phi(x_i^r), (1 - \lambda_r)(e_r \otimes \psi(x_i^r))) \rangle))
$$
$$
+ \left\langle (w, \boldsymbol{v}), (I_{\mathcal{H}_\phi} \times (E \otimes I_{\mathcal{H}_\psi}))(w, \boldsymbol{v}) \right\rangle,
$$
$$\tag{5.12}$$

where ℓ is a loss function, $\mathcal{H}_\phi$, in our case it is the identity operator in $\mathcal{H}_\phi$, and E is a symmetric, positive definite operator in $\mathbb{R}^T$ in our case, $E = ((\nu L + I))$, with L being a graph Laplacian matrix. Using a modification of Lemma 3.24, we can state the following one

Lemma 5.1. *The predictions $\langle (w^*, \boldsymbol{v}^*), (\phi(x_i^r), e_r \otimes \psi(x_i^r)) \rangle$ of the solution $(w^*, \boldsymbol{v}^*)$ from the Multi-Task optimization problem (5.12) can be obtained solving the problem*

$$S((w, \boldsymbol{u})) = \sum_{r=1}^{T} \sum_{i=1}^{m_r} \ell(y_i^r, \langle (w, \boldsymbol{u}), (\lambda_r \phi(x_i^r), (1-\lambda_r)(B_r \otimes \psi(x_i^r))) \rangle) + \mu (\langle w, w \rangle + \langle \boldsymbol{u}, \boldsymbol{u} \rangle),$$
(5.13)

where $w \in \mathcal{H}_\phi$ and $\boldsymbol{u} \in \mathbb{R}^p \otimes \mathcal{H}_\psi$ with $p \geq T$ and B_r are the columns of a full rank matrix $B \in \mathbb{R}^{p \times T}$ such that $E^{-1} = B^\mathsf{T} B$.

Proof. Again, since $E \in \mathbb{R}^{T \times T}$ is positive definite, we can find $B \in \mathbb{R}^{p \times T}, p \geq T$ and $\operatorname{rank} B = T$ such that $E^{-1} = B^\mathsf{T} B$, using for example the SVD; then,

$$E^{-1} \otimes I = (B^\mathsf{T} B) \otimes I = (B^\mathsf{T} \otimes I_{\mathcal{H}_\psi})(B \otimes I),$$

Consider the change of variable $\boldsymbol{v} = (B^\mathsf{T} \otimes I_{\mathcal{H}_\psi})\boldsymbol{u}$, where $\boldsymbol{u} \in \mathbb{R}^p \otimes \mathcal{H}$, which can always be done because B is full rank, then

$$R(w, \boldsymbol{u}) = \sum_{r=1}^{T} \sum_{i=1}^{m_r} \ell(y_i^r, \langle (w, (B^\mathsf{T} \otimes I_{\mathcal{H}_\psi})\boldsymbol{v}), (\lambda_r \phi(x_i^r), (1-\lambda_r)(e_r \otimes \psi(x_i^r))) \rangle)$$
$$+ \mu \langle (w, (B^\mathsf{T} \otimes I_{\mathcal{H}_\psi})\boldsymbol{v}), (I_{\mathcal{H}_\phi} \times (E \otimes I_{\mathcal{H}_\psi}))(w, (B^\mathsf{T} \otimes I_{\mathcal{H}_\psi})\boldsymbol{v}) \rangle$$

This is equivalent to problem (5.13), where the regularizer is increasing in $\|(w, \boldsymbol{u})\|^2$, so can apply the Representer theorem, which states that the minimizer of $S(w, \boldsymbol{u})$ has the form

$$(w^*, \boldsymbol{u}^*) = \sum_{r=1}^{T} \sum_{i=1}^{m_r} \alpha_i^r (\lambda_r \phi(x_i^r), (1-\lambda_r)(B_r \otimes \psi(x_i^r))),$$
(5.14)

Using the correspondence between $\boldsymbol{u}^*$ and $\boldsymbol{v}^*$,

$$(w^*, \boldsymbol{v}^*) = (w^*, (B^\mathsf{T} \otimes I_{\mathcal{H}_\psi})\boldsymbol{u}^*)$$
$$= \sum_{r=1}^{T} \sum_{i=1}^{m_r} \alpha_i^r (\lambda_r \phi(x_i^r), (1-\lambda_r) \operatorname{vect}((\langle B_1, B_r \rangle, \ldots, \langle B_T, B_r \rangle)) \otimes \psi(x_i^r)).$$

Then, we can recover the predictions corresponding to the solutions $(w^*, \boldsymbol{v}^*)$ as

$$\langle (w^*, \boldsymbol{v}^*), (\lambda_t \phi(\hat{x}^t), (1-\lambda_t)e_t \otimes \psi(\hat{x}^t)) \rangle$$
$$= \left\langle \sum_{r=1}^{T} \sum_{i=1}^{m_r} \alpha_i^r (w, \operatorname{vect}((\langle B_1, B_r \rangle, \ldots, \langle B_T, B_r \rangle)) \otimes \phi(x_i^r)), (\phi(\hat{x}^t), e_t \otimes \psi(\hat{x}^t)) \right\rangle$$
$$= \sum_{r=1}^{T} \sum_{i=1}^{m_r} \alpha_i^r (\lambda_r \lambda_t \langle \phi(x_i^r), \phi(x_i^r) \rangle + (1-\lambda_r)(1-\lambda_t) \langle B_s, B_r \rangle \langle \psi(x_i^r), \psi(\hat{x}^t) \rangle)$$
$$= \sum_{r=1}^{T} \sum_{i=1}^{m_r} \alpha_i^r (\lambda_r \lambda_t \langle \phi(x_i^r), \phi(x_i^r) \rangle + (1-\lambda_r)(1-\lambda_t)E_{rs}^{-1} \langle \psi(x_i^r), \psi(\hat{x}^t) \rangle).$$

$\square$

As with Lemma 3.24, the Multi-Task problem 5.12 can be expressed as a Single-Task problem with the Multi-Task kernel function

$$\widehat{\widehat{k}}(x_i^r, x_j^s) = \lambda_r \lambda_s k_\phi(x_i^r, x_j^s) + (1 - \lambda_r)(1 - \lambda_s)(\nu L + I_T)_{rs}^{-1} k_\psi(x_i^r, x_j^s), \qquad (5.15)$$

where $k_\phi(\cdot, \cdot)$ and $k_\psi(\cdot, \cdot)$ are the reproducing kernels corresponding to transformations ϕ and ψ, respectively.

Convex Graph Laplacian L1-SVM.

The primal problem for the linear L1-SVM using this approach, as shown in (Ruiz et al., 2021a), is the following one

$$\underset{\substack{v_1,\dots,v_T; \\ b_1,\dots,b_T; \\ \boldsymbol{\xi}, w;}}{\arg\min} \quad C \sum_{r=1}^{T} \sum_{i=1}^{m_r} \xi_i^r + \frac{\nu}{2} \sum_{r=1}^{T} \sum_{s=1}^{T} A_{rs} \|v_r - v_s\|^2 + \frac{1}{2} \sum_r \|v_r\|^2 + \frac{1}{2} \|w\|^2$$

$$\text{s.t.} \quad y_i^r(\lambda_r (w \cdot x_i^r) + (1 - \lambda_r)(v_r \cdot x_i^r) + b_r) \geq p_i^r - \xi_i^r,$$

$$\xi_i^r \geq 0, \ i = 1, \dots, m_r, \ r = 1, \dots, T. \qquad (5.16)$$

Note that with $\mu = 0$, this problem is equivalent to the one shown in (4.17). The extension to the kernel case requires using a different formulation, that is

$$\underset{\substack{\boldsymbol{v}; \\ b_1,\dots,b_T; \\ \boldsymbol{\xi}, w;}}{\arg\min} \quad C \sum_{r=1}^{T} \sum_{i=1}^{m_r} \xi_i^r + \frac{\nu}{2} \boldsymbol{v}^\mathsf{T}(L \otimes I_\mathcal{H})\boldsymbol{v} + \frac{1}{2} \boldsymbol{v}^\mathsf{T}(I_T \otimes I_\mathcal{H})\boldsymbol{v} + \frac{1}{2} \|w\|^2$$

$$\text{s.t.} \quad y_i^r(\lambda_r \langle w, \phi(x_i^r) \rangle + (1 - \lambda_r)(\langle \boldsymbol{v}, e_r \otimes \psi(x_i^r) \rangle) + b_r) \geq p_i^r - \xi_i^r,$$

$$\xi_i^r \geq 0, \ i = 1, \dots, m_r, \ r = 1, \dots, T. \qquad (5.17)$$

Although the result from Lemma 5.1 can be applied for this problem, for illustration purposes the whole procedure is shown for this particular case. The Lagrangian corresponding to (5.17) is

$$\mathcal{L}(w, \boldsymbol{v}, b_r, \xi_i^r, \boldsymbol{\alpha}, \boldsymbol{\beta})$$

$$= C \sum_{r=1}^{T} \sum_{i=1}^{m_r} \xi_i^r + \frac{\nu}{2} \boldsymbol{v}^\mathsf{T}(L \otimes I_\mathcal{H})\boldsymbol{v} + \frac{1}{2} \boldsymbol{v}^\mathsf{T}(I_T \otimes I_\mathcal{H})\boldsymbol{v} + \frac{1}{2} \|w\|^2$$

$$- \sum_{r=1}^{T} \sum_{i=1}^{m_r} \alpha_i^r [y_i^r(\lambda_r \langle w, \phi(x_i^r) \rangle + (1 - \lambda_r)(\langle \boldsymbol{v}, e_r \otimes \psi(x_i^r) \rangle) + b_r) - p_i^r + \xi_i^r]$$

$$- \sum_{r=1}^{T} \sum_{i=1}^{m_r} \beta_i^r \xi_i^r, \qquad (5.18)$$

where $\alpha_i^r, \beta_i^r \geq 0$. Taking derivatives

$$\nabla_w \mathcal{L} = 0 \implies w^* = \sum_{r=1}^{T} \lambda_r \sum_{i=1}^{m_r} \alpha_i^r \left\{ y_i^r \phi(x_i^r) \right\}, \tag{5.19}$$

$$\nabla_{\boldsymbol{v}} \mathcal{L} = 0 \implies \left((I_T + \nu L) \otimes I_{\mathcal{H}} \right) \boldsymbol{v} = \sum_{r=1}^{T} (1 - \lambda_r) \sum_{i=1}^{m_r} \alpha_i^r y_i^r (e_r \otimes \psi(x_i^r)), \tag{5.20}$$

$$\nabla_{b_r} \mathcal{L} = 0 \implies \sum_{i=1}^{m_r} \alpha_i^r y_i^r = 0, \tag{5.21}$$

$$\nabla_{\xi_i^r} \mathcal{L} = 0 \implies C - \alpha_i^r - \beta_i^r = 0. \tag{5.22}$$

