UNIVERSIDAD AUTÓNOMA DE MADRID

# Advanced Kernel Methods for Multi-Task Learning

by

Carlos Ruiz Pastor

A thesis submitted in partial fulfillment for the
degree of Doctor of Philosophy

in the
Escuela Politécnica Superior
Computer Science Department

under the supervision of José R. Dorronsoro Ibero and Carlos M. Alaíz Gudín

December 2022

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

Aristotle.

# *Abstract*

**Advanced Kernel Methods for Multi-Task Learning**

by Carlos Ruiz Pastor

Machine Learning (ML), which tries to automatize the process of learning, has a great influence in our current society. The ML algorithms try to infer general patterns from data, which can then be applicable to unseen data. These algorithms, such as the Support Vector Machine (SVM) or Neural Network (NN), are present in many daily situations and have strongly impacted multiple areas such as engineering or advertising, among many others. Multi-Task Learning (MTL) is a field of ML that considers learning different tasks jointly to improve the learning process. This is the natural way of learning for humans: we do not learn each task in an isolated manner, but there exist tasks are better learned together. The goal in MTL is developing strategies to mimic this behaviour, where learning different tasks jointly offers an advantage.

This thesis begins with a presentation of some basic concepts and definitions that we will use in the rest of this work.

After that, a theoretical motivation for MTL is given, and some of the most relevant works on MTL are reviewed. A taxonomy for these methods is proposed, where three categories are considered: feature-based, parameter-based and combination-based strategies. Different ML algorithms, depending on their characteristics, can adopt more easily one of these strategies. The feature-based strategies are more natural for NNs, while the kernel methods, such as the SVM, present a more rigid framework, and the combination-based strategies are better suited for them.

Considering the combination-based strategies, a new convex formulation is proposed: in each task we consider a convex combination of a common and task-specific part as the model. This formulation offers some nice properties, such as better interpretability of the models, or the possibility to dismiss the common or task-specific parts with a particular choice of the hyperparameters. This approach is applied to kernel methods, in particular, the L1, L2 and LS-SVM with this convex MTL formulation are proposed, and the solutions for the corresponding training problems can be obtained with standard SVM solvers. One natural alternative, which considers the convex combination of pre-trained common and task-specific models, is also described. In multiple experiments it is observed that the kernel methods with a convex MTL formulation obtains results better than those considering just a common model, task-specific ones or the convex combination of pre-trained models. As a real world application, the prediction of solar and wind energy is presented using these models, where our proposal outperforms or ties with the competition, which consider either a common model or task-specific ones.

Following this formulation, a MTL proposal for NNs is made, where a convex combination of a common and a task-specific networks is used. These models can also be trained with standard optimization techniques for NNs. In experiments with four image datasets, it is shown that the results of this proposal are better than using standard approaches, such as sharing the weights in the hidden layers.

Another approach with Graph Laplacian regularization, which we categorize as a parameter-based strategy is proposed. Here, considering a GL regularization, where the tasks are interpreted as nodes in a graph, the pairwise distances between the task models are penalized. Here, the adjacency matrix of the graph defines the weights for the distances. A new formulation to apply this regularization in kernel spaces is developed, and the GL approach is applied to the L1, L2 and LS-SVM. It is exemplified with experiments also that this approach can obtain competitive results. Moreover, an algorithm to automatically learn the adjacency matrix from the data is proposed. Examples of the advantages of taking this approach are given using experiments with synthetic and real data.

The thesis ends with some general conclusions and presents ideas for future research.

# *Resumen*

Small.

# *Acknowledgements*

.

# Contents

# List of Figures

# List of Tables

# Abbreviations

**CTL** Common-Task Learning.

**CV** Cross Validation.

**DNNs** Deep Neural Networks.

**ECMWF** European Centre for Medium-Range Weather Forecasts.

**ERM** Empirical Risk Minimization.

**GL** Graph Laplacian.

**GP** Gaussian Process.

**GPs** Gaussian Processes.

**iid** independent and identically distributed.

**ITL** Independent-Task Learning.

**KKT** Karush-Kuhn-Tucker.

**LTL** Learning to Learn.

**LUPI** Learning Using Privileged Information.

**MAE** Mean Absolute Error.

**ML** Machine Learning.

**MSE** Mean Squared Error.

**MT** Multi-Task.

**MTL** Multi-Task Learning.

**NN** Neural Network.

**NNs** Neural Networks.

**NWP** Numerical Weather Prediction.

**PV** Photovoltaic.

**RKHS** Reproducing Kernel Hilbert Space.

**RKHSs** Reproducing Kernel Hilbert Spaces.

**SGD** Stochastic Gradient Descent.

**SMO** Sequential Minimal Optimization.

**SRM** Structural Risk Minimization.

**STL** Single-Task Learning.

**SVM** Support Vector Machine.

**SVMs** Support Vector Machines.

**SVR** Support Vector Regressor.

**VC** Vapnik-Chervonenkis.

*To my family*

Chapter 1

# Foundations and Concepts

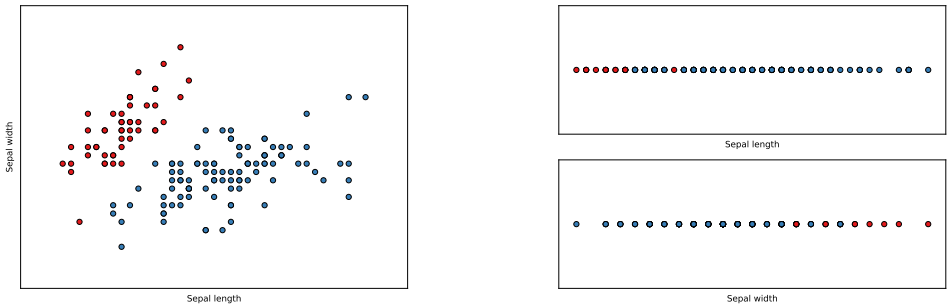
This chapter presents some concepts and definitions that we will use in the rest of this work. First, in Section 1.1 we introduce the kernel functions, the kernel trick and the Reproducing Kernel Hilbert Space (RKHS). Next, in Section 1.2 we define loss function, empirical and expected risks. Here, we also describe the concept of regularization, which is present in the definition of regularized risk. In this context we also present the Representer Theorem, which gives a closed form solution of regularized risk problems when using kernels.

Related to the concept of regularization, we give in Section 1.3 a brief introduction to learning theory, where we present the Vapnik-Chervonenkis (VC)-dimension. Then, in Section 1.4 we present the optimization theory, and we give the definitions and results used to obtain the solutions of optimization problems. Here we define the primal and dual problems, and the relation between them. In particular, we are interested on finding the solutions of the optimization problem related to the training of the Support Vector Machine (SVM). In Section 1.5 we present the SVM, describing the primal and dual formulations. Finally, in Section 1.6 we describe two popular SVM variants, the L2 and LS-SVM.

## 1.1 Kernels and Implicit Features

Kernels are central in Machine Learning (ML), and some of the most successful methods use them, such as SVMs or Gaussian Processes (GPs). The main advantage of using kernels, as we will see, is that we can send our data to a high, even infinite, dimensional space, where the problems of interest, regression or classification, are expected to be easier. Moreover, this transformation is made implicitly by exploiting the reproducing property of the kernels. This is named the kernel trick, which we will explain more carefully later in this section.

As said, by using kernels, we change our feature space, from a finite-dimensional one to a much larger one, with possibly infinite dimensions. Although this might ease the problem of learning a function that explains our data, we also can encounter some problems. One is the curse of dimensionality, that is, the cost of finding a solution to optimization problems typically scales super-linearly with the dimension of our data (Chapelle, 2007). Other problem is overfitting; when the data is placed in a high-dimensional space the hypotheses that can explain the data are more expressive, and if they fit too exactly the




(A) Simplified iris example. (B) Projections of simplified iris example.

FIGURE 1.1: Representations of problem iris.

training data, they could not generalize well to new data. However, as we will see later in this chapter, these problems can be solved.

### 1.1.1 Learning in Feature Spaces

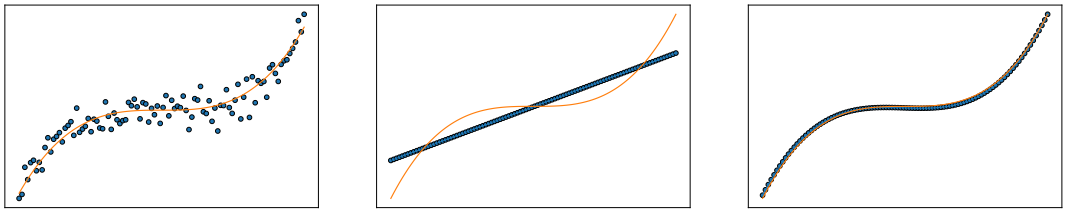
In ML the choice of the feature space is crucial. If it is too small or too poor, it is not possible to find a good solution to our problem. Consider the classification setting; here, when the feature space is low-dimensional, the classes are more overlapped and finding a decision boundary is more challenging. Take for example the popular iris dataset, where the goal is to discriminate between three species of iris flowers: *setosa*, *virginica* and *versicolor*. To do this, 50 samples from each species are gathered, and four features are measured for each sample: the length and width of the petals and sepals. Now, for illustration purposes, we take a simplified version in which the goal is to discriminate the *setosa* class from the rest, and where we consider only two features: the length and width of the sepals. In Figure 1.1a we represent the data corresponding to this simplified version of the iris problem, and we can observe that the two clouds of points are easily separable by a linear function. However, if, instead of the two-dimensional space, we consider just one feature, the problem is no longer separable, as it can be seen in the example of Figure 1.1b.

For an example of how enlarging the feature space can lead to better solutions, consider a regression problem where we take 100 one-dimensional points linearly distributed in the feature space, in this case $[-2, 2]$; then we compute the target values as $y_i = x_i^3 + \epsilon_i$, where $\epsilon_i \sim N(0, 1)$; this problem is illustrated in Figure 1.2a. If we consider a linear regression model with the original feature space, it will not be able to approximate well the function $f(x) = x^3$, and the solution is depicted in Figure 1.2b. However, if we enlarge the feature space using the transformation $\phi(\cdot)$ defined as

$$\begin{aligned} \phi : \mathcal{X} &\to \mathcal{X}^3 \\ x &\to (x, x^2, x^3), \end{aligned}$$

then a linear regression model in this new space can find a good approximation, as the one shown in Figure 1.2c.

This kind of results is one of the motivations for using kernels. As we will see later, linear models, such as Linear Regression or SVMs, have a dual formulation, which does not depend directly on the features $x_i$, but on the inner products $\langle x_i, x_j \rangle$. Consider now



(A) Cubic regression problem. The underlying function, $f(x) = x^3$ is shown in orange, and the target values are the dots in blue.

(B) Predictions with a linear feature space, including only $x$. The real function $f(x) = x^3$ is shown in orange and the predictions are the blue dots.

(C) Predictions with an extended feature space, including $x, x^2$ and $x^3$. The real function $f(x) = x^3$ is shown in orange and the predictions are the blue dots.

FIGURE 1.2: Cubic regression problem example and two regression functions: one linear and one cubic.

that we have the feature space $\mathcal{X} = \mathbb{R}^2$, and we define the following function:

$$\begin{aligned} k: \quad \mathbb{R}^2 \times \mathbb{R}^2 &\to \mathbb{R} \\ ((x_1, x_2),(y_1, y_2)) &\to ((x_1, x_2)^\intercal (y_1, y_2) + c)^2. \end{aligned}$$

Now, we can observe the following fact:

$$\begin{aligned} &((x_1, x_2)^\intercal (y_1, y_2) + c)^2 \\ &= (x_1 y_1 + x_2 y_2 + c)^2 \\ &= x_1^2 y_1^2 + 2x_1 y_1 x_2 y_1 + x_2^2 y_2^2 + 2cx_1 y_1 + 2cx_2 y_2 + c^2 \\ &= \left\langle (c, \sqrt{2c}x_1, \sqrt{2c}x_2, \sqrt{2}x_1 x_2, x_1^2, x_2^2), (c, \sqrt{2c}y_1, \sqrt{2c}y_2, \sqrt{2}y_1 y_2, y_1^2, y_2^2) \right\rangle, \end{aligned}$$

where $\langle \cdot, \cdot \rangle$ indicates the inner product. Therefore, if we have a model that depends on the inner products of the data in our feature space $x, y \in \mathcal{X}$, and we define these inner products as $k(x, y) = (x^\intercal y + c)^2$, we are implicitly enlarging the feature space with the transformation $\psi$, defined as

$$\begin{aligned} \psi: \quad \mathbb{R}^2 &\to \mathbb{R}^6 \\ (x_1, x_2) &\to (c, \sqrt{2c}x_1, \sqrt{2c}x_2, \sqrt{2}x_1 x_2, x_1^2, x_2^2), \end{aligned}$$

such that $k(x, y) = \langle \psi(x), \psi(y) \rangle$. Here, the function $k(\cdot, \cdot)$ is a kernel function; in particular, this is a polynomial kernel, and $\psi(\cdot)$ is its associated feature transformation. We remark that in the dual formulation of linear models, which will be explained in detail later, all we need is to define a proper kernel function that must meet some requirements, and, as we will see, it has an associated feature transformation function; thus, by defining the kernel function, we are implicitly enlarging our feature space.

### 1.1.2 The Reproducing Kernel Map

To make use of kernels and to ensure that they have a corresponding feature transformation function that extends our feature space, we have to define some concepts and give some previous results. Here we follow the work of Schölkopf and Smola (2002). In the first place, we define the inner product.