With

$$E = (I_T + \nu L), \tag{5.23}$$

and

$$E_{\otimes} = (E \otimes I_{\mathcal{H}}), \tag{5.24}$$

and using these results to substitute in the Lagrangian, the result is

$$
\begin{aligned}
&\mathcal{L}(w, \boldsymbol{v}, b_r, \xi_i^r, \boldsymbol{\alpha}, \boldsymbol{\beta}) \\
&= \frac{1}{2} \langle \boldsymbol{v}, E_{\otimes} \boldsymbol{v} \rangle + \frac{1}{2} \langle w, w \rangle \\
&\quad - \sum_{r=1}^{T} \sum_{i=1}^{m_r} \alpha_i^r [y_i^r (\lambda_r \langle w, \phi(x_i^r) \rangle + (1 - \lambda_r)(\langle \boldsymbol{v}, e_r \otimes \psi(x_i^r) \rangle) + b_r) - p_i^r] \\
&= \frac{1}{2} \left\langle E_{\otimes}^{-1} \sum_{r=1}^{T} (1 - \lambda_r) \sum_{i=1}^{m_r} \alpha_i^r y_i^r (e_r \otimes \psi(x_i^r)), E_{\otimes} E_{\otimes}^{-1} \sum_{r=1}^{T} (1 - \lambda_r) \sum_{i=1}^{m_r} \alpha_i^r y_i^r (e_r \otimes \psi(x_i^r)) \right\rangle \\
&\quad + \frac{1}{2} \left\langle \sum_{r=1}^{T} \lambda_r \sum_{i=1}^{m_r} \alpha_i^r y_i^r \phi(x_i^r), \sum_{r=1}^{T} \lambda_r \sum_{i=1}^{m_r} \alpha_i^r y_i^r \phi(x_i^r) \right\rangle \\
&\quad - \sum_{r=1}^{T} (1 - \lambda_r) \sum_{i=1}^{m_r} \alpha_i^r y_i^r \left\{ \left\langle E_{\otimes}^{-1} \sum_{s=1}^{T} (1 - \lambda_s) \sum_{j=1}^{m_s} \alpha_j^s y_j^s (e_s \otimes \psi(x_j^s)), e_r \otimes \psi(x_i^r) \right\rangle \right\} \\
&\quad - \sum_{r=1}^{T} \lambda_r \sum_{i=1}^{m_r} \alpha_i^r y_i^r \left\{ \left\langle \sum_{s=1}^{T} \lambda_s \sum_{j=1}^{m_s} \alpha_j^s y_j^s \phi(x_j^s), \phi(x_i^r) \right\rangle \right\} - \sum_{r=1}^{T} \sum_{i=1}^{m_r} \alpha_i^r p_i^r \\
&= \frac{1}{2} \sum_{r,s=1}^{T} (1 - \lambda_r)(1 - \lambda_s) \sum_{i,j=1}^{m_r, m_s} \alpha_j^s \alpha_i^r y_j^s y_i^r \left\langle (e_s \otimes \psi(x_j^s)), E_{\otimes}^{-1} (e_r \otimes \psi(x_i^r)) \right\rangle \\
&\quad + \frac{1}{2} \sum_{r,s=1}^{T} \lambda_r \lambda_s \sum_{i,j=1}^{m_r, m_s} \alpha_j^s \alpha_i^r y_j^s y_i^r \left\langle \phi(x_j^s), \phi(x_i^r) \right\rangle \\
&\quad - \sum_{r,s=1}^{T} (1 - \lambda_r)(1 - \lambda_s) \sum_{i,j=1}^{m_r, m_s} \alpha_j^s \alpha_i^r y_j^s y_i^r \left\langle (e_s \otimes \psi(x_j^s)), E_{\otimes}^{-1} (e_r \otimes \psi(x_i^r)) \right\rangle \\
&\quad - \sum_{r,s=1}^{T} \lambda_r \lambda_s \sum_{i,j=1}^{m_r, m_s} \alpha_j^s \alpha_i^r y_j^s y_i^r \left\langle \phi(x_j^s), \phi(x_i^r) \right\rangle - \sum_{r=1}^{T} \sum_{i=1}^{m_r} \alpha_i^r p_i^r.
\end{aligned}
$$

Due to (5.22) and $\alpha_i^r, \beta_i^r \geq 0$, we have the box constraints $0 \leq \alpha_i^r \leq C$; then, the dual problem is

$$
\begin{aligned}
\arg\min_{\boldsymbol{\alpha}} \quad & \Theta(\boldsymbol{\alpha}) = \frac{1}{2}\boldsymbol{\alpha}^t \left(\Lambda Q\Lambda + (I_n - \Lambda)\,\widetilde{Q}\,(I_n - \Lambda) \right) \boldsymbol{\alpha} - \boldsymbol{p\alpha} \\
\text{s.t.} \quad & 0 \leq \alpha_i^r \leq C, \ i = 1, \ldots, m_r; r = 1, \ldots, T, \\
& \sum_{i=1}^{n_r} \alpha_i^r y_i^r = 0, \ r = 1, \ldots, T.
\end{aligned}
\tag{5.25}
$$

Here, we use the matrix

$$
\Lambda = \mathrm{diag}(\overbrace{\lambda_1, \ldots, \lambda_1}^{m_1}, \ldots, \overbrace{\lambda_T, \ldots, \lambda_T}^{m_T}),
\tag{5.26}
$$

I_n is the $n \times n$ identity matrix, with $n = \sum_{r=1}^{T} m_r$, Q is the common, standard, kernel matrix, and $\widetilde{Q}$ is the kernel matrix with the graph Laplacian information. We can define the kernel matrix

$$
\widehat{\widetilde{Q}} = \Lambda Q\Lambda + (I_n - \Lambda)\,\widetilde{Q}\,(I_n - \Lambda),
\tag{5.27}
$$

as the matrix that is generated using the kernel function (5.15).

Convex Graph Laplacian L2-SVM.
The primal problem for convex graph Laplacian MTL based on the L2-SVM is

$$
\begin{aligned}
\arg\min_{\substack{\boldsymbol{v}; \\ b_1, \ldots, b_T; \\ \boldsymbol{\xi}, w;}} \quad & C\sum_{r=1}^{T}\sum_{i=1}^{m_r} (\xi_i^r)^2 + \frac{\nu}{2}\boldsymbol{v}^{\mathsf{T}}(L \otimes I_{\mathcal{H}})\boldsymbol{v} + \frac{1}{2}\boldsymbol{v}^{\mathsf{T}}(I_T \otimes I_{\mathcal{H}})\boldsymbol{v} + \frac{1}{2}\|w\|^2 \\
\text{s.t.} \quad & y_i^r(\lambda_r \langle w, \phi(x_i^r) \rangle + (1 - \lambda_r)(\langle \boldsymbol{v}, e_r \otimes \psi(x_i^r) \rangle) + b_r) \geq p_i^r - \xi_i^r,
\end{aligned}
\tag{5.28}
$$

and the corresponding Lagrangian is

$$
\begin{aligned}
\mathcal{L}&(w, \boldsymbol{v}, b_r, \xi_i^r, \boldsymbol{\alpha}) \\
&= C\sum_{r=1}^{T}\sum_{i=1}^{m_r} (\xi_i^r)^2 + \frac{\nu}{2}\boldsymbol{v}^{\mathsf{T}}(L \otimes I_{\mathcal{H}})\boldsymbol{v} + \frac{1}{2}\boldsymbol{v}^{\mathsf{T}}(I_T \otimes I_{\mathcal{H}})\boldsymbol{v} + \frac{1}{2}\|w\|^2 \\
&\quad - \sum_{r=1}^{T}\sum_{i=1}^{m_r} \alpha_i^r[y_i^r(\lambda_r \langle w, \phi(x_i^r) \rangle + (1 - \lambda_r)(\langle \boldsymbol{v}, e_r \otimes \psi(x_i^r) \rangle) + b_r) - p_i^r + \xi_i^r],
\end{aligned}
\tag{5.29}
$$

where $\alpha_i^r \geq 0$. The gradients with respect to the primal variables are

$$
\nabla_w \mathcal{L} = 0 \implies w^* = \sum_{r=1}^{T} \lambda_r \sum_{i=1}^{m_r} \alpha_i^r \{y_i^r \phi(x_i^r)\},
\tag{5.30}
$$

$$
\nabla_{\boldsymbol{v}} \mathcal{L} = 0 \implies ((I_T + \nu L) \otimes I_{\mathcal{H}})\,\boldsymbol{v} = \sum_{r=1}^{T} (1 - \lambda_r) \sum_{i=1}^{m_r} \alpha_i^r y_i^r (e_r \otimes \psi(x_i^r)),
\tag{5.31}
$$

$$
\nabla_{b_r} \mathcal{L} = 0 \implies \sum_{i=1}^{m_r} \alpha_i^r y_i^r = 0,
\tag{5.32}
$$

$$
\nabla_{\xi_i^r} \mathcal{L} = 0 \implies C\xi_i^r - \alpha_i^r = 0.
\tag{5.33}
$$