**Definition 1.1** (Inner Product)**.** An inner product on a vector space $\mathcal{V}$ is a map

$$\langle \cdot, \cdot \rangle : \mathcal{V} \times \mathcal{V} \to \mathbb{R},$$

such that for all $u, v, w \in \mathcal{V}$ and $a, b \in \mathbb{R}$ satisfies:

1. $\langle u, av + bw \rangle = a \langle u, v \rangle + b \langle u, w \rangle$ (linear);
2. $\langle u, v \rangle = \langle v, u \rangle$ (symmetric);
3. $\langle u, u \rangle \geq 0$ and $\langle u, u \rangle = 0 \implies u = 0$ (positive definite).

Then, we define a positive definite kernel function, which has some similarities with the inner product.

**Definition 1.2** (Positive Definite Kernel)**.** Given a non-empty set $\mathcal{X}$, a symmetric function

$$k : \mathcal{X} \times \mathcal{X} \to \mathbb{R}$$

is a positive definite kernel if for any set $x_1, \ldots, x_n \in \mathcal{X}$ and $c_1, \ldots, c_n \in \mathbb{R}$,

$$\sum_{i=1}^{n} \sum_{j=1}^{n} c_i c_j k(x_i, x_j) \geq 0,$$

and if $\sum_{i=1}^{n} \sum_{j=1}^{n} c_i c_j k(x_i, x_j) = 0$, then $c_1, \ldots, c_n = 0$.

It can be seen that the inner product and the kernel function are symmetric and positive-definite. The connection between both goes further, and we will see it below. For simplicity, we will say positive kernel or just kernel instead of positive definite kernel. We can characterize a kernel also using finite sets and the corresponding matrices built using such kernel. To do this we need to define the Gram (or kernel) matrix and positive definite matrix first.

**Definition 1.3** (Gram Matrix)**.** Given a symmetric function

$$k : \mathcal{X} \times \mathcal{X} \to \mathbb{R}$$

and patterns $x_1, \ldots, x_n \in \mathcal{X}$, the matrix $K$ with elements $K_{ij} = k(x_i, x_j)$, with $i, j = 1, \ldots, n$, is called the Gram matrix.

**Definition 1.4** (Positive Definite Matrix)**.** A matrix $K \in \mathbb{R}^{n \times n}$ is positive definite if $\sum_{i=1}^{n} \sum_{i=1}^{n} K_{ij} c_i, c_j \geq 0$ for any $c_1, \ldots, c_n \in \mathbb{R}$ and if $\sum_{i=1}^{n} \sum_{i=1}^{n} K_{ij} c_i, c_j = 0$, then $c_1, \ldots, c_n = 0$.

Then, it is easy to characterize a kernel function by its kernel matrices, as stated in the following lemma (Schölkopf and Smola, 2002).

**Lemma 1.5.** *Given a non-empty set $\mathcal{X}$, a symmetric function*

$$k : \mathcal{X} \times \mathcal{X} \to \mathbb{R}$$

*is a positive definite kernel if for any set $x_1, \ldots, x_n \in \mathcal{X}$, the corresponding Gram matrix is positive definite.*

As illustrated in the previous subsection, we are interested in using kernels to implicitly transform our data. Consider the map from $\mathcal{X}$, a non-empty set, into the space of functions $f : \mathcal{X} \to \mathbb{R}$, which we denote as $\mathbb{R}^{\mathcal{X}}$, defined as

$$\begin{aligned} \phi : \mathcal{X} &\to \mathbb{R}^{\mathcal{X}} \\ x &\to k(\cdot, x), \end{aligned}$$

where $k$ is a positive definite kernel. Here, we are transforming each point $x$ into a function that assigns the value $\phi(\tilde{x}) = k(x, \tilde{x})$, a function on $x$, to every $\tilde{x} \in \mathcal{X}$. That is, in the new space, $\tilde{x}$ is defined by its similarity, expressed by $k(x, \tilde{x})$, to every other point $x \in \mathcal{X}$. This transformation seems difficult to work with because our new features are functions. However, we can define a feature space associated with $\phi$, with an inner product associated with the kernel function $k(\cdot, \cdot)$. To do this, we follow three steps: we define a vector space on the image of $\phi$, we define an inner product in such space, and we show that this inner product satisfies $\langle \phi(\tilde{x}), \phi(x) \rangle = k(\tilde{x}, x)$. We begin by considering all linear combinations of images of $\phi$, that is, the functions

$$f(\cdot) = \sum_{i=1}^{n} \alpha_i \phi(x_i)(\cdot) = \sum_{i=1}^{n} \alpha_i k(\cdot, x_i),$$

with $\alpha_1, \ldots, \alpha_n \in \mathbb{R}$. The set of such functions is a vector space $\mathcal{V}_\phi \subset \mathbb{R}^{\mathcal{X}}$, since given other function

$$g(\cdot) = \sum_{i=1}^{m} \beta_i \phi(\tilde{x}_i)(\cdot) = \sum_{i=1}^{n} \beta_i k(\cdot, \tilde{x}_i),$$

with $\beta_1, \ldots, \beta_m \in \mathbb{R}$, we have that $f(\cdot) + g(\cdot) \in \mathcal{V}_\phi$, and for any $a \in \mathbb{R}$, $af(\cdot) \in \mathcal{V}_\phi$. Now, we can define an inner product in this vector space as

$$\langle f(\cdot), g(\cdot) \rangle = \sum_{i=1}^{n} \sum_{i=1}^{m} \alpha_i \beta_j k(x_i, \tilde{x}_j),$$

and we have to check that it satisfies the properties of an inner product. It is easy to see its linearity, since it is a sum, and the symmetry property holds from the symmetry of the kernel function. Finally, to check that it is positive definite we observe that because $k(\cdot, \cdot)$ is positive definite,

$$\langle f(\cdot), f(\cdot) \rangle = \sum_{i=1}^{n} \sum_{j=1}^{n} \alpha_i \alpha_j k(x_i, x_j) \geq 0$$

for any $x_1, \ldots, x_n \in \mathcal{X}$ and $\alpha_1, \ldots, \alpha_n \in \mathbb{R}$, and if $\langle f(\cdot), f(\cdot) \rangle = 0$, then $\alpha_1, \ldots, \alpha_n = 0$, hence $f(\cdot) = 0$. This inner product is defined for any pair of functions $f(\cdot), g(\cdot) \in \mathcal{V}_\phi$. In particular, consider $g(\cdot) = \phi(\tilde{x})(\cdot) = k(\cdot, \tilde{x})$; then, from the definitions, we have that

$$\langle f(\cdot), k(\cdot, \tilde{x}) \rangle = \left\langle \sum_{i=1}^{n} \alpha_i \phi(x_i)(\cdot), \phi(\tilde{x})(\cdot) \right\rangle = \sum_{i=1}^{n} \alpha_i k(x_i, \tilde{x}) = f(\tilde{x}).$$

That is, $k(\cdot, \tilde{x})$ is the representative of the evaluation on $\tilde{x}$ since to evaluate any function $f$ on $\tilde{x}$ is just necessary to compute $\langle f, k(\cdot, \tilde{x}) \rangle$. Moreover, we observe that with $f(\cdot) = \phi(x)(\cdot)$,

$$\langle \phi(x)(\cdot), \phi(\tilde{x})(\cdot) \rangle = \langle k(\cdot, x), k(\cdot, \tilde{x}) \rangle = k(x, \tilde{x}).$$

This is the reproducing property of the kernel, and it defines an easy way of computing inner products in the feature space of functions $\mathbb{R}^{\mathcal{X}}$: to compute the inner product between $\phi(x)$ and $\phi(\tilde{x})$, we just need to compute the kernel value $k(x, \tilde{x})$.

### 1.1.3 Reproducing Kernel Hilbert Spaces

Kernels, with their reproducing property, have a close connection with a particular class of Hilbert spaces, which are named Reproducing Kernel Hilbert Spaces (RKHSs). Following the work of Schölkopf and Smola (2002), first we give the definition of a Hilbert space.

**Definition 1.6** (Hilbert Space)**.** A Hilbert space is a vector space $\mathcal{H}$ with an inner product $\langle \cdot, \cdot \rangle$, such that under the norm defined by $\|u\| = \sqrt{\langle u, u \rangle}$, $u \in \mathcal{H}$, it is a complete metric space.

An example of Hilbert space is $\mathbb{R}^d$ with the inner product defined as $\langle u, v \rangle = u^\intercal v$. We are interested in drawing the connection between kernels and the RKHSs, which is given in the following definition.

**Definition 1.7** (RKHS)**.** Given a non-empty set $\mathcal{X}$, a Hilbert space $\mathcal{H}$ of functions $f : \mathcal{X} \to \mathbb{R}$ with an inner product $\langle \cdot, \cdot \rangle$ is an RKHS if there exists a symmetric function $k : \mathcal{X} \times \mathcal{X} \to \mathbb{R}$ such that $\langle f, k(x, \cdot) \rangle = f(x)$ for all $f \in \mathcal{H}$, and $\overline{\text{span}\,\{k(\cdot, x), x \in \mathcal{X}\}} = \mathcal{H}$. Such function $k$ is called a reproducing kernel.

Here, $\text{span}\,\{k(\cdot, x), x \in \mathcal{X}\}$ is the span of the functions $k(\cdot, x)$, that is, functions which are defined as

$$f(\cdot) = \sum_{i=1}^{n} \alpha_i \phi(x_i)(\cdot) = \sum_{i=1}^{n} \alpha_i k(\cdot, x_i),$$

and $\overline{\mathcal{A}}$ is the completion of the set $\mathcal{A}$, which include the set itself and the limits of all its Cauchy sequences (Kelley, 2017). Other way to characterize an RKHS is as a Hilbert space $\mathcal{H}$ with continuous evaluation functionals, i.e. those spaces where the maps

$$\begin{aligned} E_x : \mathcal{H} &\to \mathbb{R} \\ f &\to f(x) \end{aligned}$$

are continuous. In this case, according to the Riesz Theorem (Whittaker, 1991), for every $x \in \mathcal{X}$ there exists a single $g_x \in \mathcal{H}$ such that

$$E_x(f) = \langle f, g_x \rangle \,, \; \forall f \in \mathcal{H}.$$

In particular, for $f = g_{\tilde{x}}$ we have $E_x(g_{\tilde{x}}) = g_{\tilde{x}}(x) = \langle g_x, g_{\tilde{x}} \rangle$.

In fact, if we define a function $k(x, \tilde{x}) = \langle g_x, g_{\tilde{x}} \rangle$, then we can see that it is a reproducing kernel. It is symmetric because of the symmetry of the inner product, and definite positive because the inner product is also positive definite. The reproducing property holds from the definitions if we consider $k(\cdot, x) = g_x$. Observe that, given an RKHS, which we have defined as a Hilbert space with continuous evaluation functionals, there exists a unique function that is the reproducing kernel for such space. This uniqueness of the kernel function for each RKHS is a consequence of the uniqueness of the evaluation representatives in the Riesz Theorem (Riesz and Nagy, 2012).

It is also possible to prove the converse, that is, given a positive definite kernel function, there exists an associated RKHS. To do that we need to generate a Hilbert

space from a kernel function, which is done by considering its associated integral operator, whose properties are expressed in the Mercer theorem (Schölkopf and Smola, 2002) given below. For this theorem we use some concepts of measure theory. Given a non-empty set $\mathcal{X}$, we consider a measurable space $(\mathcal{X}, \mu)$ where $\mu$ is the measure. The expression "almost everywhere" means everywhere except in a subset $\mathcal{S} \subset \mathcal{X}$ such that $\mu(\mathcal{S}) = 0$. We also use the notation $L_2(\mathcal{X})$ for the functions $f : \mathcal{X} \to \mathbb{R}$ that are measurable and square-integrable, that is,

$$\int_{\mathcal{X}} (f(x))^2 d\mu(x) < \infty.$$

**Definition 1.8** (Mercer Kernel)**.** Let $k : \mathcal{X} \times \mathcal{X} \to \mathbb{R}$ a symmetric continuous function, such that the integral operator

$$\begin{aligned} T_k : L_2(\mathcal{X}) &\to L_2(\mathcal{X}) \\ f &\to \int_{\mathcal{X}} k(x, \tilde{x}) f(x) d\mu(\tilde{x}) \end{aligned}$$

is positive definite; that is, for any $f \in L_2(\mathcal{X})$,

$$\int_{\mathcal{X} \times \mathcal{X}} k(x, \tilde{x}) f(x) f(\tilde{x}) d\mu(x) d\mu(\tilde{x}) \geq 0;$$

then $k$ is a Mercer kernel.

**Theorem 1.9** (Mercer Theorem)**.** *Let $k$ be a Mercer kernel and consider the normalized eigenfunctions $\psi_i \in L_2(\mathcal{X})$ and associated eigenvalues $\lambda_i$ of the operator $T_k$, i.e.*

$$T_k(\psi_i) = \lambda_i \psi_i;$$

*then $\lambda_i > 0$ and $k(x, \tilde{x}) = \sum_{i=1}^{\infty} \lambda_i \psi_i(x) \psi_i(\tilde{x})$ almost everywhere, where the eigenvalues $\lambda_i$ are sorted in non-increasing order.*

With this result, we can construct a Hilbert space whose reproducing kernel is $k$. Consider the following feature transformation

$$\begin{aligned} \phi : \mathcal{X} &\to l_2 \\ x &\to \left( \sqrt{\lambda_1} \psi_1(x), \sqrt{\lambda_2} \psi_2(x), \ldots \right), \end{aligned}$$

where $l_2$ is the space of all sequences $(x_n)_{n \in \mathbb{N}}$ such that

$$\sum_{n \in \mathbb{N}} (x_n)^2 < \infty.$$

In this space, the inner product between $\phi(x)$ and $\phi(\tilde{x})$ is defined as

$$\langle \phi(x), \phi(\tilde{x}) \rangle = \sum_{i=1}^{\infty} \lambda_i \psi_i(x) \psi_i(\tilde{x}),$$

which, by the Mercer theorem, is equivalent to saying that $\langle \phi(x), \phi(\tilde{x}) \rangle = k(x, \tilde{x})$.