Using $E_\otimes$ as defined in (5.24) and substituting in the Lagrangian, we get

$$\mathcal{L}(w, \boldsymbol{v}, b_r, \xi_i^r, \boldsymbol{\alpha}, \boldsymbol{\beta})$$

$$= \frac{1}{2C}\sum_{r=1}^{T}\sum_{i=1}^{m_r}(\alpha_i^r)^2 - \frac{1}{C}\sum_{r=1}^{T}\sum_{i=1}^{m_r}(\alpha_i^r)^2 + \frac{1}{2}\langle \boldsymbol{v}, E_\otimes \boldsymbol{v}\rangle + \frac{1}{2}\langle w, w\rangle$$

$$- \sum_{r=1}^{T}\sum_{i=1}^{m_r}\alpha_i^r[y_i^r(\lambda_r\langle w, \phi(x_i^r)\rangle + (1-\lambda_r)(\langle \boldsymbol{v}, e_r\otimes\psi(x_i^r)\rangle) + b_r) - p_i^r]$$

$$= \frac{1}{2C}\sum_{r=1}^{T}\sum_{i=1}^{m_r}(\alpha_i^r)^2 - \frac{1}{C}\sum_{r=1}^{T}\sum_{i=1}^{m_r}(\alpha_i^r)^2$$

$$+ \frac{1}{2}\left\langle E_\otimes^{-1}\sum_{r=1}^{T}(1-\lambda_r)\sum_{i=1}^{m_r}\alpha_i^r y_i^r(e_r\otimes\psi(x_i^r)), E_\otimes E_\otimes^{-1}\sum_{r=1}^{T}(1-\lambda_r)\sum_{i=1}^{m_r}\alpha_i^r y_i^r(e_r\otimes\psi(x_i^r))\right\rangle$$

$$+ \frac{1}{2}\left\langle \sum_{r=1}^{T}\lambda_r\sum_{i=1}^{m_r}\alpha_i^r y_i^r\phi(x_i^r), \sum_{r=1}^{T}\lambda_r\sum_{i=1}^{m_r}\alpha_i^r y_i^r\phi(x_i^r)\right\rangle$$

$$- \sum_{r=1}^{T}(1-\lambda_r)\sum_{i=1}^{m_r}\alpha_i^r y_i^r\left\{\left\langle E_\otimes^{-1}\sum_{s=1}^{T}(1-\lambda_s)\sum_{j=1}^{m_s}\alpha_j^s y_j^s(e_s\otimes\psi(x_j^s)), e_r\otimes\psi(x_i^r)\right\rangle\right\}$$

$$- \sum_{r=1}^{T}\lambda_r\sum_{i=1}^{m_r}\alpha_i^r y_i^r\left\{\left\langle \sum_{s=1}^{T}\lambda_s\sum_{j=1}^{m_s}\alpha_j^s y_j^s\phi(x_j^s), \phi(x_i^r)\right\rangle\right\} - \sum_{r=1}^{T}\sum_{i=1}^{m_r}\alpha_i^r p_i^r$$

$$= \frac{1}{2C}\sum_{r=1}^{T}\sum_{i=1}^{m_r}(\alpha_i^r)^2 - \frac{1}{C}\sum_{r=1}^{T}\sum_{i=1}^{m_r}(\alpha_i^r)^2$$

$$\frac{1}{2}\sum_{r,s=1}^{T}(1-\lambda_r)(1-\lambda_s)\sum_{i,j=1}^{m_r,m_s}\alpha_j^s\alpha_i^r y_j^s y_i^r\langle (e_s\otimes\psi(x_j^s)), E_\otimes^{-1}(e_r\otimes\psi(x_i^r))\rangle$$

$$+ \frac{1}{2}\sum_{r,s=1}^{T}\lambda_r\lambda_s\sum_{i,j=1}^{m_r,m_s}\alpha_j^s\alpha_i^r y_j^s y_i^r\langle \phi(x_j^s), \phi(x_i^r)\rangle$$

$$- \sum_{r,s=1}^{T}(1-\lambda_r)(1-\lambda_s)\sum_{i,j=1}^{m_r,m_s}\alpha_j^s\alpha_i^r y_j^s y_i^r\langle (e_s\otimes\psi(x_j^s)), E_\otimes^{-1}(e_r\otimes\psi(x_i^r))\rangle$$

$$- \sum_{r,s=1}^{T}\lambda_r\lambda_s\sum_{i,j=1}^{m_r,m_s}\alpha_j^s\alpha_i^r y_j^s y_i^r\langle \phi(x_j^s), \phi(x_i^r)\rangle - \sum_{r=1}^{T}\sum_{i=1}^{m_r}\alpha_i^r p_i^r.$$

The dual problem for the L2-SVM based formulation is then

$$\begin{aligned}
\operatorname*{arg\,min}_{\boldsymbol{\alpha}} \quad &\Theta(\boldsymbol{\alpha}) = \frac{1}{2}\boldsymbol{\alpha}^t\left\{\left(\Lambda Q\Lambda + (I_n - \Lambda)\widetilde{Q}(I_n - \Lambda)\right) + \frac{1}{C}I_n\right\}\boldsymbol{\alpha} - \boldsymbol{p}\boldsymbol{\alpha} \\
\text{s.t.} \quad &0 \le \alpha_i^r, \ i = 1, \ldots, m_r; r = 1, \ldots, T, \\
&\sum_{i=1}^{n_r}\alpha_i^r y_i^r = 0, \ r = 1, \ldots, T.
\end{aligned} \tag{5.34}$$

Here, again we use the matrix Λ defined in (5.26). Now, there are no box constraints, but an additional diagonal term that can be interpreted as a soft constraint for the dual

coefficients α_i^r.

Convex Graph Laplacian LS-SVM.

Recall that in the LS-SVM (Suykens and Vandewalle, 1999) the inequalities in the constraints are substituted for equalities, which lead to a simpler dual solution. The primal problem for the convex graph Laplacian formulation based on the LS-SVM is

$$\underset{\substack{\boldsymbol{v}; \\ b_1,\dots,b_T; \\ \boldsymbol{\xi}, w;}}{\arg\min} \quad C \sum_{r=1}^{T} \sum_{i=1}^{m_r} (\xi_i^r)^2 + \frac{\nu}{2} \boldsymbol{v}^\mathsf{T} (L \otimes I_\mathcal{H}) \boldsymbol{v} + \frac{1}{2} \boldsymbol{v}^\mathsf{T} (I_T \otimes I_\mathcal{H}) \boldsymbol{v} + \frac{1}{2} \|w\|^2 \tag{5.35}$$

$$\text{s.t.} \qquad y_i^r (\lambda_r \langle w, \phi(x_i^r) \rangle + (1 - \lambda_r)(\langle \boldsymbol{v}, e_r \otimes \psi(x_i^r) \rangle) + b_r) = p_i^r - \xi_i^r.$$

The Lagrangian corresponding to this optimization problem is

$$\mathcal{L}(w, \boldsymbol{v}, b_r, \xi_i^r, \boldsymbol{\alpha})$$

$$= C \sum_{r=1}^{T} \sum_{i=1}^{m_r} (\xi_i^r)^2 + \frac{\nu}{2} \boldsymbol{v}^\mathsf{T} (L \otimes I_\mathcal{H}) \boldsymbol{v} + \frac{1}{2} \boldsymbol{v}^\mathsf{T} (I_T \otimes I_\mathcal{H}) \boldsymbol{v} + \frac{1}{2} \|w\|^2 \tag{5.36}$$

$$- \sum_{r=1}^{T} \sum_{i=1}^{m_r} \alpha_i^r [y_i^r (\lambda_r \langle w, \phi(x_i^r) \rangle + (1 - \lambda_r)(\langle \boldsymbol{v}, e_r \otimes \psi(x_i^r) \rangle) + b_r) - p_i^r + \xi_i^r],$$

where now, unlike in the L1 and L2-SVM cases, there are no restrictions for the Lagrange multipliers α_i^r. The KKT conditions for this problem are then

$$\nabla_w \mathcal{L} = 0 \implies w^* = \sum_{r=1}^{T} \lambda_r \sum_{i=1}^{m_r} \alpha_i^r \left\{ y_i^r \phi(x_i^r) \right\}, \tag{5.37}$$

$$\nabla_{\boldsymbol{v}} \mathcal{L} = 0 \implies ((I_T + \nu L) \otimes I_\mathcal{H}) \, \boldsymbol{v} = \sum_{r=1}^{T} (1 - \lambda_r) \sum_{i=1}^{m_r} \alpha_i^r y_i^r (e_r \otimes \psi(x_i^r)), \tag{5.38}$$

$$\nabla_{b_r} \mathcal{L} = 0 \implies \sum_{i=1}^{m_r} \alpha_i^r y_i^r = 0, \tag{5.39}$$

$$\nabla_{\xi_i^r} \mathcal{L} = 0 \implies C \xi_i^r - \alpha_i^r = 0, \tag{5.40}$$

$$\nabla_{\alpha_i^r} \mathcal{L} = 0 \implies y_i^r (\lambda_r \langle w, \phi(x_i^r) \rangle + (1 - \lambda_r)(\langle \boldsymbol{v}, e_r \otimes \psi(x_i^r) \rangle) + b_r) + \xi_i^r = p_i^r. \tag{5.41}$$

Substituting in (5.41), with $E_\otimes$ as defined in (5.24), we get

$$y_i^r (\lambda_r \langle w, \phi(x_i^r) \rangle + (1 - \lambda_r)(\langle \boldsymbol{v}, e_r \otimes \psi(x_i^r) \rangle) + b_r) + \xi_i^r = p_i^r$$

$$\implies \lambda_r \left\langle \sum_{s=1}^{T} \lambda_s \sum_{j=1}^{m_s} \alpha_i^s \left\{ y_i^s \phi(x_j^s) \right\}, y_i^r \phi(x_i^r) \right\rangle$$

$$+ (1 - \lambda_r) \left\langle E_\otimes^{-1} \sum_{s=1}^{T} (1 - \lambda_s) \sum_{j=1}^{m_s} \alpha_i^s y_i^s (e_s \otimes \psi(x_i^s)), y_i^r (e_r \otimes \psi(x_i^r)) \right\rangle + b_r + \frac{\alpha_i^r}{C} = p_i^r.$$

For each coefficient α_i^r we get an equality like this one, which, all together, form a system of equations that can be expressed as

$$\left[\begin{array}{c|c} 0_{T\times T} & A^\intercal Y \\ \hline YA & \left(\Lambda Q\Lambda + (I_n - \Lambda)\,\widetilde{Q}\,(I_n - \Lambda)\right) + \frac{1}{C}I_n \end{array}\right] \begin{bmatrix} b_1 \\ \vdots \\ b_T \\ \boldsymbol{\alpha} \end{bmatrix} = \begin{bmatrix} \mathbf{0}_T \\ \boldsymbol{p} \end{bmatrix}. \qquad (5.42)$$

Again, Λ is the matrix defined in (5.26), and we have the kernel matrix that is defined using the kernel function (5.15).

5.3   Adaptive Graph Laplacian Algorithm

Until now, in this chapter we have seen graph Laplacian formulations for kernel methods, where we assume that there exists a graph whose nodes represent the tasks, and the weights of the edges determine the pairwise relations between tasks. If A is the graph adjacency matrix containing these weights, we use the Laplacian regularizer presented in (5.1) to enforce a coupling between tasks according to the adjacency information. To do this, an adjacency matrix A must be chosen a priori, and it defines the optimization problem. However, these weights of A are not known in real-world problems. Even if we have an expert knowledge of the problem at hand, manually selecting the weight between each pair of tasks seems unfeasible, even for a few tasks. It is more sensible to use a data-driven approach and automatically learn the matrix A. In this section, we propose one method to learn A from data, discuss the procedure and computational cost, and also show how the distances can be computed in kernel spaces.

Entropy-based Interpretation.
To explain our data-driven procedure for selecting the adjacency weights, first we would like the matrix A to meet the following requirements:

- A is symmetric.

- All the weights A_{rs} are positive, for $r, s = 1, \ldots, T$.

- The rows of A add up to 1.

The first one is a necessary condition to express the regularizer of (5.1) using the Laplacian matrix L. The second and third requirements offer a better interpretability of the matrix A, since we can view each row as a probability distribution. Then, we can look at the entropy of the rows to study how connected each task is to the rest. That is, let $\boldsymbol{a}^r$ be the row for task r, its entropy can be computed as

$$H(\boldsymbol{a}^r) = -\sum_{s=1}^{T} a_s^r \log(a_s^r).$$

Observe that this is a non-negative quantity since $a_{rs} \in [0, 1]$ due the conditions stated before, and reaches its maximum when the distribution is uniform. If the weight at $a_r^r = A_{rr}$ is 1 and the rest is 0, then the r-th task is not connected to any other task and the entropy is minimal. In the other extreme case, when $\boldsymbol{a}^r = \frac{1}{T}\mathbf{1}_T^\intercal$, the task is equally connected to all the other tasks and the entropy is maximal.

With these considerations, there are two trivial options when choosing the adjacency matrix: using a diagonal matrix A, i.e. $A = I_T$, where there are no connections between different tasks and the entropy of the rows is minimal, and using an agnostic view, with the constant matrix $A = \frac{1}{T}\mathbf{1}_T\mathbf{1}_T^\mathsf{T}$ where the degree of relation is the same among all tasks and the entropy of the rows is maximal.

For a better understanding of these situation we look at the following simplified formulation for the convex graph Laplacian MTL problem,

$$
\begin{aligned}
\underset{\substack{v_1,\ldots,v_T; \\ b_1,\ldots,b_T; \\ w;}}{\arg\min} \quad & C \sum_{r=1}^{T}\sum_{i=1}^{m_r} \ell(\lambda_r(w \cdot \phi(x_i^r)) + (1-\lambda_r)(v_r \cdot \psi(x_i^r)) + b_r, y_i^r) \\
& + \nu \sum_{r=1}^{T}\sum_{s=1}^{T} A_{rs}\|v_r - v_s\|^2 + \sum_{r=1}^{T}\|v_r\|^2 + \|w\|^2 .
\end{aligned}
$$

When we use the minimal entropy matrix $A = I_T$, it is equivalent to the problem

$$
\underset{\substack{v_1,\ldots,v_T; \\ b_1,\ldots,b_T; \\ w;}}{\arg\min} \quad C \sum_{r=1}^{T}\sum_{i=1}^{m_r} \ell(\lambda_r(w \cdot \phi(x_i^r)) + (1-\lambda_r)(v_r \cdot \psi(x_i^r)) + b_r, y_i^r) + \sum_{r=1}^{T}\|v_r\|^2 + \|w\|^2,
$$

which is a convex MTL approach, without Laplacian information. Here, the task-specific models are completely independent, with no coupling between them. In the case that we use the constant matrix $A = \frac{1}{T}\mathbf{1}_T\mathbf{1}_T^\mathsf{T}$, if ν is large enough, it is equivalent to

$$
\underset{\substack{v; \\ b_1,\ldots,b_T; \\ w;}}{\arg\min} \quad C \sum_{r=1}^{T}\sum_{i=1}^{m_r} \ell(\lambda_r(w \cdot \phi(x_i^r)) + (1-\lambda_r)(v \cdot \psi(x_i^r)) + b_r, y_i^r) + T\|v\|^2 + \|w\|^2
$$

where a convex combination of two common models is use. In the linear case, or when $\phi = \psi$, it is equivalent to using a CTL approach, where a single common model for all tasks.

Between these two extremes of minimal and maximal entropy, there exists a whole range of matrices with intermediate entropy rows. Our goal is then to find an adjacency matrix A that reflects the underlying tasks relations and, therefore, helps to improve the learning process.

Optimization Procedure.
To find such adjacency matrix, we rely on the distances computed between the task parameters, if two task parameters are close, those tasks should be strongly related. Consider the Laplacian term

$$
\sum_{r=1}^{T}\sum_{s=1}^{T} A_{rs}\|v_r - v_s\|^2 ;
$$

then, the matrix that minimizes this quantity is the diagonal matrix $A = I_T$ with minimal entropy rows. The interpretation of this solution is that each task is isolated and the greater degree of connection is with itself; however, this is a trivial choice of matrix A, because it does not give any information about the underlying task relations. To avoid falling in the minimal entropy solution, the entropy of the rows is penalized. The

optimization problem to be solved is

$$\underset{\substack{v_1,\dots,v_T;\\ b_1,\dots,b_T;\\ w;\\ A\in(\mathbb{R}_{\geq 0})^T\times(\mathbb{R}_{\geq 0})^T,\\ A\mathbf{1}_T=\mathbf{1}_T}}{\arg\min} \quad C\sum_{r=1}^{T}\sum_{i=1}^{m_r}\ell(\lambda_r(w\cdot\phi(x_i^r))+(1-\lambda_r)(v_r\cdot\psi(x_i^r))+b_r,y_i^r)$$

$$+\frac{\nu}{2}\sum_{r=1}^{T}\sum_{s=1}^{T}A_{rs}\left\|v_r-v_s\right\|^2+\frac{1}{2}\sum_{r=1}^{T}\left\|v_r\right\|^2+\frac{1}{2}\left\|w\right\|^2$$

$$-\mu\sum_{r=1}^{T}H(\boldsymbol{a}^r). \tag{5.43}$$

Here, $\mathbb{R}_{\geq 0}$ are the non-negative real numbers, and $(\mathbb{R}_{\geq 0})^T\times(\mathbb{R}_{\geq 0})^T$ are the $T\times T$ real matrices with non-negative entries. The condition $A\mathbf{1}_T=\mathbf{1}_T$ is to enforce that the rows add up to 1. The parameter μ regulates how much the entropy is penalized, that is, how close the solution for A should be to the constant matrix. Using the tensor product formulation of (5.12), this problem is expressed as

$$\underset{\substack{(w,\boldsymbol{v})\in\mathcal{H}_\phi\times(\mathbb{R}^T\otimes\mathcal{H}_\psi);\\ A\in(\mathbb{R}_{\geq 0})^T\times(\mathbb{R}_{\geq 0})^T,\\ A\mathbf{1}_T=\mathbf{1}_T}}{\arg\min} \sum_{r=1}^{T}\sum_{i=1}^{m_r}\ell(y_i^r,(\langle(w,\boldsymbol{v}),(\lambda_r\phi(x_i^r),(1-\lambda_r)(e_r\otimes\psi(x_i^r)))\rangle))$$

$$+\left\langle(w,\boldsymbol{v}),(I_{\mathcal{H}_\phi}\times(E\otimes I_{\mathcal{H}_\psi}))(w,\boldsymbol{v})\right\rangle-\mu\sum_{r=1}^{T}H(\boldsymbol{a}^r), \tag{5.44}$$

where $E=(\nu L+I_T)$, $L=\mathrm{diag}(A\mathbf{1}_T)-A$, and $\boldsymbol{v}$ has been defined in (5.11). When the matrix A is fixed and we optimize in the task parameters $w,\boldsymbol{v}$, the optimization problem is (5.12); however, the dual problem is the one that is solved, and optimal dual coefficients $\boldsymbol{\alpha}^*$ are obtained. In the case of the L1-SVM-based model, problem (5.25) is used, while for the L2, and LS-SVM variants, we have (5.34) and (5.42), respectively.