Therefore, we have seen that there is a one-to-one correspondence between positive definite kernels and RKHSs:

- Given an RKHS there is a single kernel function that has the reproducing property in that space.

- Also, given a positive definite kernel $k$, there exists an associated RKHS such that $k$ is its reproducing kernel.

## 1.2 Risk and Regularization

To automatize the learning process, which is the goal of ML, it is necessary to define a criterion to measure the performance of the model functions $f : \mathcal{X} \to \mathcal{Y}$ estimated from the data. At each particular sample $(x_i, y_i)$ we can give a metric of how close we are to our desired goal; this is the selected loss function. If we have a classification problem, we may want to minimize the number of incorrectly classified patterns, but this quantity is not continuous and not easy to work with. In regression problems, we would like to minimize the distance between the regression function and the training target values, but there are also multiple ways to define this distance.

Anyway, independently of the chosen loss function, the data that we use have been sampled, and we do not know its distribution. There may be some samples more important than others, because they belong to an area of high probability, according to this unknown distribution. Getting a bad result, as defined by the loss function, in a sample that is an outlier of the distribution is not that relevant, because few data are expected to be sampled in its neighborhood. To account for this, we define the expected risk, which is the expectation of the loss function value over the space $\mathcal{X} \times \mathcal{Y}$, and we would like to minimize this. Nevertheless, since the distribution is unknown, we instead minimize the empirical risk, that is, we use as a proxy the average of the loss function over a particular sample of data points.

When minimizing the empirical risk instead of the expected one, we might find that we are too influenced by the particular sample that we have been given. If the training data is too scarce, or also if we have some prior knowledge that we want to incorporate to the learning process, the regularization of the models can be helpful, as we will later describe.

Moreover, when we want to minimize the regularized risk using linear models, there is a result that describes the solution in terms of the training data, which is crucial for kernel methods, particularly when the linear model is built in an infinite dimensional space.

### 1.2.1 Loss Functions

Given a feature space $\mathcal{X}$ and a target space $\mathcal{Y}$, the criterion that we use to measure the performance of our estimates $f : \mathcal{X} \to \mathcal{Y}$ is defined by the loss functions. Here, we give a general definition of a loss function and present the losses that we will use in the rest of this work.

**Definition 1.10** (Loss Function)**.** Denote $(y, f(x))$ as the pairs consisting on an observation $y \in \mathcal{Y}$ and a prediction $f(x) \in \mathcal{Y}$, with $x \in \mathcal{X}$ an element of the feature space; then any map

$$\begin{array}{rcl} \ell : \mathcal{Y} \times \mathcal{Y} & \to & [0, \infty) \\ (y, f(x)) & \to & \ell(y, f(x)) \end{array}$$

such that $\ell(y, y) = 0$ for all $y \in \mathcal{Y}$ is a loss function.

Observe that $\ell(\cdot, \cdot)$ is a non-negative function and the perfect prediction always achieves its minimum value.

In a binary classification problem, if we want to minimize the number of misclassified patterns, we use the *accuracy* loss, which, with the labels $\{-1, 1\}$ for the classes, we can define as

$$\ell(y, f(x)) = \mathbf{1}_{y \neq f(x)} = \begin{cases} 0, & y = f(x), \\ 1, & y \neq f(x), \end{cases} \tag{1.1}$$

where $\mathbf{1}$ is the indicator function. However, this function only takes into account whether the example is correctly classified or not, but we might also want to take into account the confidence that we have in this decision. If we use the labels $\{-1, 1\}$ to denote the two classes, then we can look at the number $yf(x)$ to check if the decision is correct: only if $yf(x) > 0$ is the pattern correctly classified. That is, we are looking at a single value to determine the success of our decision, and we can also use it to define a confidence. For example, we can consider that in those patterns where $0 < yf(x) < 1$, although correctly classified, the confidence that we have to make such a decision is not enough, and only when $yf(x) \geq 1$ we consider that it is a correct classification. We can interpret this as a confidence margin, and the loss function that implements this is called *soft margin*, or also *hinge* loss, which is the loss function used in the SVM and it is defined as

$$\ell(y, f(x)) = [1 - yf(x)]_+ = \begin{cases} 0, & yf(x) \geq 1, \\ 1 - yf(x), & yf(x) < 1. \end{cases} \tag{1.2}$$

This is a continuous function where we can observe two characteristics. One is a confidence margin where we consider that only those instances such that $yf(x) > 1$ do not require a penalization. The other, is that the penalization grows linearly as $yf(x)$ decreases. A modification of the hinge loss considers the same margin, but the penalization for the misclassified patterns grows as a quadratic function. This is the *squared hinge* loss, whose expression is

$$\ell(y, f(x)) = [1 - yf(x)]_+^2 = \begin{cases} 0, & yf(x) \geq 1, \\ (1 - yf(x))^2, & yf(x) < 1. \end{cases} \tag{1.3}$$

Finally, another commonly used function is the *logistic loss*, where the classes are labeled as $\{0, 1\}$. With this loss, we model $P(y = 1|x)$, that is, the posterior probability of $x$ being a pattern from class 1, by using the sigmoid function $s(\cdot)$ as

$$s(f(x)) = \frac{1}{1 + \exp(-f(x))}.$$

Here, although $f(x) \in \mathbb{R}$, $s(f(x)) \in (0, 1)$, so we can see it as the posterior probability of $x$ being from class 1. Then, the loss function is defined as

$$\ell(y, f(x)) = -y \log(s(f(x))) - (1 - y) \log(1 - s(f(x))), \tag{1.4}$$

which, using $s(f(x)) = P(y = 1|x)$, can be interpreted as the negative logarithm of the likelihood of the observation. Here, given for example a pattern from class $y = 1$, only the first term is active, and the larger $f(x)$ is, the closer $s(f(x))$ gets to 1, so the penalization is smaller. In the binary classification scenario, the *logistic* loss resembles the *hinge* one.

The formula (1.4) also looks like the definition of entropy, so it is also named *cross-entropy* loss, which, is the negative logarithm of the likelihood. With the entropy interpretation, this loss is easily adaptable to the multi-class case with classes $\{1, \ldots, C\}$. Consider the one-hot encoding vector $\boldsymbol{y}$ of length $C$, such that $\boldsymbol{y}_c = 1$ if $y = c$ and

$\boldsymbol{y}_j = 0$ otherwise, for $j, c \in \{1, \ldots, C\}$ and $j \neq c$. Now, we can model the probabilities $P(y = c|x) = f_c(x)$ with $c \in \{1, \ldots, C\}$ and form the corresponding vector $\boldsymbol{f}(x)$; then, the multi-class *cross-entropy* loss is defined as

$$\ell(\boldsymbol{y}, f(x)) = -\sum_{c=1}^{C} \boldsymbol{y}_i \log\left(s(\boldsymbol{f}_i(x))\right).$$

For the other loss functions presented there are also proposals for extending them to the multi-class case, but we will not use them in this work.

In regression problems, given a pair $(x, y) \in \mathcal{X} \times \mathcal{Y}$, the goal is to minimize the distance between the observed target $y$ and our prediction $f(x)$; then, a natural loss is the *absolute* loss function, defined as

$$\ell(y, f(x)) = |y - f(x)|\,.$$

Here, any mistake is penalized, but there might be some cases when the targets are noisy, so we would like to have some room for small errors. Concretely, if $f(\cdot)$ is the perfect regression function, the target values might be $y = f(x) + z$ with $z$ being some random variable, for example, $z \sim N(0, \sigma)$; then, it might be sensible not to penalize those errors that can be explained by this noise. This is done in the $\epsilon$-*insensitive* loss function, which is expressed as

$$\ell(y, f(x)) = [y - f(x)]_\epsilon = \begin{cases} 0, & |y - f(x)| \leq \epsilon, \\ |y - f(x)| - \epsilon, & |y - f(x)| > \epsilon. \end{cases} \tag{1.5}$$

Here, $\epsilon$ is a hyperparameter and has to be selected for each specific problem. Also, observe that with $\epsilon = 0$, we have the *absolute* loss. The regression losses presented are based on the absolute value $|y - f(x)|$, thus, they are not differentiable at every point. One alternative is to use the squared value of the distance, as expressed in the *squared* loss:

$$\ell(y, f(x)) = (y - f(x))^2.$$

The *squared* loss is also easily adaptable to multi-target regression problems. Consider the case with $K$ targets for each feature vector $x$, that is, the $\boldsymbol{y}$ are vectors of length $K$, and we produce the corresponding vector of predictions $\boldsymbol{f}(x)$; then, the multi-target *squared* loss is

$$\ell(y, f(x)) = \|\boldsymbol{y} - \boldsymbol{f}(x)\|^2\,.$$

The $\epsilon$-insensitive loss can also be extended to use the squared value of the distance. The *squared* $\epsilon$-*insensitive* loss is defined as

$$\ell(y, f(x)) = [y - f(x)]_\epsilon^2 = \begin{cases} 0, & (y - f(x))^2 \leq \epsilon, \\ (y - f(x))^2 - \epsilon, & (y - f(x))^2 > \epsilon. \end{cases} \tag{1.6}$$

### 1.2.2 Empirical and Expected Risk

In ML, we have a feature space $\mathcal{X}$ and a target (or label) space $\mathcal{Y}$, and we believe that there is a function $f : \mathcal{X} \to \mathcal{Y}$; if we have a good estimate of such function, we will be able to predict a new target value $y$ given a pattern $x$.

We begin with the classification setting. Take for example the popular problem MNIST of handwritten digits, where the feature space $\mathcal{X}$ is the space of $28 \times 28$ matrices

with values in $[0,1]$, corresponding to the pixels of $28 \times 28$ grayscale images, i.e. $[0,1]^{28\times 28}$. We believe that there exists a function $f : \mathcal{X} \to \mathbb{R}$ that captures the properties of the handwritten digit 0 such that $f(x) = 1$ only when $x$ is an image of digit 0 and $f(x) = 0$ otherwise. Here, for simplicity, we forget about scores or probabilities and consider only a binary function $f$. Then, we would like to learn such function, so when a new image $\tilde{x}$ comes in, we can automatically decide if it represents the digit 0 by looking at the value $f(x)$. We suppose that there exists a distribution for images of digits and its labels, such that $P(x, y = 0) = P(x|y = 0)P(y = 0)$ captures the properties of the images of digit 0. Then, we want our function $f$ to minimize the expected accuracy

$$\int_{\mathcal{X}\times\mathcal{Y}} \mathbf{1}_{y\neq f(x)} dP(x, y),$$

where we are using the *accuracy* loss function $\mathbf{1}_{y\neq f(x)}$ defined in (1.1). If we want to consider other losses, such as the *hinge* loss as defined in (1.2), we can consider that $\mathcal{Y} = \mathbb{R}$, where the labels are encoded as $\{-1, 1\}$; the goal is to minimize

$$\int_{\mathcal{X}\times\mathcal{Y}} [1 - yf(x)]_+ \, dP(x, y).$$

In the regression setting we find a similar situation. Consider the problems where we want to predict the economic value of houses using descriptive features of each property: location, number of rooms, floor number, etc. These characteristics are our feature space $\mathcal{X}$, and the output space is $\mathcal{Y} = \mathbb{R}^+$, i.e. the positive real numbers, since we want to predict prices. Again, we suppose that the prices are dependent on the characteristics of each house, such that there exists a joint distribution $P(x, y)$. Now, we want to find the function $f : \mathcal{X} \to \mathcal{Y}$ that minimizes the expected *squared* loss

$$\int_{\mathcal{X}\times\mathcal{Y}} (y - f(x))^2 dP(x, y).$$

Now, the function $f$ minimizing such quantity is a good estimate to predict the value of a new property given its characteristics.

In both classification and regression cases, we arrive at an integral of a loss function. This is called the expected risk, because we take the expectation of the loss function with respect to the real distribution of data. In general, in ML we consider a set of hypothesis $\mathcal{H} = \{h(\cdot, \alpha),\ \alpha \in A\}$ where $\alpha$ are the parameters that define each hypothesis and $A$ is the space of such parameters. Then, the desired goal is to select the candidate hypothesis that minimizes the expected risk, which is defined as follows.

**Definition 1.11** (Expected Risk Minimization)**.** Given the space $\mathcal{X} \times \mathcal{Y}$ with a distribution $P(x, y)$ and a loss function $\ell$, consider a set of hypothesis $\mathcal{H} = \{h(\cdot, \alpha),\ \alpha \in A\}$ parametrized by $\alpha \in A$. Then the expected risk minimization problem is

$$\arg\min_{\alpha\in A} \left\{ R_P(\alpha) = \int_{\mathcal{X}\times\mathcal{Y}} \ell(y, h\,(x, \alpha)) dP(x, y) \right\}. \tag{1.7}$$

The distribution $P$ that appears in the expected risk is in general unknown, so it is not possible to find a solution to the minimization problem (1.7). Instead, what we have is a sample

$$D = \{(x_i, y_i),\ i = 1, \ldots, n\},$$

where each pair $(x_i, y_i)$ has been independently sampled from $P(x, y)$. Then, we use this sample $D$ to define the empirical risk minimization problem.