When $w,\boldsymbol{v}$ are fixed and we want to find the optimal matrix A, the problem is

$$\underset{\substack{A\in(\mathbb{R}_{\geq 0})^T\times(\mathbb{R}_{\geq 0})^T,\\ A\mathbf{1}_T=\mathbf{1}_T}}{\arg\min} \frac{\nu}{2}\sum_{r=1}^{T}\sum_{s=1}^{T}A_{rs}\left\|v_r-v_s\right\|^2-\mu\sum_{r=1}^{T}H(\boldsymbol{a}^r). \tag{5.45}$$

Here the entropy term seems unavoidable since without it the trivial solution would be $A_{rs}=\delta_{rs}$, the Dirac delta, which corresponds to the minimum-entropy solution $A=I_T$; however, using the entropy term different solutions for A can be obtained. This is a separable problem for each row of $\boldsymbol{A}$ and by taking derivatives of the objective function in (5.45), we have

$$\frac{\partial}{\partial A_{rs}}J(A)=\frac{1}{2}\left(\nu\left\|v_r-v_s\right\|^2+\mu\log A_{rs}+\mu\right),$$

and setting it to zero we get that $A_{rs} \propto \exp{-\frac{\nu}{\mu} \|v_r - v_s\|^2}$; then, since $\sum_s A_{rs} = 1$, the result is

$$A_{rs} = \frac{\exp{-\frac{\nu}{\mu} \|v_r - v_s\|^2}}{\sum_t \exp{-\frac{\nu}{\mu} \|v_r - v_t\|^2}}. \tag{5.46}$$

Starting from an agnostic point of view, with a fixed constant, maximal entropy matrix A^0, where all tasks share the same of degree of relation, we solve the problem shown (5.12), and obtain the optimal parameters $\boldsymbol{v}$. Then, we can compute the distances between each pair of parameters $v_r, v_s, r, s = 1, \ldots, T$, and use these distances to obtain an adjacency matrix A^1 using (5.46). This procedure can be then repeated with the new adjacency matrix, until convergence of the objective function value or a maximum number of iterations is reached. That is, the algorithm consists on an iterated procedure where the following steps are repeated until convergence:

- Solve the convex GL MTL problem, as shown in (5.12), with fixed matrix A

- Compute the distances between task parameters v_r

- Update the matrix A according to these distances using (5.46)

Distance Computation.
In our proposed algorithm, the steps for optimizing the task parameters $(w, \boldsymbol{v})$ and, given the distances $\|v_r - v_s\|^2$, the adjacency matrix are detailed; however, it is still necessary to show how these distances can be computed when $\boldsymbol{v}_r$ are elements of the RKHS $\mathcal{H}_\psi$.

Recall that the first step requires solving a dual problem, where optimal dual coefficients α^* are obtained. Then, using the result from (5.14), we have that

$$\boldsymbol{v} = ((I_T + \nu L) \otimes I_{\mathcal{H}})^{-1} \sum_{r=1}^{T} (1 - \lambda_r) \sum_{i=1}^{m_r} \alpha_i^r y_i^r (e_r \otimes \psi(x_i^r)).$$

Then, since the distances can be expressed as

$$\|v_r - v_s\|^2 = \langle v_r, v_r \rangle + \langle v_s, v_s \rangle - 2 \langle v_r, v_s \rangle,$$

we are interested in computing the dot products $\langle v_r, v_s \rangle$ for $r, s = 1, \ldots, T$. This inner products, using our formulation for $\boldsymbol{v}$ are

$$\langle v_r, v_s \rangle = \langle (e_r^\mathsf{T} \otimes I_{\mathcal{H}}) \boldsymbol{v}, (e_s^\mathsf{T} \otimes I_{\mathcal{H}}) \boldsymbol{v} \rangle,$$

where

$$(e_r^\mathsf{T} \otimes I_{\mathcal{H}}) \boldsymbol{v} = (e_r^\mathsf{T} \otimes I_{\mathcal{H}}) ((I_T + \nu L) \otimes I_{\mathcal{H}})^{-1} \sum_{t=1}^{T} (1 - \lambda_t) \sum_{i=1}^{m_t} \alpha_i^t y_i^t (e_t \otimes \psi(x_i^t))$$

$$= \sum_{t=1}^{T} (1 - \lambda_t) \sum_{i=1}^{m_t} \alpha_i^t y_i^t \left((e_r^\mathsf{T} (I_T + \nu L)^{-1} e_t) \otimes \psi(x_i^t) \right)$$

$$= \sum_{t=1}^{T} (1 - \lambda_t) \sum_{i=1}^{m_t} \alpha_i^t y_i^t \left(((I_T + \nu L)_{rt}^{-1}) \otimes \psi(x_i^t) \right).$$

Input: $(X, y) = \{(x_i^r, y_i^r), i = 1, \ldots, m_r; r = 1, \ldots, T\}$ `// Data`
Output: $\boldsymbol{\alpha}^*$ `// Optimal dual coefficients`
Output: A^* `// Optimal adjacency matrix`
Data: params $= \{C, \lambda, \nu, \mu, \sigma_\phi, \sigma_\psi(, \epsilon)\}$ `// Hyperparameters`
$o^{\mathrm{old}} = \infty$
$A = A_0$ `// Constant matrix`
while *True* **do**
    $L_{\mathrm{inv}} \leftarrow \mathrm{getInvLaplacian}(A)$ `// Step 0`
    $\alpha_{\mathrm{opt}} \leftarrow \mathrm{solveDualProblem}((X, y), L_{\mathrm{inv}}, \mathrm{params})$ `// Step 1`
    $o \leftarrow \mathrm{computeObjectiveValue}((X, y), L_{\mathrm{inv}}, \alpha_{\mathrm{opt}})$ `// objective function value`
    if $o^{old} - o \leq \delta_{tol}$ **then**
      break   `// Exit condition`
    end
    $o^{old} \leftarrow o$
    $D \leftarrow \mathrm{computeDistances}((X, y), L_{\mathrm{inv}}, \alpha_{\mathrm{opt}})$ `// Step 2`
    $A \leftarrow \mathrm{updateAdjMatrix}(D, \mathrm{params})$ `// Step 3`
end
return α_{opt}, A

Algorithm 2: Adaptive graph Laplacian algorithm.

Then, using E as defined in (5.23), the inner product is

$$\langle v_r, v_s \rangle$$
$$= \langle (e_r^{\mathsf{T}} \otimes I_{\mathcal{H}}) \boldsymbol{v}, (e_s^{\mathsf{T}} \otimes I_{\mathcal{H}}) \boldsymbol{v} \rangle$$
$$= \left\langle \sum_{t=1}^{T} (1 - \lambda_t) \sum_{i=1}^{m_t} \alpha_i^t y_i^t \left((E_{rt}^{-1}) \otimes \psi(x_i^t) \right), \sum_{\tau=1}^{T} (1 - \lambda_\tau) \sum_{i=1}^{m_t} \alpha_i^\tau y_i^\tau \left((E_{st}^{-1}) \otimes \psi(x_i^\tau) \right) \right\rangle$$
$$= \sum_{t=1}^{T} \sum_{\tau=1}^{T} (1 - \lambda_t)(1 - \lambda_\tau) \sum_{i=1}^{m_t} \sum_{i=1}^{m_t} \alpha_i^t y_i^t \alpha_i^\tau y_i^\tau \left\langle \left((E_{rt}^{-1}) \otimes \psi(x_i^t) \right), \left((E_{st}^{-1}) \otimes \psi(x_i^\tau) \right) \right\rangle,$$

which, using a matrix formulation can be expressed as

$$\langle v_r, v_s \rangle = \boldsymbol{\alpha}^{\mathsf{T}} (I_n - \Lambda) \widetilde{Q^{rs}} (I_n - \Lambda) \boldsymbol{\alpha}, \tag{5.47}$$

where $\widetilde{Q^{rs}}$ is the kernel matrix computed using the kernel function

$$\widetilde{k^{rs}}(x_i^t, x_j^\tau) = (I_T + \nu L)_{rt}^{-1} (I_T + \nu L)_{s\tau}^{-1} k_\psi(x_i^t, x_j^\tau). \tag{5.48}$$

That is, the distances are computed as

$$\|v_r - v_s\|^2 = \boldsymbol{\alpha}^{\mathsf{T}} (I_n - \Lambda) (\widetilde{Q^{rr}} + \widetilde{Q^{ss}} - 2\widetilde{Q^{rs}}) (I_n - \Lambda) \boldsymbol{\alpha}. \tag{5.49}$$

Analysis and Computational Cost.
In first place, a comment about the convergence of the proposed algorithm, shown in Algorithm 2, is necessary. Given a convex loss function ℓ, the optimization problem of (5.44) is convex in the parameters $(w, \boldsymbol{v})$ and in A, but not jointly convex in $(w, \boldsymbol{v}, A)$. Then, an iterated procedure where a coordinated optimization is done, as our proposal, ensures that a local minimum is found, albeit, not a global one.