**Definition 1.12** (Empirical Risk Minimization)**.** Given the space $\mathcal{X} \times \mathcal{Y}$ with a distribution $P(x, y)$ and a loss function $\ell$, consider a set of hypothesis $\mathcal{H} = \{h(\cdot, \alpha),\ \alpha \in A\}$ parametrized by $\alpha \in A$, and a set

$$D = \{(x_i, y_i),\ i = 1, \ldots, n\}$$

of pairs $(x_i, y_i)$ independently sampled from $P(x, y)$; then, the empirical risk minimization problem is

$$\arg\min_{\alpha \in A} \left\{ \hat{R}_D(\alpha) = \frac{1}{n} \sum_{i=1}^{n} \ell(y_i, h\left(x_i, \alpha\right)) \right\}. \tag{1.8}$$

The paradigm of minimizing the expected risk to find a good estimate, as an alternative to minimizing the real expected risk, is called Empirical Risk Minimization (ERM). It is easy to observe that the empirical risk $\hat{R}_D$ is an unbiased estimator of $R_P$. The samples $(x_i, y_i),\ i = 1, \ldots, n$, are independent and identically distributed (iid) random variables, so, using E for the expected value, we have

$$\mathrm{E}_D\left[\hat{R}_D(\alpha)\right] = \mathrm{E}_{(x_1,y_1),\ldots,(x_n,y_n)}\left[\frac{1}{n}\sum_{i=1}^{n} \ell(y_i, h\left(x_i, \alpha\right))\right] = \frac{1}{n}\sum_{i=1}^{n} \mathrm{E}_{(x_i,y_i)}\left[\ell(y_i, h\left(x_i, \alpha\right))\right],$$

where we have used that the samples are independent. Then, since all the pairs $(x_i, y_i)$ are sampled from the same distribution $P$, the conclusion is

$$\frac{1}{n}\sum_{i=1}^{n} \mathrm{E}_{(x_i,y_i)}\left[\ell(y_i, h\left(x_i, \alpha\right))\right] = \mathrm{E}_P\left[\ell(y_i, h\left(x_i, \alpha\right))\right] = R_P(\alpha).$$

This is a desirable property of the expected risk; however, as we will see, it is not sufficient to assure that the learning process is selecting a good estimate.

### 1.2.3 Regularization and Representer Theorem

The empirical risk presented in (1.8) can be interpreted as a map from the space of hypothesis $\mathcal{H} = L_2(\mathcal{X})$ to the non-negative real numbers, namely

$$\hat{R}_D : \mathcal{H} \to \mathbb{R}_{\geq 0}.$$

The regularization theory considers adding a "stabilization" or regularization term $\Omega$ and define the regularized risk as

$$\hat{R}_D + \lambda\Omega : \mathcal{H} \to \mathbb{R}_{\geq 0},$$

where $\lambda$ is a hyperparameter. This is motivated by the concept of a well-posed problem defined by Tikhonov (Schölkopf and Smola, 2002). Although this definition falls out of the scope of this work, a well-posed problem has some desirable properties such as:

- a solution exists,
- the solution is unique,
- the solution's behaviour changes continuous with the initial conditions.

More concretely, it can be observed (Schölkopf and Smola, 2002) that finding the minimum of $\hat{R}_D$ is not well-posed, but if we consider the map $\hat{R}_D + \lambda\Omega$, it might be well-posed. From other perspective, the regularization term is also typically used to avoid the overfitting of the models to the data. This has to do with the capacity of the models considered and how adding a regularization term we penalize those models that are more complex. We will see more carefully later how the regularization and capacity concepts relate to each other.

In our case, where our functional of interest is the empirical risk $\hat{R}_D$, and we typically select a regularization term, or regularizer, that is convex, if $\hat{R}_D$ is also convex, then, the corresponding regularized functional is so too. Therefore, a usual choice is $\Omega(f) = \|f\|^2_{\mathcal{H}}$, but we can use $\Omega(f) = \Theta(\|f\|^2_{\mathcal{H}})$, where $\Theta(\cdot)$ is a monotonic increasing function. We consider then the problem of minimizing the regularized functional defined as follows.

**Definition 1.13** (Regularized Risk Functional)**.** Given the space $\mathcal{X} \times \mathcal{Y}$ with a distribution $P(x, y)$ and a loss function $\ell$, consider a set of hypothesis $\mathcal{H} = \{h(\cdot, \alpha),\ \alpha \in A\}$ parametrized by $\alpha \in A$, and a set

$$D = \{(x_i, y_i),\ i = 1, \ldots, n\}$$

of pairs $(x_i, y_i)$ independently sampled from $P(x, y)$. Then, the regularized risk functional is defined as

$$\begin{aligned}\hat{R}_{D,\lambda}(\alpha) &= \hat{R}_D(\alpha) + \lambda\Theta\left(\|h\left(\cdot, \alpha\right)\|^2_{\mathcal{H}}\right) \\ &= \frac{1}{n}\sum_{i=1}^{n} \ell(y_i, h\left(x_i, \alpha\right)) + \lambda\Theta\left(\|h\left(\cdot, \alpha\right)\|^2_{\mathcal{H}}\right).\end{aligned}$$

Here $\lambda$ is the regularization hyperparameter that regulates the tradeoff between the errors and the complexity of the model.

Minimizing the regularized functional $\hat{R}_{D,\lambda}(\alpha)$ in a general setting, for example, when $\mathcal{H} = L_2(\mathcal{X})$, is a challenging problem, because computing the norm of the hypotheses in such space is not trivial. However, when our hypothesis space $\mathcal{H}$ is an RKHS, there is an explicit solution for the minimizer, as stated in the following theorem, the Representer Theorem (Schölkopf et al., 2001).

**Theorem 1.14** (Representer Theorem)**.** *Let $\Theta : \mathbb{R} \to \mathbb{R}$ be a strictly monotonic increasing function, let $\mathcal{X}$ be a non-empty set, and let $\ell$ by an arbitrary loss. Assume that $\mathcal{H}$ is the RKHS associated to the kernel $k(\cdot, \cdot)$. Then every minimizer $f \in \mathcal{H}$ of the regularized risk*

$$\hat{R}_{D,\lambda}(\alpha) = \frac{1}{n}\sum_{i=1}^{n} \ell(y_i, f(x_i)) + \lambda\Theta\left(\|f\|^2_{\mathcal{H}}\right),$$

*with $\lambda > 0$, admits a representation of the form*

$$f(\cdot) = \sum_{i=1}^{n} \alpha_i k(x_i, \cdot).$$

*Proof.* Consider a decomposition of any function $f \in \mathcal{H}$ in two parts: one, $f_{\text{span}}$, in the span of the functions $\{k(x_1, \cdot), \ldots, k(x_n, \cdot)\}$, and other part, $f_\perp$, in the corresponding orthogonal space. That is,

$$f_{\text{span}}(\cdot) = \sum_{i=1}^{n} \alpha_i k(x_i, \cdot)$$

and $\langle k(x_j, \cdot), f_\perp \rangle = 0$ for $j = 1, \ldots, n$. Since $k(\cdot, \cdot)$ is a reproducing kernel, we have that for all $x \in \mathcal{X}$, $f(x) = \langle k(x, \cdot), f(\cdot) \rangle$, and in particular, for $x_j$, $j = 1, \ldots, n$,

$$f(x_j) = \langle k(x_j, \cdot), f(\cdot) \rangle = \langle k(x_j, \cdot), f_{\text{span}}(\cdot) \rangle + \langle k(x_j, \cdot), f_\perp(\cdot) \rangle = \langle k(x_j, \cdot), f_{\text{span}}(\cdot) \rangle .$$

Thus, we have that $f(x_j) = f_{\text{span}}(x_j)$ for $j = 1, \ldots, n$. Moreover, we observe that for all $f_\perp \in \mathcal{H}$,

$$\Theta(\|f\|^2_{\mathcal{H}}) = \Theta(\|f_{\text{span}}\|^2_{\mathcal{H}} + \|f_\perp\|^2_{\mathcal{H}}) \geq \Theta(\|f_{\text{span}}\|^2_{\mathcal{H}}).$$

Since the orthogonal part $f_\perp$ does not play a role in the loss term, then the minimizer of the empirical risk must have $f_\perp = 0$, so

$$f(\cdot) = \sum_{i=1}^{n} \alpha_i k(x_i, \cdot).$$

□

This is a relevant result because it gives an explicit form for the minimizer of the expected risk when working with RKHSs, which we will often do in this work. It is also related to the concept of duality, that, when considering linear models in some RKHS, offers the possibility of formulating the solution of optimization problems in terms of the feature space (primal formulation) or the space of inputs (dual formulation). We will present later the duality concept in detail.

## 1.3 Learning Theory

The question of what learning is and when it takes place has frequently been proposed in computer science or even philosophy, but it is difficult to formalize it in order to provide meaningful answers. Vapnik and Chervonenkis, two of the founders of statistical learning theory, expressed this question in terms of how well we minimize the expected risk when we focus on minimizing the empirical risk. They developed a theory to give specific conditions, necessary and sufficient, for when does minimizing the empirical risk leads to a small expected risk, and also how can we expect this to happen.

One crucial result that they provided is that these conditions depend on the entire hypothesis space, so different concepts characterizing it are developed. The most important one is the Vapnik-Chervonenkis (VC) dimension, which encapsulates the capacity of a hypothesis space with a single value. The smaller capacity one space has, the better the empirical risk results generalize to the expected one. Moreover, using this notion of capacity, Vapnik *et al.* derive methods that consider a particular class of functions with limited capacity, so they offer better generalization capabilities.

### 1.3.1 Uniform Convergence and Consistency

As explained in the previous section, in the supervised learning framework we have an input space $\mathcal{X}$ and an output space $\mathcal{Y}$, where there exists a distribution $P$ in $\mathcal{X} \times \mathcal{Y}$ that is unknown. Recall that given a function $h : \mathcal{X} \to \mathcal{Y}$ the corresponding expected risk is defined as

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

where $\ell$ is some loss function, which is different depending on the context. We would like then to select the function $h^*$ that minimizes $R_P(h)$, but it is not possible since we do not know $P$. However, we have an empirical sample

$$D_n = \{(x_i, y_i) \sim P(x, y),\ i = 1, \dots, n\},$$

so we actually try to minimize the empirical risk, which recall is defined as

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

Consider the case now where we admit any function as a model. In particular, we can select one defined as:

$$f(x) = \begin{cases} y_i & \text{if } x = x_i \text{ for some } i = 1, \dots, n, \\ 0 & \text{otherwise.} \end{cases} \tag{1.9}$$

Although this function $f$ minimizes the expected error, it does not account for any type of learning. If we sample new points $(\tilde{x}_i, \tilde{y}_i)$ from $P(x, y)$, if $\mathcal{X}$ is a continuous domain, the new feature vectors $\tilde{x}_i$ will almost never be equal to $x_i$, and $f$ will predict 0 almost always. Therefore, the expected risk will not be minimized. To avoid this kind of situations, we should somehow restrict the functions that we consider. For example, if we restrict ourselves to continuous functions, we cannot select the one defined in (1.9).

At the same time, we have shown that the empirical risk is an unbiased estimator of the expected one, but we can state a stronger assertion using the Chernoff bound (Chernoff, 1952). For simplicity, we will consider the binary classification case where we label the classes with $\{-1, 1\}$, we use a function $h$ with values in this label set, and the loss is $\xi_i = \frac{1}{2} |h(x_i) - y_i|$, whose values are 0 or 1. We can see that $\xi_i$ are Bernoulli trials sampled from a random variable

$$\xi = \frac{1}{2} |h(x) - y|;$$

then, we can apply the Chernoff inequality to characterize how the empirical mean converges to the expected value:

$$P\left( \left| \frac{1}{n} \sum_{i=1}^{n} \xi_i - \mathrm{E}[\xi] \right| \geq \epsilon \right) \leq 2 \exp(-2n\epsilon^2).$$

Therefore, the empirical risk converges in probability to the expected one, that is

$$\left| \hat{R}_{D_n}(h) - R_P(h) \right| \xrightarrow[n \to \infty]{P} 0 \iff \lim_{n \to \infty} P\left( \left| \hat{R}_{D_n}(h) - R_P(h) \right| \geq \epsilon \right) = 0, \tag{1.10}$$

and a similar result can be obtained for the regression case using the Hoeffding bound (Hoeffding, 1963). This means that, given a fixed function $h$, the probability that there exists a large deviation between the empirical and expected risks is small. Moreover, this convergence is exponentially fast in the number of training samples.

These two point of view seem contradictory. While the empirical risk converges to the expected risk with high probability, we can still find functions, such as the one defined in (1.9), for which minimizing the empirical risk does not imply minimizing the expected one. To reconcile these two results, we have to observe that (1.10) is a probabilistic result,

and it states that the probability of finding a function $h$ for which the empirical risk does not converge to the expected one is small; however, there might be some cases in which this occurs, as it is the case with (1.9). This happens because we are not selecting one random function $h$ from $\mathcal{H}$, but we are selecting the one that minimizes the empirical risk. Therefore, the convergence result of (1.10) is not enough to ensure that minimizing the empirical risk is a useful strategy. Instead, we want that $h_n^*$, the function minimizing the empirical risk, satisfies

$$\hat{R}_{D_n}(h_n^*) \xrightarrow[n\to\infty]{P} \inf_{h\in\mathcal{H}} R_P(h).$$

This is the consistency condition, and if this occurs, we say that the ERM is consistent. However, we want to remove from the definition the trivial cases, such as those in which there exists a function $g(\cdot)$, that does better than all others for any instance, that is $\ell(g(x_i), y_i) \leq \ell(f(x_i), y_i)$ for any $(x, y) \in \mathcal{X} \to \mathcal{Y}$. This is a trivial problem where we would just select always $g$ as the function minimizing both the empirical and expected risks, so the consistency holds trivially. To avoid this, we give the definition of nontrivial consistency.

**Definition 1.15** (Nontrivial Consistency)**.** Given a set of functions $\mathcal{H}$, and $c \in \mathbb{R}$, the ERM method is nontrivially consistent if for any non-empty subset

$$\mathcal{H}_c = \left\{ h : \hat{R}_{D_n}(h) > c,\ h \in \mathcal{H} \right\}$$

the convergence

$$\inf_{h\in\mathcal{H}_c} \hat{R}_{D_n}(h) \xrightarrow[n\to\infty]{P} \inf_{h\in\mathcal{H}_c} R_P(h)$$

is valid.

Observe that now, for large enough $c$ we rule out the function $g$, so we do not have a trivial case anymore. Vapnik then describes the sufficient and necessary conditions for ERM to be nontrivially consistent (Vapnik, 2000).

**Theorem 1.16** (Key Theorem of Statistical Learning)**.** *One-sided uniform convergence in probability,*

$$\sup_{h\in\mathcal{H}} \left( R_P(h) - \hat{R}_{D_n}(h) \right) \xrightarrow[n\to\infty]{P} 0,$$

*or, equivalently, for all* $\epsilon \geq 0$

$$\lim_{n\to\infty} P\left( \sup_{h\in\mathcal{H}} \left( R_P(h) - \hat{R}_{D_n}(h) \right) > \epsilon \right) = 0,$$

*is a necessary and sufficient condition for nontrivial consistency of ERM.*

Observe that here we have replaced the initial condition for consistency, which involved finding the minimizers of empirical and expected risks, for bounding the difference $R_P(h) - \hat{R}_{D_n}(h)$. However, this is still not a very useful theorem, because we do not know the distribution $P$ and hence $R_P(h)$, so it is difficult to find such bound. Nevertheless, Vapnik is also able to remove the explicit dependency on $P$ by using the symmetrization lemma (Vapnik, 1982).

**Lemma 1.17** (Symmetrization)**.** *For $n\epsilon^2 \geq 2$, we have*

$$P_n\left(\sup_{h\in\mathcal{H}}\left(R_P(h) - \hat{R}_{D_n}(h)\right) > \epsilon\right) \leq 2P_{2n}\left(\sup_{h\in\mathcal{H}}\left(\hat{R}_{D_{1:n}}(h) - \hat{R}_{D_{n:2n}}(h)\right) > \epsilon/2\right),$$

*where $P_n$ is the distribution of samples $D$ of size $n$ and $P_{2n}$ is the distribution of samples of size $2n$, where we use $D_{1:n}$ for the first half of size $n$ and $D_{n:2n}$ for the second one.*

Although we do not provide the proof, it is sensible to think that if two empirical risks for different samples are close, they are both close to the expected risk. Consider again the binary classification case, with the labels $-1$ and 1. Then, given a sample of size $2n$,

$$D_{2n} = \{(x_1, y_1), \ldots, (x_{2n}, y_{2_n})\},$$

there are at most $2^{2n}$ different values for the vector

$$(h(x_1), \ldots, h(x_{2n})),\ h \in \mathcal{H}.$$

The power of this lemma lies on the fact that, now, for bounding

$$P\left(\sup_{h\in\mathcal{H}}\left(R_P(h) - \hat{R}_{D_n}(h)\right) > \epsilon\right)$$

we need to take into account only a finite class of functions, that is, the $2^{2n}$ possible vectors of outputs, instead of the infinite space of functions $\mathcal{H}$.

However, it is possible that not all the possible $2^{2n}$ values can be achieved using functions from $\mathcal{H}$. Given a particular sample $D_{2n}$, denote by $\mathcal{N}(\mathcal{H}, D_{2n})$ the cardinality of the functions $h$ from $\mathcal{H}$ that have distinct values $(h(x_1), \ldots, h(x_{2n}))$. Then, we can consider the maximum of $\mathcal{N}(\mathcal{H}, D_{2n})$ among all possible samples of size $2n$, which we define as $\mathcal{N}(\mathcal{H}, 2n)$. The function,

$$\mathcal{N}(\mathcal{H}, n) = \max_{D_n} \mathcal{N}(\mathcal{H}, D_n),$$

where $n$ is the variable, is named the **shattering coefficient**, and it represents the number of different outputs that the functions in $\mathcal{H}$ can achieve on samples of size $n$. We can interpret it as the maximum number of ways the functions from $\mathcal{H}$ can separate a sample of size $n$ into two classes. When $\mathcal{N}(\mathcal{H}, n) = 2^n$, and hence all possible separations are possible, and we say the $\mathcal{H}$ shatters $n$ points. Recall that this means that there exists at least one sample of size $n$ for which the functions from $\mathcal{H}$ can achieve all possible values, but it does not imply that this is the case for all samples of size $n$.

### 1.3.2 VC dimension

Considering this finite class of functions

$$(h(x_1), \ldots, h(x_{2n})),\ h \in \mathcal{H},$$

and alongside Lemma 1.17, it can be proved that for any $\epsilon > 0$,

$$\begin{aligned} P\left(\sup_{h\in\mathcal{H}}\left(R_P(h) - \hat{R}_{D_n}(h)\right) > \epsilon\right) &\leq 4\,\mathrm{E}\left[\mathcal{N}(\mathcal{H}, D_{2n})\right]\exp\left(-\frac{n\epsilon^2}{8}\right) \\ &= 4\exp\left(\log\mathrm{E}\left[\mathcal{N}(\mathcal{H}, D_{2n})\right] - \frac{n\epsilon^2}{8}\right), \end{aligned} \tag{1.11}$$

where the expectation is taken with respect the distribution of samples of size $2n$. Therefore, if $\mathrm{E}\left[\mathcal{N}(\mathcal{H}, D_{2n})\right]$ does not grow exponentially with $n$, then the right side goes to zero as $n$ grows, and we can use the ERM to approximate the expected risk minimizer. The inequality (1.11) bounds the probability that the deviation between the minimal values of the empirical and expected risks is larger than a given $\epsilon$. However, sometimes it is more practical to fix $\delta \in \mathbb{R}$ and find what for what value of $\epsilon$ we have that

$$P\left(\sup_{h\in\mathcal{H}}\left(R_P(h) - \hat{R}_{D_n}(h)\right) > \epsilon\right) \leq \delta.$$

It can be proved (Vapnik, 1982) that with a probability of at least $1-\delta$,

$$R_P(h) \leq \hat{R}_{D_n}(h) + \sqrt{\frac{8}{n}\left(\log\mathrm{E}\left[\mathcal{N}(\mathcal{H}, D_{2n})\right] + \log\frac{4}{\delta}\right)} \tag{1.12}$$

for any $h \in \mathcal{H}$. Observe that this bound is valid for any function $h$, not just for $h_n^*$, which is a strength for those methods that do not minimize the empirical risk. Moreover, instead of minimizing the empirical risk directly, in Vapnik (2000) the author proposes to minimize the right-hand side. That is, we need to minimize

$$\mathcal{H}_P^{\text{ann}}(n) = \log\mathrm{E}\left[\mathcal{N}(\mathcal{H}, D_{2n})\right],$$

which is referred as the **annealed entropy**, and it is a characteristic of the capacity of the entire function space $\mathcal{H}$. Recall that $\mathcal{N}(\mathcal{H}, D_{2n})$ indicates in how many ways we can separate the sample $D_{2n}$ using functions $h$ from $\mathcal{H}$. This **annealed entropy** depends, however, on the distribution of samples of size $2n$, which is based on the unknown $P$ distribution. To get rid of the distribution dependence, we can use the following notion of capacity,

$$\mathcal{G}(n) = \max_{D_n}\log\mathrm{E}\left[\mathcal{N}(\mathcal{H}, D_n)\right],$$

that is called the **growth function**. Observe that by definition, and since the logarithm is an increasing monotone function, this **growth function** is the logarithm of the shattering coefficient, $\mathcal{G}(n) = \log\mathcal{N}(\mathcal{H}, n)$. It can be shown (Vapnik, 2000) that the convergence

$$\lim_{n\to\infty}\frac{\mathcal{G}(n)}{n} = 0 \tag{1.13}$$

is a necessary and sufficient condition for exponentially fast convergence of the minimal empirical risk to the minimal expected risk for all underlying distributions $P$. If for any $n$, $\mathcal{H}$ shatters $n$ points, we have that

$$\mathcal{G}(n) = n\log(2) \tag{1.14}$$

and the convergence (1.13) will not take place. However, Vapnik states that either (1.14) happens for every $n$ or there exists a maximal $m$ for which it is satisfied, and for any $n > m$

we have $\mathcal{G}(n) < n \log(2)$. This maximal $m$ value is called the Vapnik-Chervonenkis (VC) dimension, and we will denote it by $\text{VCdim}(\mathcal{H})$. Therefore, $\text{VCdim}(\mathcal{H})$ is the maximal number of points that can be shattered by $\mathcal{H}$, and we use it as a measure of the capacity of a hypothesis space. It can proved (Vapnik, 2000) that

$$\mathcal{G}(n) \leq \text{VCdim}(\mathcal{H}) \left( \log \left( \frac{n}{\text{VCdim}(\mathcal{H})} \right) + 1 \right).$$

Combining all these capacity concepts, we have the following chain of inequalities that helps us to bound the right-hand side of (1.12),

$$\mathcal{H}_P^{\text{ann}}(n) \leq \mathcal{G}(n) \leq \text{VCdim}(\mathcal{H}) \left( \log \left( \frac{n}{\text{VCdim}(\mathcal{H})} \right) + 1 \right). \tag{1.15}$$

Here, each bound is less precise than the previous one. The annealed entropy is the most precise, but it is distribution dependent. The **growth function** $\mathcal{G}(n)$ is independent of $P$, but has a definition that is difficult to use, and the bound on the right side, although being less precise, it is easy to apply because it sums up the information of the convergence in one number, the VC dimension. Therefore, we can combine (1.12) with the inequalities in (1.15) to bound the expected risk using the VC dimension as

$$R_P(h) \leq \hat{R}_D(h) + \sqrt{\frac{8}{n} \left( \text{VCdim}(\mathcal{H}) \left( \log \left( \frac{n}{\text{VCdim}(\mathcal{H})} \right) + 1 \right) + \log \frac{4}{\delta} \right)}. \tag{1.16}$$

We have defined the VC-dimension of a family of binary functions as the maximum number of points that can be shattered by such family. However, this is a definition that is valid for classification problems only. For real functions, which are applied in regression, the VC-dimension is defined as the VC-dimension of the set of indicator functions $I(x, \alpha, \beta) = \mathbf{1}_{\{h(x,\alpha)-\beta\}}$ (Vapnik, 1982).

### 1.3.3 Structural Learning

Looking at the bound (1.16), just minimizing the empirical risk might not be enough to ensure that the learning process works. Instead, Vapnik proposes to minimize the entire right-hand side, minimizing both the empirical risk and the term involving the VC dimension. To do this, he proposes the paradigm of Structural Risk Minimization (SRM), which consists on building a series of hypothesis spaces $\mathcal{H}_1, \ldots, \mathcal{H}_m$ of increasing capacity, such that

$$\text{VCdim}(\mathcal{H}_1) \leq \text{VCdim}(\mathcal{H}_2) \leq \ldots \leq \text{VCdim}(\mathcal{H}_m).$$

Then, for $j = 1, \ldots, m$, we can find the function $h_j^* \in \mathcal{H}_j$ that minimizes the empirical risk and compute $\hat{R}_D(h_j^*)$; then, we select the function $h_j^*$ for which the value

$$\hat{R}_D(h_j^*) + \sqrt{\frac{8}{n} \left( \text{VCdim}(\mathcal{H}_j) \left( \log \left( \frac{n}{\text{VCdim}(\mathcal{H}_j)} \right) + 1 \right) + \log \frac{4}{\delta} \right)}$$

is smallest.

Implementing the SRM paradigm is not easy, because it is necessary to construct that series of hypothesis classes of increasing capacity. Moreover, computing the exact VC dimension of a hypothesis space is not always possible, except for some particular

cases. For example, when using linear models, it can be proved that a hyperplane in a $d$-dimensional space can shatter at most $d + 1$ points, so this is its VC dimension. However, the VC dimension of non-linear hypothesis spaces, or that of linear models in an RKHS with infinite dimensions are not computable in general. However, with some additional considerations, Vapnik gives the following result that is useful to bound the VC dimension of any linear model, even in infinite-dimensional spaces (Vapnik, 1982).

**Theorem 1.18** (VC Dimension of Margin Hyperplanes)**.** *Consider the linear hyperplanes* $\langle w, x\rangle = 0$ *such that for a set of points* $X = \{x_1, \dots, x_r\}$*,*

$$\langle w, x_i\rangle = 1,\ i = 1, \dots, r.$$

*The VC dimension of the set of decision functions defined as* $h_w(x) = \operatorname{sign}\langle w, x\rangle$ *over the set* $X$*, with* $\|w\| \leq \Lambda$*, satisfies*

$$\operatorname{VCdim}(\mathcal{H}) \leq R^2\Lambda^2,$$

*where* $R$ *is the radius of the smallest sphere centered at the origin containing the points in* $X$*.*

Although this theorem is restricted to functions defined on $X$, with some modifications it is possible to extend it to functions defined over all the input domain $\mathcal{X}$, but we will use this version just for illustration purposes. Note that the condition $\langle w, x_i\rangle = 1$ implies that there exists a margin between points, which are correctly classified, and the decision boundary. For such kind of hyperplanes, according to this result, we can control the VC dimension of the hypothesis space by bounding the norm of $w$, independently on the dimension of the input space. This is the idea motivating the SVM, that we will present later. These are models where we use hyperplanes that keep a margin between the points and the decision boundary, and where the norm $\|w\|^2$ is penalized, which is a regularization term as those presented below.