To study the computational cost of our algorithm we consider the cost of each step, where, for a simpler formulation, we will use $n = \sum_{r=1}^{T} m_r$. At each iteration we need to perform three steps: solving a dual problem, computing the distances, updating the adjacency matrix A; and also the inverse of the $T \times T$ Laplacian matrix, which is used in the first and second step, is needed. The inversion of the Laplacian matrix has a cost of $C_0 = T^3$. The first step, where the dual problem is solved, has the standard cost corresponding to these SVM variants, which all have a cost $C_1 = O(n^{2+\epsilon})$ such that $O(n^2) \leq C_1 \leq O(n^3)$. The second step, the distances computation one, involves the product of $T \times T$ inner products $\langle v_r, v_s \rangle$ shown in (5.47), each with a cost n_{sv}^2 with N_{sv} being the number of support vectors; then, the cost of the second step is $C_2 = O(T^2 N_{\text{sv}}^2)$. Finally, the third step, involving the update of the adjacency matrix A, needs the computation of the $T \times T$ elements of A using equation (5.46), which has then a total cost of $C_3 = T^2$. The total computational cost of each iteration is then

$$C_0 + C_1 + C_2 + C_3 = O(T^3 + n^{2+\epsilon} + T^2 n_{\text{sv}}^2 + T^2)$$

Assuming a standard situation, where n is much larger than T, the two first steps have clearly the greater cost; however, it is difficult to determine which one has greater cost, since it depends on the specific problem being solved. Anyway, the second step, unlike the first one, can be easily parallelized, computing each distance at the same time, which would result in a cost of $O(n_{\text{sv}}^2)$.

5.4   Conclusions

In this chapter, we have...

Bibliography

Agarwal, A., Daumé III, H., and Gerber, S. (2010). Learning multiple tasks using manifold regularization. In Lafferty, J. D., Williams, C. K. I., Shawe-Taylor, J., Zemel, R. S., and Culotta, A., editors, *Advances in Neural Information Processing Systems 23: 24th Annual Conference on Neural Information Processing Systems 2010. Proceedings of a meeting held 6-9 December 2010, Vancouver, British Columbia, Canada*, pages 46–54. Curran Associates, Inc.

Akhiezer, N. I. and Glazman, I. M. (1961). Theory of linear operators in hilbert space.

Álvarez, M. A., Rosasco, L., and Lawrence, N. D. (2012). Kernels for vector-valued functions: A review. *Found. Trends Mach. Learn.*, 4(3):195–266.

Ando, R. K. and Zhang, T. (2005). A framework for learning predictive structures from multiple tasks and unlabeled data. *J. Mach. Learn. Res.*, 6:1817–1853.

Argyriou, A., Clémençon, S., and Zhang, R. (2013). Learning the graph of relations among multiple tasks. *HAL*.

Argyriou, A., Evgeniou, T., and Pontil, M. (2006). Multi-task feature learning. In Schölkopf, B., Platt, J. C., and Hofmann, T., editors, *Advances in Neural Information Processing Systems 19, Proceedings of the Twentieth Annual Conference on Neural Information Processing Systems, Vancouver, British Columbia, Canada, December 4-7, 2006*, pages 41–48. MIT Press.

Argyriou, A., Evgeniou, T., and Pontil, M. (2008). Convex multi-task feature learning. *Mach. Learn.*, 73(3):243–272.

Argyriou, A., Micchelli, C. A., Pontil, M., and Ying, Y. (2007). A spectral regularization framework for multi-task structure learning. In Platt, J. C., Koller, D., Singer, Y., and Roweis, S. T., editors, *Advances in Neural Information Processing Systems 20, Proceedings of the Twenty-First Annual Conference on Neural Information Processing Systems, Vancouver, British Columbia, Canada, December 3-6, 2007*, pages 25–32. Curran Associates, Inc.

Baldassarre, L., Rosasco, L., Barla, A., and Verri, A. (2012). Multi-output learning via spectral filtering. *Mach. Learn.*, 87(3):259–301.

Bartlett, P. L. and Mendelson, S. (2002). Rademacher and gaussian complexities: Risk bounds and structural results. *J. Mach. Learn. Res.*, 3:463–482.

Barzilai, A. and Crammer, K. (2015). Convex multi-task learning by clustering. In Lebanon, G. and Vishwanathan, S. V. N., editors, *Proceedings of the Eighteenth International Conference on Artificial Intelligence and Statistics, AISTATS 2015, San Diego, California, USA, May 9-12, 2015*, volume 38 of *JMLR Workshop and Conference Proceedings*. JMLR.org.

Bauschke, H. H. and Combettes, P. L. (2011). *Convex Analysis and Monotone Operator Theory in Hilbert Spaces*. CMS Books in Mathematics. Springer.

Baxter, J. (2000). A model of inductive bias learning. *Journal of artificial intelligence research*, 12:149–198.

Ben-David, S. and Borbely, R. S. (2008). A notion of task relatedness yielding provable multiple-task learning guarantees. *Mach. Learn.*, 73(3):273–287.

Ben-David, S. and Schuller, R. (2003). Exploiting task relatedness for mulitple task learning. In Schölkopf, B. and Warmuth, M. K., editors, *Computational Learning Theory and Kernel Machines, 16th Annual Conference on Computational Learning Theory and 7th Kernel Workshop, COLT/Kernel 2003, Washington, DC, USA, August 24-27, 2003, Proceedings*, volume 2777 of *Lecture Notes in Computer Science*, pages 567–580. Springer.

Bergstra, J. and Bengio, Y. (2012). Random search for hyper-parameter optimization. *J. Mach. Learn. Res.*, 13:281–305.

Bonilla, E. V., Chai, K. M. A., and Williams, C. K. I. (2007). Multi-task gaussian process prediction. In Platt, J. C., Koller, D., Singer, Y., and Roweis, S. T., editors, *Advances in Neural Information Processing Systems 20, Proceedings of the Twenty-First Annual Conference on Neural Information Processing Systems, Vancouver, British Columbia, Canada, December 3-6, 2007*, pages 153–160. Curran Associates, Inc.

Burges, C. J. C. (1998). A tutorial on support vector machines for pattern recognition. *Data Min. Knowl. Discov.*, 2(2):121–167.

Cai, F. and Cherkassky, V. (2009). SVM+ regression and multi-task learning. In *International Joint Conference on Neural Networks, IJCNN 2009, Atlanta, Georgia, USA, 14-19 June 2009*, pages 418–424. IEEE Computer Society.

Cai, F. and Cherkassky, V. (2012). Generalized SMO algorithm for svm-based multitask learning. *IEEE Trans. Neural Networks Learn. Syst.*, 23(6):997–1003.

Caruana, R. (1997). Multitask learning. *Mach. Learn.*, 28(1):41–75.

Cavallanti, G., Cesa-Bianchi, N., and Gentile, C. (2010). Linear algorithms for online multitask classification. *J. Mach. Learn. Res.*, 11:2901–2934.

Chen, J., Liu, J., and Ye, J. (2010). Learning incoherent sparse and low-rank patterns from multiple tasks. In Rao, B., Krishnapuram, B., Tomkins, A., and Yang, Q., editors, *Proceedings of the 16th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, Washington, DC, USA, July 25-28, 2010*, pages 1179–1188. ACM.

Chen, J., Tang, L., Liu, J., and Ye, J. (2009). A convex formulation for learning shared structures from multiple tasks. In Danyluk, A. P., Bottou, L., and Littman, M. L., editors, *Proceedings of the 26th Annual International Conference on Machine Learning, ICML 2009, Montreal, Quebec, Canada, June 14-18, 2009*, volume 382 of *ACM International Conference Proceeding Series*, pages 137–144. ACM.

Chen, J., Zhou, J., and Ye, J. (2011). Integrating low-rank and group-sparse structures for robust multi-task learning. In Apté, C., Ghosh, J., and Smyth, P., editors, *Proceedings of the 17th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, San Diego, CA, USA, August 21-24, 2011*, pages 42–50. ACM.

Ciliberto, C., Mroueh, Y., Poggio, T. A., and Rosasco, L. (2015). Convex learning of multiple tasks and their structure. In Bach, F. R. and Blei, D. M., editors, *Proceedings of the 32nd International Conference on Machine Learning, ICML 2015, Lille, France, 6-11 July 2015*, volume 37 of *JMLR Workshop and Conference Proceedings*, pages 1548–1557. JMLR.org.

Crammer, K. and Mansour, Y. (2012). Learning multiple tasks using shared hypotheses. In Bartlett, P. L., Pereira, F. C. N., Burges, C. J. C., Bottou, L., and Weinberger, K. Q., editors, *Advances in Neural Information Processing Systems 25: 26th Annual Conference on Neural Information Processing Systems 2012. Proceedings of a meeting held December 3-6, 2012, Lake Tahoe, Nevada, United States*, pages 1484–1492.