## 1.4 Optimization

The training of some learning models involve finding a solution of some optimization problem, e.g. SVMs or NNs. In general, we are interested in minimizing the empirical risk, and we expect that the solution also minimizes the expected risk. The results concerning the VC dimension help us to understand to what extent and in what situations this is true. Nevertheless, the key resides in the choice of the family of hypotheses. In practice, we can limit the capacity of our set of hypotheses by different means, some of which are more direct, as considering a maximum margin hyperplane, and others more indirect, as including regularization. Anyway, the result is also a regularized risk functional that has to be minimized, which eventually translates to a minimization problem.

Optimization is the branch of mathematics that studies the solution of minimization problems, that is, finding the minimum of some objective function $f(x)$ in some domain $\mathcal{D}$, that can be possibly constrained. Note that we can focus on minimization problems, since maximizing $f(x)$ is equivalent to minimizing $-f(x)$. By studying the characteristics of the objective function and the constraints, we can give some necessary and sufficient conditions for the existence of solutions of the minimization problem. Sufficient conditions indicate that the problem satisfying them has a solution, but not necessarily a unique one. Necessary conditions, on the other hand, act in a reverse way: if the problem does not satisfy them, there is no solution for such a problem.

Finding a solution is not always enough, as sometimes we want to be sure that it is the only one. This is important for reproducibility: if we minimize the same problem several times, we would like to always obtain the same solution. Having a unique solution is a very desirable property, but it is not a general one. However, there are some kinds of problems that have this characteristic; these are the convex problems. Convexity is a property, that can be defined in terms of sets of functions, that lead to this uniqueness of solutions. Therefore, convex problems are interesting to study, and they have some specific properties such as the duality concept, which we will develop during this section.

In this section, we give some definitions regarding convexity, such as convex set or convex function, and detail their properties. We also present some optimization concepts and give results about the necessary and sufficient conditions that a problem needs in order to have a solution, and also when this solution is unique. Finally, we study the convex problems and the concept of duality.

### 1.4.1 Convexity

Convexity is at the center of many optimization problems related with ML. Minimizing the empirical risk can lead to minimization problems that may exhibit many local minima; however, convex problems, those minimizing a convex function in a convex set, have unique solutions. In order to properly define a convex problem, we need to present some concepts. We first define convex set and convex function.

**Definition 1.19** (Convex Set)**.** A set $X$ in a vector space $\mathbb{R}^d$ is called a convex set if for any $\lambda \in [0,1]$,

$$\lambda x_1 + (1-\lambda)x_2,\ \forall x_1, x_2 \in X.$$

**Definition 1.20** (Convex Function)**.** A function $f : X \to \mathbb{R}$ is called convex if for any $\lambda \in [0,1]$,

$$f\left(\lambda x_1 + (1-\lambda)x_2\right) \leq \lambda f(x_1) + (1-\lambda) f(x_2),\ \forall x_1, x_2 \in X,$$

and it is called strictly convex if for any $x_1, x_2 \in X,\ x_1 \neq x_2$, and $\lambda \in (0,1)$,

$$f\left(\lambda x_1 + (1-\lambda)x_2\right) < \lambda f(x_1) + (1-\lambda) f(x_2).$$

Observing that the sets below convex functions are convex will be helpful when we have inequality constraints in optimization problem. The two previous definitions are connected, as shown in the following lemma.

**Lemma 1.21.** *Let $f : \mathbb{R}^d \to \mathbb{R}$ be a convex function, then the sets*

$$X_c = \left\{x \in X, f(x) \leq c\right\},\ \forall c \in \mathbb{R},$$

*are convex.*

*Proof.* Consider any $c \in \mathbb{R}$ and any $\lambda \in [0,1]$; we want to check that $f(\lambda x_1 + (1-\lambda)x_2) \leq c$. Since $f$ is convex we have that

$$f\left(\lambda x_1 + (1-\lambda)x_2\right) \leq \lambda f(x_1) + (1-\lambda) f(x_2),$$

and if $x_1, x_2 \in X_c$ they satisfy $f(x_1) \leq c$ and $f(x_2) \leq c$; thus, we can write

$$f\left(\lambda x_1 + (1-\lambda)x_2\right) \leq \lambda f(x_1) + (1-\lambda) f(x_2) \leq \lambda c + (1-\lambda)c = c.$$

□

With the definitions that we have given we have the tools to prove the following theorem, which has an immediate application to optimization problems.

**Theorem 1.22** (Minima in Convex Sets)**.** *Let $f : \mathbb{R}^d \to \mathbb{R}$ be a convex function; if $f$ achieves its minimum value $m$ in a convex set $X \subset \mathbb{R}^d$, i.e. if the set*

$$X_{\min} = \{x \in X \text{ such that } \forall \tilde{x} \in X,\ f(x) = m \leq f(\tilde{x})\}$$

*is not empty, then $X_{\min}$ is also a convex set. Moreover, if $f$ is strictly convex, then $X_{\min}$ has a single element.*

*Proof.* Consider $x_1, x_2 \in X_{\min}$ such that $f(x_1) = f(x_2) = c$; we want to check that $x_\lambda = \lambda x_1 + (1 - \lambda x_2) \in X_{\min}$ for any $\lambda \in [0, 1]$. Since $f$ is convex, we have that

$$f(x_\lambda) = f(\lambda x_1 + (1 - \lambda)x_2) \leq \lambda f(x_1) + (1 - \lambda) f(x_2).$$

Now, for any $\tilde{x} \in X$ we have $\lambda f(x_1) \leq \lambda f(\tilde{x})$ and $(1 - \lambda) f(x_2) \leq (1 - \lambda) f(\tilde{x})$, so

$$f(x_\lambda) \leq \lambda f(\tilde{x}) + (1 - \lambda) f(\tilde{x}) = f(\tilde{x}),$$

hence $x_\lambda \in X_{\min}$. When $f$ is strictly convex, we want to check that $X_{\min}$ has a single element. Suppose that $X_{\min}$ has two distinct elements $x_1, x_2,\ x_1 \neq x_2$; then, for any $\lambda \in (0, 1)$ we have that

$$f(x_\lambda) = f(\lambda x_1 + (1 - \lambda)x_2) < \lambda f(x_1) + (1 - \lambda) f(x_2) = \lambda c + (1 - \lambda)c = c.$$

Thus, we have an element $x_\lambda \in X_{\min}$, where $x_\lambda \neq x_1, x_2$ such that $f(x_\lambda) < c$, but $c$ was the minimum value of $f$ in $X$, so we find a contradiction. Therefore, there is a single element in $X_{\min}$. □

This theorem can be applied to a particular class of problems, the constrained convex optimization problems, as we show in the following corollary.

**Corollary 1.23** (Constrained Convex Optimization)**.** *Consider the convex functions $f, c_1, \ldots, c_p$ with domain in the vector space $\mathbb{R}^d$; then the set of solutions of the problem*

$$\begin{aligned} \min_{x \in \mathbb{R}^d} \quad & f(x) \\ s.t. \quad & c_i(x) \leq 0,\ i = 1, \ldots, p, \end{aligned}$$

*is a convex set, and if $f$ is strictly convex, there is a unique solution.*

*Proof.* Observe that

$$X_f = \Big\{x \in \mathbb{R}^d, c_i(x) \leq 0 \text{ for } i = 1, \ldots, p\Big\} = \bigcap_{i=1}^{p} \Big\{x \in \mathbb{R}^d, c_i(x) \leq 0\Big\}$$

is the intersection of the sets $\big\{x \in \mathbb{R}^d, c_i(x) \leq 0\big\}$, which are the sets below convex function values, and thus convex. Therefore, the solutions of the optimization problem are the minima of the convex function $f$ on the convex set $X_f$, which is also a convex set. Also, if $f$ is strictly convex, there is a single minimizer. □

Although this is a nice result, it needs some strong assumptions: the objective function and all the constraint functions must be convex. Moreover, it does not give any clue on how to obtain those solutions; it states only some properties about them.

### 1.4.2 Constrained Problems

To study the properties of the solutions of a broader class of problems, as well as to learn under what circumstances and how we can obtain them, we will present some concepts related to the optimization of constrained problems.

Typically, when we want to find the minimum of a differentiable function $f : \mathcal{X} \to \mathbb{R}$ with $\mathcal{X} \subset \mathbb{R}^d$, we look at its stationary points, i.e. those points where the gradient of $f$ is zero. This idea is generalized in the following theorem Cristianini and Shawe-Taylor (2000).

**Theorem 1.24** (Fermat's Theorem)**.** *If $f(x)$ is a real-valued differentiable function with open domain in $\mathbb{R}^d$, a necessary condition for $x^* \in \mathbb{R}^d$ to minimize $f(\cdot)$ is $\nabla_x f(x^*) = 0$. Moreover, if $f(x)$ is a convex function, then $\nabla_x f(x^*) = 0$ is also sufficient.*

This is a well-known result, but it is not always applicable. Consider for example a minimization problem with equality constraints:

$$\min_{x \in \mathbb{R}^d} f(x) \text{ subject to } e_i(x) = 0 \text{ for } i = 1, \dots, q.$$

Then, it is possible that the stationary points $\tilde{x}$, where $\nabla_x f(\tilde{x}) = 0$, do not satisfy the constraints, i.e. $e_i(\tilde{x}) \neq 0$ for some $i = 1, \dots, q$. In this situation, Fermat's theorem is not useful; and, instead, to get some intuition on the concepts that are going to be presented, we can reason as follows. First, observe that $e_i(x) = 0$ are level curves of the function $e_i(\cdot)$, so each one is orthogonal to their respective gradient $\nabla_x e_i(\cdot)$. Let $\hat{x} \in \mathbb{R}^d$ be a feasible point, that is $e_i(\hat{x}) = 0$ for $i = 1, \dots, q$. At any point $x \in \mathbb{R}^d$, the vector $-\nabla_x f(x)$ points to the direction of maximum descent of $f(\cdot)$. If $x^*$ is a solution of the minimization problem such that $\nabla_x f(x^*) \neq 0$, then if $\nabla_x f(x^*) \notin \text{span}\left(\nabla_x e_1(x^*), \dots, \nabla_x e_q(x^*)\right)$, we can move in a direction that is parallel to some level curve $e_j(\cdot) = 0$ and is also a descent direction, which is not possible because we assumed that $x^*$ was a minimum. Therefore,

$$\nabla_x f(x^*) = \sum_{i=1}^{q} \beta_i \nabla_x e_i(x^*) \tag{1.17}$$

is a necessary condition for $x^*$ to be a solution, replacing the necessary condition $\nabla_x f(x^*) = 0$ from Fermat's theorem. Now, if we consider the function

$$\mathcal{L}(x, \boldsymbol{\beta}) = f(x) + \sum_{i=1}^{q} \beta_i e_i(x),$$

where $\boldsymbol{\beta} = (\beta_1, \dots, \beta_q)^\intercal$, we can see that we can sum up the necessary conditions as

$$\nabla_x \mathcal{L} = 0, \ \nabla_{\boldsymbol{\beta}} \mathcal{L} = 0,$$

where the first one is (1.17) and the second one reduces to the equality constraints. This is one of the intuitions behind the more general Lagrangian function that we will present next. However, before that, we need to define the constrained optimization problems, with both equality and inequality constraints.

**Definition 1.25** (Constrained Optimization Problem)**.** Given real-valued functions $f$, $c_1, \ldots, c_p$, $e_1, \ldots, e_2$, with domain in the vector space $\mathbb{R}^d$, we say that the problem

$$\begin{aligned} \min_{x \in \mathbb{R}^d} \quad & f(x) \\ \text{s.t.} \quad & c_i(x) \leq 0, \ i = 1, \ldots, q, \\ & e_i(x) = 0, \ i = 1, \ldots, p \end{aligned} \tag{1.18}$$

is a constrained optimization problem, where $f$ is the objective function, $e_1, \ldots, e_p$ are the equality constraints, and $c_1, \ldots, c_q$ the inequality constraints. Moreover, the set where the constraints are satisfied, namely

$$\left\{ x \in \mathbb{R}^d \text{ where } e_i(x) = 0, i = 1, \ldots, p; \ c_j(x) \leq 0, j = 1, \ldots, q \right\},$$

is called the feasible region, and its elements are the feasible points.

For a shorter notation, we may also use the vector-valued functions

$$\boldsymbol{c}(x) = (c_1(x), \ldots, c_q(x))^\intercal, \ \boldsymbol{e}(x) = (e_1(x), \ldots, e_p(x))^\intercal.$$

Now, as we have done in the previous example with only equality constraints, we define a function that, somehow, contains the information about the conditions of the optimization problem; this is the Lagrangian function.

**Definition 1.26** (Lagrangian)**.** Given a constrained optimization problem such as the one in (1.18), the corresponding Lagrangian is the function

$$\mathcal{L}(x, \boldsymbol{\alpha}, \boldsymbol{\beta}) = f(x) + \sum_{i=1}^{q} \alpha_i c_i(x) + \sum_{j=1}^{p} \beta_j e_i(x), \tag{1.19}$$

where $\boldsymbol{\alpha} = (\alpha_1, \ldots, \alpha_q)^\intercal \in \mathbb{R}^q$ and $\boldsymbol{\beta} = (\beta_1, \ldots, \beta_p)^\intercal \in \mathbb{R}^p$ such that $\alpha_i \geq 0$ for $i = 1, \ldots, q$ are the Lagrange multipliers, and using the vector formulation we can write the Lagrangian as

$$\mathcal{L}(x, \boldsymbol{\alpha}, \boldsymbol{\beta}) = f(x) + \boldsymbol{\alpha}^\intercal \boldsymbol{c}(x) + \boldsymbol{\beta}^\intercal \boldsymbol{e}(x).$$

With the definition of the Lagrangian function, we can present two theorems that characterize the sufficient and necessary conditions for $x \in \mathbb{R}^d$ to be a solution of the optimization problem. These are the Karush-Kuhn-Tucker (KKT) conditions. First, however, note that we can replace the equality constraints by inequality constraints. Concretely, given $q$ equality constraints $e_i(x) = 0$ for $i = 1, \ldots, q$ we can replace them by the $2q$ inequality constraints $c_i^-(x) \leq 0, -c_i^+(x) \leq 0$. Thus, given an optimization problem like the one presented in (1.18), with $p$ inequality constraints and $q$ equality constraints, we can consider an equivalent problem with $m = p + 2q$ inequality constraints, namely,

$$\min_{x \in \mathbb{R}^d} f(x) \text{ subject to } c_i(x) \leq 0 \text{ for } i = 1, \ldots, m.$$

Now we can give the first theorem to characterize solutions of the optimization problem by looking for saddle points of its corresponding Lagrangian.

**Theorem 1.27** (Saddle Point Theorem)**.** *Consider the constrained optimization problem* (1.18)*, with its corresponding Lagrangian* $\mathcal{L}(x, \boldsymbol{\alpha}, \boldsymbol{\beta})$*. If there exists* $x^* \in \mathbb{R}^d$*,*

$\boldsymbol{\alpha}^* \in \mathbb{R}^q_{\geq 0}$ *and* $\boldsymbol{\beta}^* \in \mathbb{R}^p$ *such that*

$$\forall x \in \mathbb{R}^d,\ \forall \boldsymbol{\alpha} \in \mathbb{R}^q_{\geq 0},\ \forall \boldsymbol{\beta} \in \mathbb{R}^p,\ \mathcal{L}(x^*, \boldsymbol{\alpha}, \boldsymbol{\beta}) \leq \mathcal{L}(x^*, \boldsymbol{\alpha}^*, \boldsymbol{\beta}^*) \leq \mathcal{L}(x, \boldsymbol{\alpha}^*, \boldsymbol{\beta}^*), \tag{1.20}$$

*then* $x^*$ *is a solution of the optimization problem.*

*Proof.* Following the idea of the proof (Schölkopf and Smola, 2002), consider the equivalent problem

$$\min_{x \in \mathbb{R}^d} f(x) \text{ subject to } c_i(x) \leq 0 \text{ for } i = 1, \ldots, m,$$

and its corresponding Lagrangian

$$\mathcal{L}(x, \boldsymbol{\alpha}) = f(x) + \sum_{i=1}^{m} \alpha_i c_i(x),\ \alpha_1, \ldots, \alpha_m \geq 0.$$

From the first inequality in (1.20), it follows that

$$\sum_{i=1}^{m} (\alpha_i - \alpha_i^*) c_i(x^*) \leq 0.$$

If we select $\alpha_i = {\alpha_i}^*$ for all $i = 1, \ldots, m,\ i \neq j$ and $\alpha_j = \alpha_j^* + 1$, then $c_j(x^*) \leq 0$ for all $j = 1, \ldots, m$, so $x^*$ is feasible. Moreover, choosing $\alpha_j = 0$, we have that $\alpha_j^* c_j(x^*) \geq 0$, which is only possible if

$$\alpha_i^* c_i(x^*) = 0,\ i = 1, \ldots, m. \tag{1.21}$$

Now, from the second inequality, we have that

$$f(x^*) + \sum_{i=1}^{m} \alpha_i^* c_i(x^*) \leq f(x) + \sum_{i=1}^{m} \alpha_i^* c_i(x),$$

and using (1.21) and $\alpha_i^* c_i(x) \leq 0$, for all $x \in \mathbb{R}^d$,

$$f(x^*) \leq f(x) + \sum_{i=1}^{m} \alpha_i^* c_i(x) \leq f(x),$$

so $x^*$ is a solution of the optimization problem. □

This theorem proves that the saddle-point condition is a sufficient one for $x$ to be a solution of the optimization problem (1.18). Nevertheless, if the objective function and constraints of the optimization problem (1.18) are convex, and the constraints satisfy some requirements, it is possible to prove also that the saddle-point condition is a necessary one. There are different equivalent such requirements, but here we will use the Slater's condition, which we define next.

**Definition 1.28** (Slater's Condition)**.** Let $\mathbb{R}^d$ a vector space and consider the convex functions $c_1, \ldots, c_q$ that define the feasible region

$$X = \left\{ x \in \mathbb{R}^d,\ c_i(x) \leq 0,\ i = 1, \ldots, q \right\}.$$

Then, if there exists an $x \in \mathbb{R}^d$ such that $c_i(x) < 0$ for $i = 1, \ldots, q$, we say that the feasible region $X$ satisfies Slater's condition.

Now we can give the KKT theorem, whose proof can be found in Schölkopf and Smola (2002), that describes the sufficient and necessary conditions.

**Theorem 1.29** (Saddle Point Theorem)**.** *Given the constrained optimization problem* (1.18) *with convex objective function $f$ and convex constraint functions $c_1, \ldots, c_q$ and $h_1, \ldots, h_p$, under the assumptions of Theorem 1.27, if the feasible region defined by the constraints satisfies Slater's condition, then the saddle point condition of* (1.20) *is also a necessary condition for the existence of a solution.*

This theorem characterizes the solutions of convex constrained optimization problems; however, it does not provide an algorithm or method to find such solutions.

### 1.4.3 Convex Problems and Duality

When the convex objective function and constraints are also differentiable, it is possible to define specific necessary conditions to characterize the solutions of the optimization problem; these are the KKT conditions. Moreover, using affine constraints (or Slaters' condition) they are also necessary ones.

Before that, observe that in the saddle point condition presented in (1.20), which is a sufficient condition and with Slaters' condition also a necessary one, we want to find a saddle point $(x^*, \alpha^*, \beta^*)$, that is, $\sup_{\alpha,\beta} \inf_x \mathcal{L}(x, \alpha, \beta)$ or $\inf_x \sup_{\alpha,\beta} \mathcal{L}(x, \alpha, \beta)$. These two approaches for looking for the saddle points inspire the concept of duality. For this explanation we only consider inequality constraints, and if we have equality constraints we can transform them in two inequality constraints, so we get a total of $m = p + 2q$ constraints, as shown above. Then, for any $\boldsymbol{\alpha} \in \mathbb{R}^m_{\geq 0}$ we have that

$$\inf_x \mathcal{L}(x, \boldsymbol{\alpha}) \leq \inf_x \sup_{\boldsymbol{\alpha} \geq 0} \mathcal{L}(x, \boldsymbol{\alpha});$$

thus, in particular, we can write

$$\sup_{\boldsymbol{\alpha} \geq 0} \inf_x \mathcal{L}(x, \boldsymbol{\alpha}) \leq \inf_x \sup_{\boldsymbol{\alpha} \geq 0} \mathcal{L}(x, \boldsymbol{\alpha}). \tag{1.22}$$

The left-hand side of this inequality and the inequality itself give the motivation of the dual problem and the weak duality theorem, concepts that we define next.

**Definition 1.30** (Dual Problem)**.** Given the constrained optimization problem (1.18), called the primal problem, and the corresponding Lagrangian (1.19), its dual problem is:

$$\begin{aligned} \max_{\boldsymbol{\alpha},\boldsymbol{\beta}} \quad & \Theta(\boldsymbol{\alpha}, \boldsymbol{\beta}) \\ \text{s.t.} \quad & \boldsymbol{\alpha} \geq 0, \end{aligned} \tag{1.23}$$

where $\Theta(\boldsymbol{\alpha}, \boldsymbol{\beta}) = \inf_{x \in \mathbb{R}^d} \mathcal{L}(x, \boldsymbol{\alpha}, \boldsymbol{\beta})$ is the dual objective function.

Now, we can prove a result that is known as the weak duality theorem, which illustrates the fundamental relation between the primal and dual problems.

**Theorem 1.31** (Weak Duality)**.** *Let $x^*$ be a feasible solution of the primal problem* (1.18)*, and let $(\boldsymbol{\alpha}^*, \boldsymbol{\beta}^*)$ be a feasible solution of the dual problem* (1.23)*; then*

$$\Theta(\boldsymbol{\alpha}^*, \boldsymbol{\beta}^*) \leq f(x^*).$$

*Proof.* By definition of the dual objective function, we have that

$$\begin{aligned}\Theta(\boldsymbol{\alpha}^*, \boldsymbol{\beta}^*) &= \inf_{x \in \mathbb{R}^d} \mathcal{L}(x, \boldsymbol{\alpha}^*, \boldsymbol{\beta}^*) \\ &\leq \mathcal{L}(x^*, \boldsymbol{\alpha}^*, \boldsymbol{\beta}^*) \\ &= f(x^*) + (\boldsymbol{\alpha}^*)^\intercal \boldsymbol{c}(x^*) + (\boldsymbol{\beta}^*)^\intercal \boldsymbol{e}(x^*) \leq f(x^*),\end{aligned}$$

where in the last inequality we have used the feasibility of $\boldsymbol{\alpha}^*$ and $x^*$. □

With this theorem, we see that we can upper bound the value of the optimal values of the dual problem by the optimal values of the primal one, that is

$$\sup \{\Theta(\boldsymbol{\alpha}, \boldsymbol{\beta}),\ \boldsymbol{\alpha} \geq 0\} \leq \inf \{f(x),\ \boldsymbol{c}(x) \leq 0,\ \boldsymbol{e}(x) = 0\},$$

which is an equivalent formulation of (1.22). The dual gap is the difference between the primal value and the dual one, defined as

$$\inf \{f(x),\ \boldsymbol{c}(x) \leq 0,\ \boldsymbol{e}(x) = 0\} - \sup \{\Theta(\boldsymbol{\alpha}, \boldsymbol{\beta}),\ \boldsymbol{\alpha} \geq 0\}.$$

Moreover, if we find feasible $\tilde{x}$ and $\tilde{\boldsymbol{\alpha}}, \tilde{\boldsymbol{\beta}}$ such that $\Theta(\tilde{\boldsymbol{\alpha}}, \tilde{\boldsymbol{\beta}}) = f(\tilde{x})$, then, the dual gap is zero, and $\tilde{\boldsymbol{\alpha}}, \tilde{\boldsymbol{\beta}}$ is a dual solution and $\tilde{x}$ is a primal one.

Looking back again at the formulation (1.22), assume that there exist a $x^*$ and $\boldsymbol{\alpha}^* \in \mathbb{R}^m_{\geq 0}$ such that

$$\mathcal{L}(x^*, \boldsymbol{\alpha}) \leq \mathcal{L}(x^*, \boldsymbol{\alpha}^*) \leq \mathcal{L}(x, \boldsymbol{\alpha}^*)$$

for any feasible $x \in \mathbb{R}^d$ and $\boldsymbol{\alpha} \in \mathbb{R}^m_{\geq 0}$; then, we have that

$$\begin{aligned}\inf_x \sup_{\boldsymbol{\alpha} \geq 0} \mathcal{L}(x, \boldsymbol{\alpha}) &\leq \sup_{\boldsymbol{\alpha} \geq 0} \mathcal{L}(x^*, \boldsymbol{\alpha}) \\ &= \mathcal{L}(x^*, \boldsymbol{\alpha}^*) \\ &= \inf_x \mathcal{L}(x, \boldsymbol{\alpha}^*) \\ &\leq \sup_{\boldsymbol{\alpha}} \inf_x \mathcal{L}(x, \boldsymbol{\alpha}),\end{aligned}$$

which, alongside (1.22), yields

$$\sup_{\boldsymbol{\alpha} \geq 0} \inf_{x \in \mathbb{R}^d} \mathcal{L}(x, \boldsymbol{\alpha}) = \inf_{x \in \mathbb{R}^d} \sup_{\boldsymbol{\alpha} \geq 0} \mathcal{L}(x, \boldsymbol{\alpha}).$$

Therefore, finding a saddle point $(x^*, \boldsymbol{\alpha}^*)$ of $\mathcal{L}(x, \boldsymbol{\alpha})$ implies that the dual gap is zero and, thus, $x^*$ and $\boldsymbol{\alpha}^*$ are primal and dual solutions. We can observe now the connection of duality with Theorem 1.27, which we present in the following lemma, and whose proof is similar to the reasoning we have given above.

**Lemma 1.32.** *A triplet $(x^*, \boldsymbol{\alpha}^*, \boldsymbol{\beta}^*)$ is a saddle-point satisfying (1.20) if and only if the dual gap is zero, that is $\Theta(\boldsymbol{\alpha}^*, \boldsymbol{\beta}^*) = f(x^*)$.*

We know now that finding a saddle point of the Lagrangian is equivalent to finding primal and dual solutions, and, moreover, that the dual gap is zero. Anyway, when the functions $f$, $c_i$ and $e_i$ are convex and differentiable, we can give conditions that are sufficient and necessary for the dual gap to be zero; these are the KKT conditions that we present next.

**Theorem 1.33** (KKT conditions)**.** *Consider the optimization problem* (1.18) *where* $f, c_i$ *are continuously differentiable functions over* $\mathbb{R}^d$ *and* $e_i$ *are affine functions. Suppose that there exists* $x^* \in \mathbb{R}^d$ *such that there exist* $\alpha \in \mathbb{R}^q$ *and* $\boldsymbol{\beta}^* \in \mathbb{R}^p$ *that satisfy*

$$\nabla_x f(x^*) + \sum_{i=1}^{q} \alpha_i^* \nabla_x c_i(x^*) + \sum_{j=1}^{p} \beta_i^* \nabla_x e_i(x^*) = 0 \text{ (stationarity)}, \tag{1.24}$$

$$\alpha_i^* c_i(x^*) = 0, \text{ (complementary slackness)}, \tag{1.25}$$

$$c_i(x^*) \leq 0, \; e_j(x) = 0, \text{ (primal feasibility)},$$

$$\alpha_i \geq 0, \text{ (dual feasibility)}, \tag{1.26}$$

*where* $i = 1, \ldots, q$, *and* $j = 1, \ldots, p$. *Then* $x^*$ *is a solution of the optimization problem* (1.18). *Also, if* $c_i$ *are affine functions as well, or if there exists* $\hat{x} \in \mathbb{R}^d$ *such that for* $i = 1, \ldots, q$,

$$c_i(\hat{x}) < 0 \text{ (Slaters' condition)},$$

*then the conditions* (1.24)-(1.26) *are necessary.*

The KKT conditions for problems with convex, differentiable functions are closely related with the strong duality theorem, that we give next.