Daumé III, H. (2007). Frustratingly easy domain adaptation. In Carroll, J. A., van den Bosch, A., and Zaenen, A., editors, *ACL 2007, Proceedings of the 45th Annual Meeting of the Association for Computational Linguistics, June 23-30, 2007, Prague, Czech Republic*. The Association for Computational Linguistics.

Daumé III, H. (2009). Bayesian multitask learning with latent hierarchies. In Bilmes, J. A. and Ng, A. Y., editors, *UAI 2009, Proceedings of the Twenty-Fifth Conference on Uncertainty in Artificial Intelligence, Montreal, QC, Canada, June 18-21, 2009*, pages 135–142. AUAI Press.

Dinuzzo, F. (2013). Learning output kernels for multi-task problems. *Neurocomputing*, 118:119–126.

Evgeniou, T., Micchelli, C. A., and Pontil, M. (2005). Learning multiple tasks with kernel methods. *J. Mach. Learn. Res.*, 6:615–637.

Evgeniou, T. and Pontil, M. (2004). Regularized multi–task learning. In Kim, W., Kohavi, R., Gehrke, J., and DuMouchel, W., editors, *Proceedings of the Tenth ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, Seattle, Washington, USA, August 22-25, 2004*, pages 109–117. ACM.

Fung, G. and Mangasarian, O. L. (2001). Proximal support vector machine classifiers. In Lee, D., Schkolnick, M., Provost, F. J., and Srikant, R., editors, *Proceedings of the seventh ACM SIGKDD international conference on Knowledge discovery and data mining, San Francisco, CA, USA, August 26-29, 2001*, pages 77–86. ACM.

Ghifary, M., Kleijn, W. B., Zhang, M., and Balduzzi, D. (2015). Domain generalization for object recognition with multi-task autoencoders. In *IEEE International Conference on Computer Vision, ICCV*, pages 2551–2559. IEEE Computer Society.

Gong, P., Ye, J., and Zhang, C. (2012a). Multi-stage multi-task feature learning. In Bartlett, P. L., Pereira, F. C. N., Burges, C. J. C., Bottou, L., and Weinberger, K. Q., editors, *Advances in Neural Information Processing Systems 25: 26th Annual Conference on Neural Information Processing Systems 2012. Proceedings of a meeting held December 3-6, 2012, Lake Tahoe, Nevada, United States*, pages 1997–2005.

Gong, P., Ye, J., and Zhang, C. (2012b). Robust multi-task feature learning. In Yang, Q., Agarwal, D., and Pei, J., editors, *The 18th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, KDD '12, Beijing, China, August 12-16, 2012*, pages 895–903. ACM.

Han, L. and Zhang, Y. (2015). Learning tree structure in multi-task learning. In Cao, L., Zhang, C., Joachims, T., Webb, G. I., Margineantu, D. D., and Williams, G., editors, *Proceedings of the 21th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, Sydney, NSW, Australia, August 10-13, 2015*, pages 397–406. ACM.

Han, L. and Zhang, Y. (2016). Multi-stage multi-task learning with reduced rank. In Schuurmans, D. and Wellman, M. P., editors, *Proceedings of the Thirtieth AAAI Conference on Artificial Intelligence, February 12-17, 2016, Phoenix, Arizona, USA*, pages 1638–1644. AAAI Press.

Hein, M., Bousquet, O., Hein, M., and Bousquet, O. (2004). Kernels, associated structures and generalizations.

Hernández-Lobato, D. and Hernández-Lobato, J. M. (2013). Learning feature selection dependencies in multi-task learning. In Burges, C. J. C., Bottou, L., Ghahramani, Z., and Weinberger, K. Q., editors, *Advances in Neural Information Processing Systems 26: 27th Annual Conference on Neural Information Processing Systems 2013. Proceedings of a meeting held December 5-8, 2013, Lake Tahoe, Nevada, United States*, pages 746–754.

Hernández-Lobato, D., Hernández-Lobato, J. M., and Ghahramani, Z. (2015). A probabilistic model for dirty multi-task feature selection. In Bach, F. R. and Blei, D. M., editors, *Proceedings of the 32nd International Conference on Machine Learning, ICML 2015, Lille, France, 6-11 July 2015*, volume 37 of *JMLR Workshop and Conference Proceedings*, pages 1073–1082. JMLR.org.

Jacob, L., Bach, F. R., and Vert, J. (2008). Clustered multi-task learning: A convex formulation. In Koller, D., Schuurmans, D., Bengio, Y., and Bottou, L., editors, *Advances in Neural Information Processing Systems 21, Proceedings of the Twenty-Second Annual Conference on Neural Information Processing Systems, Vancouver, British Columbia, Canada, December 8-11, 2008*, pages 745–752. Curran Associates, Inc.

Jaderberg, M., Simonyan, K., Zisserman, A., and Kavukcuoglu, K. (2015). Spatial transformer networks.

Jalali, A., Ravikumar, P., Sanghavi, S., and Ruan, C. (2010). A dirty model for multi-task learning. In Lafferty, J. D., Williams, C. K. I., Shawe-Taylor, J., Zemel, R. S., and Culotta, A., editors, *Advances in Neural Information Processing Systems 23: 24th Annual Conference on Neural Information Processing Systems 2010. Proceedings of a*

meeting held 6-9 December 2010, Vancouver, British Columbia, Canada, pages 964–972. Curran Associates, Inc.

Jawanpuria, P. and Nath, J. S. (2012). A convex feature learning formulation for latent task structure discovery. In *Proceedings of the 29th International Conference on Machine Learning, ICML 2012, Edinburgh, Scotland, UK, June 26 - July 1, 2012*. icml.cc / Omnipress.

Jebara, T. (2004). Multi-task feature and kernel selection for svms. In Brodley, C. E., editor, *Machine Learning, Proceedings of the Twenty-first International Conference (ICML 2004), Banff, Alberta, Canada, July 4-8, 2004*, volume 69 of *ACM International Conference Proceeding Series*. ACM.

Jebara, T. (2011). Multitask sparsity via maximum entropy discrimination. *J. Mach. Learn. Res.*, 12:75–110.

Jeong, J. and Jun, C. (2018). Variable selection and task grouping for multi-task learning. In Guo, Y. and Farooq, F., editors, *Proceedings of the 24th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining, KDD 2018, London, UK, August 19-23, 2018*, pages 1589–1598. ACM.

Kadri, H., Duflos, E., Preux, P., Canu, S., Rakotomamonjy, A., and Audiffren, J. (2016). Operator-valued kernels for learning from functional response data. *J. Mach. Learn. Res.*, 17:20:1–20:54.

Kang, Z., Grauman, K., and Sha, F. (2011). Learning with whom to share in multi-task feature learning. In Getoor, L. and Scheffer, T., editors, *Proceedings of the 28th International Conference on Machine Learning, ICML 2011, Bellevue, Washington, USA, June 28 - July 2, 2011*, pages 521–528. Omnipress.

Kumar, A. and Daumé III, H. (2012). Learning task grouping and overlap in multi-task learning. In *Proceedings of the 29th International Conference on Machine Learning, ICML 2012, Edinburgh, Scotland, UK, June 26 - July 1, 2012*. icml.cc / Omnipress.

Lawrence, N. D. and Platt, J. C. (2004). Learning to learn with the informative vector machine. In Brodley, C. E., editor, *Machine Learning, Proceedings of the Twenty-first International Conference (ICML 2004), Banff, Alberta, Canada, July 4-8, 2004*, volume 69 of *ACM International Conference Proceeding Series*. ACM.

LeCun, Y., Bottou, L., Bengio, Y., and Haffner, P. (1998). Gradient-based learning applied to document recognition. *Proc. IEEE*, 86(11):2278–2324.

Li, Y., Tian, X., Song, M., and Tao, D. (2015). Multi-task proximal support vector machine. *Pattern Recognit.*, 48(10):3249–3257.

Liang, L. and Cherkassky, V. (2008). Connection between SVM+ and multi-task learning. In *Proceedings of the International Joint Conference on Neural Networks, IJCNN 2008, part of the IEEE World Congress on Computational Intelligence, WCCI 2008, Hong Kong, China, June 1-6, 2008*, pages 2048–2054. IEEE.

Liu, H., Palatucci, M., and Zhang, J. (2009). Blockwise coordinate descent procedures for the multi-task lasso, with applications to neural semantic basis discovery. In Danyluk, A. P., Bottou, L., and Littman, M. L., editors, *Proceedings of the 26th*

Annual International Conference on Machine Learning, ICML 2009, Montreal, Quebec, Canada, June 14-18, 2009, volume 382 of *ACM International Conference Proceeding Series*, pages 649–656. ACM.

Long, M. and Wang, J. (2015). Learning multiple tasks with deep relationship networks. *CoRR*, abs/1506.02117.

Loshchilov, I. and Hutter, F. (2019). Decoupled weight decay regularization. In *7th International Conference on Learning Representations, ICLR 2019, New Orleans, LA, USA, May 6-9, 2019*. OpenReview.net.

Lozano, A. C. and Swirszcz, G. (2012). Multi-level lasso for sparse multi-task regression. In *Proceedings of the 29th International Conference on Machine Learning, ICML 2012, Edinburgh, Scotland, UK, June 26 - July 1, 2012*. icml.cc / Omnipress.

Maurer, A. (2006a). Bounds for linear multi-task learning. *J. Mach. Learn. Res.*, 7:117–139.

Maurer, A. (2006b). The rademacher complexity of linear transformation classes. In Lugosi, G. and Simon, H. U., editors, *Learning Theory, 19th Annual Conference on Learning Theory, COLT 2006, Pittsburgh, PA, USA, June 22-25, 2006, Proceedings*, volume 4005 of *Lecture Notes in Computer Science*, pages 65–78. Springer.

Maurer, A. (2009). Transfer bounds for linear feature learning. *Mach. Learn.*, 75(3):327–350.

Maurer, A. and Pontil, M. (2010). K -dimensional coding schemes in hilbert spaces. *IEEE Trans. Inf. Theory*, 56(11):5839–5846.

Maurer, A., Pontil, M., and Romera-Paredes, B. (2013). Sparse coding for multitask and transfer learning. In *Proceedings of the 30th International Conference on Machine Learning, ICML 2013, Atlanta, GA, USA, 16-21 June 2013*, volume 28 of *JMLR Workshop and Conference Proceedings*, pages 343–351. JMLR.org.

Maurer, A., Pontil, M., and Romera-Paredes, B. (2016). The benefit of multitask representation learning. *J. Mach. Learn. Res.*, 17:81:1–81:32.

Micchelli, C. A. and Pontil, M. (2004). Kernels for multi–task learning. In *Advances in Neural Information Processing Systems 17 [Neural Information Processing Systems, NIPS 2004, December 13-18, 2004, Vancouver, British Columbia, Canada]*, pages 921–928.

Micchelli, C. A. and Pontil, M. (2005). On learning vector-valued functions. *Neural Comput.*, 17(1):177–204.

Misra, I., Shrivastava, A., Gupta, A., and Hebert, M. (2016). Cross-stitch networks for multi-task learning. In *2016 IEEE Conference on Computer Vision and Pattern Recognition, CVPR 2016, Las Vegas, NV, USA, June 27-30, 2016*, pages 3994–4003. IEEE Computer Society.

Obozinski, G., Taskar, B., and Jordan, M. (2006). Multi-task feature selection. *Statistics Department, UC Berkeley, Tech. Rep*, 2(2.2):2.

Parameswaran, S. and Weinberger, K. Q. (2010). Large margin multi-task metric learning. In Lafferty, J. D., Williams, C. K. I., Shawe-Taylor, J., Zemel, R. S., and Culotta, A., editors, *Advances in Neural Information Processing Systems 23: 24th Annual Conference on Neural Information Processing Systems 2010. Proceedings of a meeting held 6-9 December 2010, Vancouver, British Columbia, Canada*, pages 1867–1875. Curran Associates, Inc.

Paszke, A., Gross, S., Massa, F., Lerer, A., Bradbury, J., Chanan, G., Killeen, T., Lin, Z., Gimelshein, N., Antiga, L., Desmaison, A., Kopf, A., Yang, E., DeVito, Z., Raison, M., Tejani, A., Chilamkurthy, S., Steiner, B., Fang, L., Bai, J., and Chintala, S. (2019). Pytorch: An imperative style, high-performance deep learning library. In *Advances in Neural Information Processing Systems 32*, pages 8024–8035.

Pong, T. K., Tseng, P., Ji, S., and Ye, J. (2010). Trace norm regularization: Reformulations, algorithms, and multi-task learning. *SIAM J. Optim.*, 20(6):3465–3489.

Ruder, S. (2017). An overview of multi-task learning in deep neural networks. *CoRR*, abs/1706.05098.

Ruder, S., Bingel, J., Augenstein, I., and Søgaard, A. (2017). Sluice networks: Learning what to share between loosely related tasks. *CoRR*, abs/1705.08142.

Ruiz, C., Alaíz, C. M., and Dorronsoro, J. R. (2019). A convex formulation of svm-based multi-task learning. In *HAIS 2019*, volume 11734 of *Lecture Notes in Computer Science*, pages 404–415. Springer.

Ruiz, C., Alaíz, C. M., and Dorronsoro, J. R. (2021a). Adaptive graph laplacian for convex multi-task learning SVM. In *Hybrid Artificial Intelligent Systems - 16th International Conference, HAIS 2021, Bilbao, Spain, September 22-24, 2021, Proceedings*, volume 12886 of *Lecture Notes in Computer Science*, pages 219–230. Springer.

Ruiz, C., Alaíz, C. M., and Dorronsoro, J. R. (2021b). Convex formulation for multi-task l1-, l2-, and ls-svms. *Neurocomputing*, 456:599–608.

Ruiz, C., Alaíz, C. M., and Dorronsoro, J. R. (2022). Convex mtl for neural networks. *Logic Journal of the IGPL*.

Schölkopf, B., Herbrich, R., and Smola, A. J. (2001). A generalized represented theorem. In Helmbold, D. P. and Williamson, R. C., editors, *Computational Learning Theory, 14th Annual Conference on Computational Learning Theory, COLT 2001 and 5th European Conference on Computational Learning Theory, EuroCOLT 2001, Amsterdam, The Netherlands, July 16-19, 2001, Proceedings*, volume 2111 of *Lecture Notes in Computer Science*, pages 416–426. Springer.

Sun, X., Panda, R., Feris, R., and Saenko, K. (2020). Adashare: Learning what to share for efficient deep multi-task learning. In Larochelle, H., Ranzato, M., Hadsell, R., Balcan, M., and Lin, H., editors, *Advances in Neural Information Processing Systems 33: Annual Conference on Neural Information Processing Systems 2020, NeurIPS 2020, December 6-12, 2020, virtual.*

Suykens, J. A. K. and Vandewalle, J. (1999). Least squares support vector machine classifiers. *Neural Process. Lett.*, 9(3):293–300.

Thrun, S. and O'Sullivan, J. (1996). Discovering structure in multiple learning tasks: The TC algorithm. In Saitta, L., editor, *Machine Learning, Proceedings of the Thirteenth International Conference (ICML '96), Bari, Italy, July 3-6, 1996*, pages 489–497. Morgan Kaufmann.

Vapnik, V. (1982). *Estimation of dependences based on empirical data*. Springer Science & Business Media.

Vapnik, V. (2000). *The Nature of Statistical Learning Theory*. Statistics for Engineering and Information Science. Springer.

Vapnik, V. and Izmailov, R. (2015). Learning using privileged information: similarity control and knowledge transfer. *J. Mach. Learn. Res.*, 16:2023–2049.

Vapnik, V. and Vashist, A. (2009). A new learning paradigm: Learning using privileged information. *Neural Networks*, 22(5-6):544–557.

Weinberger, K. Q. and Saul, L. K. (2009). Distance metric learning for large margin nearest neighbor classification. *J. Mach. Learn. Res.*, 10:207–244.

Xiao, H., Rasul, K., and Vollgraf, R. (2017). Fashion-mnist: a novel image dataset for benchmarking machine learning algorithms.

Xu, S., An, X., Qiao, X., and Zhu, L. (2014). Multi-task least-squares support vector machines. *Multim. Tools Appl.*, 71(2):699–715.

Xue, Y., Liao, X., Carin, L., and Krishnapuram, B. (2007). Multi-task learning for classification with dirichlet process priors. *J. Mach. Learn. Res.*, 8:35–63.

Yang, Y. and Hospedales, T. M. (2017a). Deep multi-task representation learning: A tensor factorisation approach. In *5th International Conference on Learning Representations, ICLR 2017, Toulon, France, April 24-26, 2017, Conference Track Proceedings*. OpenReview.net.

Yang, Y. and Hospedales, T. M. (2017b). Trace norm regularised deep multi-task learning. In *5th International Conference on Learning Representations, ICLR 2017, Toulon, France, April 24-26, 2017, Workshop Track Proceedings*. OpenReview.net.

Yu, K., Tresp, V., and Schwaighofer, A. (2005). Learning gaussian processes from multiple tasks. In Raedt, L. D. and Wrobel, S., editors, *Machine Learning, Proceedings of the Twenty-Second International Conference (ICML 2005), Bonn, Germany, August 7-11, 2005*, volume 119 of *ACM International Conference Proceeding Series*, pages 1012–1019. ACM.

Zhang, Y. and Yeung, D. (2010). A convex formulation for learning task relationships in multi-task learning. In Grünwald, P. and Spirtes, P., editors, *UAI 2010, Proceedings of the Twenty-Sixth Conference on Uncertainty in Artificial Intelligence, Catalina Island, CA, USA, July 8-11, 2010*, pages 733–442. AUAI Press.

Zhang, Y. and Yeung, D. (2013). A regularization approach to learning task relationships in multitask learning. *ACM Trans. Knowl. Discov. Data*, 8(3):12:1–12:31.

Zhang, Y., Yeung, D., and Xu, Q. (2010). Probabilistic multi-task feature selection. In Lafferty, J. D., Williams, C. K. I., Shawe-Taylor, J., Zemel, R. S., and Culotta, A., editors, *Advances in Neural Information Processing Systems 23: 24th Annual Conference on Neural Information Processing Systems 2010. Proceedings of a meeting held 6-9 December 2010, Vancouver, British Columbia, Canada*, pages 2559–2567. Curran Associates, Inc.

Zhou, J., Chen, J., and Ye, J. (2011). Clustered multi-task learning via alternating structure optimization. In Shawe-Taylor, J., Zemel, R. S., Bartlett, P. L., Pereira, F. C. N., and Weinberger, K. Q., editors, *Advances in Neural Information Processing Systems 24: 25th Annual Conference on Neural Information Processing Systems 2011. Proceedings of a meeting held 12-14 December 2011, Granada, Spain*, pages 702–710.

Zweig, A. and Weinshall, D. (2013). Hierarchical regularization cascade for joint learning. In *Proceedings of the 30th International Conference on Machine Learning, ICML 2013, Atlanta, GA, USA, 16-21 June 2013*, volume 28 of *JMLR Workshop and Conference Proceedings*, pages 37–45. JMLR.org.