**Theorem 1.34** (Strong Duality Theorem)**.** *Consider the optimization problem* (1.18) *where* $f, c_i$ *are continuously differentiable functions over* $\mathbb{R}^d$ *and* $e_i$ *are affine functions. Then there exists* $x^*$ *and* $(\boldsymbol{\alpha}^*, \boldsymbol{\beta}^*)$ *that are primal and dual solutions, respectively, and*

$$f(x^*) = \Theta(\boldsymbol{\alpha}^*, \boldsymbol{\beta}^*),$$

*that is, the dual gap is zero.*

These two results will be helpful to find a solution for the problems of interest in this thesis. The strategy that we will follow is to find $x^*$ as the point where $\nabla_x \mathcal{L}(x, \boldsymbol{\alpha}, \boldsymbol{\beta})$ vanishes, and define the dual problem as

$$\Theta(\boldsymbol{\alpha}, \boldsymbol{\beta}) = \inf_x \mathcal{L}(x, \boldsymbol{\alpha}, \boldsymbol{\beta}) = \mathcal{L}(x^*, \boldsymbol{\alpha}, \boldsymbol{\beta});$$

then, by finding its maximum, since $\Theta(\boldsymbol{\alpha}^*, \boldsymbol{\beta}^*) = f(x^*)$, we know that $x^*$ is a primal solution and $(\boldsymbol{\alpha}^*, \boldsymbol{\beta}^*)$ is a dual one. Moreover, since the KKT conditions are also necessary, we use (1.25) to study the properties of such solutions.

## 1.5 Support Vector Machines

Vapnik presented the Support Vector Machine (SVM) as a model that considers only a particular kind of hypothesis, that which, in a binary classification context, leaves a maximum margin between the classes. The motivation behind this is to limit the capacity of the hypothesis space. To minimize the empirical risk with such set of hypotheses, we formulate an optimization problem, which can be solved using the tools that we have given in the previous section. The SVM has many interesting and useful properties; for example, the optimization problem is a convex one, so the solution is unique. Also, one of the most relevant characteristics is the possibility of transforming the data into a possibly infinite-dimensional Hilbert space using the kernel trick. In this section we will present the SVM, also called L1-SVM, we will motivate it and give the definition for the

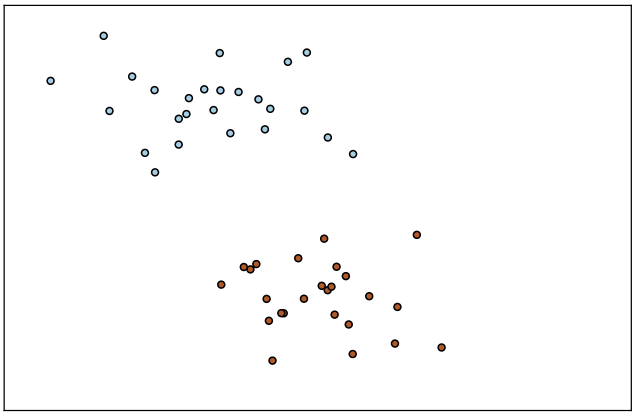


FIGURE 1.3: Example of a linearly separable binary classification problem, where classes are represented with colors blue and red.

linear formulation and its kernel extension. We will also explain SVM variants such as the L2 and LS-SVM.

### 1.5.1 Motivation

Before illustrating the idea that leads to the SVM model, we have to define some concepts. Consider a sample of a binary classification problem:

$$D = \{(x_i, y_i) \in \mathcal{X} \times \mathcal{Y},\ i = 1, \ldots, n\}$$

where we typically use the class labels $\mathcal{Y} = \{0, 1\}$ or $\mathcal{Y} = \{-1, 1\}$, and $\mathcal{X}$ is usually $\mathbb{R}^d$. As we have seen, the goal in this problem is to find a rule $f : \mathcal{X} \to \mathcal{Y}$ that minimizes the number of errors, namely $\hat{R}_D(f) = |\{i : f(x_i) \neq y_i\}|$, which using the labels $\mathcal{Y} = \{0, 1\}$ can be expressed as

$$\hat{R}_D(f) = \sum_{i=1}^{n} (y_i - f(x_i))^2.$$

If, for a classification problem with a sample $D$, there exists a linear rule $f(x) = \langle w, x \rangle + b$ that classifies correctly all the instances, that is, its corresponding $\hat{R}_D(f)$ is 0, we say that the problem is linearly separable. That is, we can find a hyperplane that divides the space in two halves, and each class is completely contained in a different half; see Figure 1.3 for an example of such problems. On the other hand, if for any linear rule $f$, its associated $\hat{R}_D(f)$ is greater than 0, we say that the problem is not linearly separable.

Given a linearly separable problem, there might exist infinitely many linear functions that separate the two classes perfectly; see, for instance, the previous example with three different boundaries that achieve perfect classification in Figure 1.4. We may wonder then which is the best boundary among all those whose error is zero. Vapnik proposes to use maximum margin hyperplanes, that is, to consider only those hyperplanes whose distance to the closest point is maximal. Vapnik also shows that this kind of functions has better generalization properties, which has to do with the capacity of the hypothesis space. The capacity of a set of functions is not easily interpretable; however, we can illustrate the properties of the maximum margin hyperplanes with the following reasoning. Given an empirical training sample, we can learn a function that perfectly separates the two classes, but the corresponding boundary is very close to the points of one of the

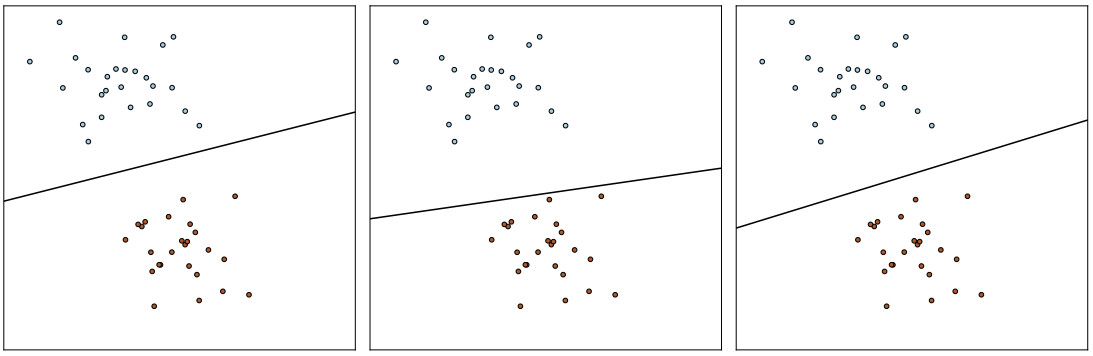


FIGURE 1.4: Linearly separable problem with three different perfect classification rules.

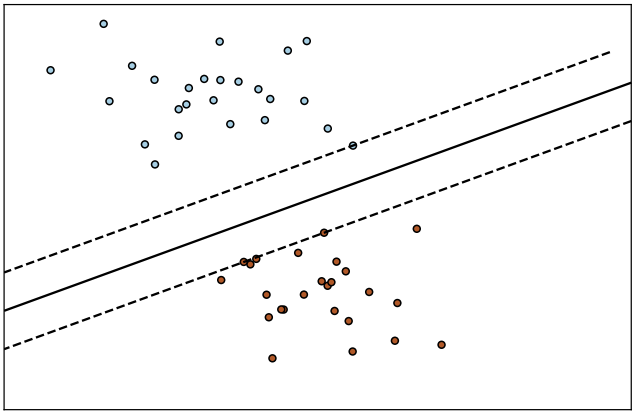


FIGURE 1.5: Linearly separable problem separated by a maximum margin boundary. The classification boundary is shown with a solid line. The lines parallel to the boundary that contain the closest points to such boundary are depicted with dashed lines.

classes, say the red one in our example. If we sampled new points, we do not know if the boundary would still achieve perfect classification, but it seems probable that some of the new red points would fall in the wrong side; see the middle example in Figure 1.4 to illustrate this explanation. However, if we select the boundary that is the furthest away from its closest points, we expect that such maximum margin boundary will be valid for new samples; see in Figure 1.5 our separable problem divided by a maximum margin hyperplane, which is a line in the two-dimensional case. We can see that, in this case, the distance from the decision boundary to its closest point is maximal, which intuitively accounts for a better generalization to new data.

To find the maximum margin hyperplane, recall first that the signed distance from a point $\tilde{x}$ to plane defined by $\langle w, \cdot \rangle + b = 0$ is given by

$$d(\tilde{x}) = \frac{1}{\|w\|}(\langle w, \tilde{x} \rangle + b).$$

Also, if we label the two classes using $\mathcal{Y} = \{-1, 1\}$, we can say that an instance $(x_i, y_i)$ is correctly classified when

$$y_i \frac{1}{\|w\|} (\langle w, x_i \rangle + b) > 0.$$

Then, if we want to find the hyperplane that maximizes the margin while classifying all instances correctly, we can express it as

$$\begin{aligned} \max_{w,b} \quad & m \\ \text{s.t.} \quad & y_i \frac{1}{\|w\|}(\langle w, x_i \rangle + b) \geq m. \end{aligned}$$

This is equivalent to the following problem

$$\begin{aligned} \max_{w,b} \quad & m \\ \text{s.t.} \quad & y_i(\langle w, x_i \rangle + b) \geq m \|w\|, \end{aligned}$$

and, since the norm of the vector $w$ is not relevant, we can choose it such that $\|w\| = 1/m$; thus, the problem is given by

$$\begin{aligned} \max_{w,b} \quad & \frac{1}{\|w\|} \\ \text{s.t.} \quad & y_i(\langle w, x_i \rangle + b) \geq 1, \end{aligned}$$

which has the same solutions as the following problem with easier derivatives:

$$\begin{aligned} \min_{w,b} \quad & \frac{1}{2} \|w\|^2 \\ \text{s.t.} \quad & y_i(\langle w, x_i \rangle + b) \geq 1. \end{aligned}$$

Observe that, in order to satisfy the constraints, it is not enough for the instances $(x_i, y_i)$ to be well classified, namely

$$y_i(\langle w, x_i \rangle + b) > 0,$$

but we are imposing a margin between the decision boundary and the data points, so the constraints are actually

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

Therefore, we have expressed the problem of finding the maximum margin hyperplane as an optimization problem, similar to those that we have studied before in this chapter. However, this is an approach that is only valid for separable problems; when using real-world data it is uncommon to find this kind of situations. To solve this issue, we have to introduce variables in the optimization problem that allow for misclassified points.

### 1.5.2 Linear Support Vector Machines

When we face a problem that is not linearly separable, it means that, for any rule $f(\cdot) = \langle w, \cdot \rangle + b$ there exist instances such that

$$y_i(\langle w, x_i \rangle + b) \leq 0.$$

Moreover, when considering the constraints of the SVM, any instance $(x_i, y_i)$ such that

$$y_i(\langle w, x_i \rangle + b) < 1$$

is not satisfying the constraints. We can overcome this by adding some slack variables $\xi_i \geq 0$, which we use to compensate in these constraints as

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

Now, for instances $(x_i, y_i)$ such that $y_i(\langle w, x_i \rangle + b) \geq 1$, its corresponding slack variable is $\xi_i = 0$. However, for instances such that $y_i(\langle w, x_i \rangle + b) < 1$, its corresponding slack variable is $\xi_i = 1 - y_i(\langle w, x_i \rangle + b)$; thus, $y_i(\langle w, x_i \rangle + b) = 1 - \xi_i$. Summing it up, we can express then the slack variables as

$$\xi_i = [1 - y_i(\langle w, x_i \rangle + b)]_+ = \max(0, 1 - y_i(\langle w, x_i \rangle + b)),$$

which is the hinge loss function defined in (1.2), so the slack variables are also named hinge variables. Now we can give the definition of the optimization problem that corresponds to finding a maximum margin hyperplane with slack variables.

**Definition 1.35** (Primal Problem for Classification SVM)**.** Given a binary classification sample

$$D = \{(x_i, y_i),\ i = 1, \ldots, m\},$$

where $x_i \in \mathbb{R}^d$ are the feature vectors and $y_i \in \{-1, 1\}$ are the class labels, the primal problem of the SVM for binary classification is:

$$\begin{aligned} \min_{w,b,\boldsymbol{\xi}} \quad & C \sum_{i=1}^{m} \xi_i + \frac{1}{2} \|w\|^2 \\ \text{s.t.} \quad & y_i(\langle w, x_i \rangle + b) \geq 1 - \xi_i, \\ & \xi_i \geq 0, \end{aligned} \tag{1.27}$$

where $\boldsymbol{\xi} = (\xi_1, \ldots, \xi_m)^\intercal$, and $w, b, \boldsymbol{\xi}$ are the primal variables.

Observe that this primal problem can be interpreted then as a regularized risk functional, where we have the loss term, we use the hinge loss, and the regularization, penalizing $\|w\|^2$; here $C$ is a hyperparameter that regulates the tradeoff between the margin and errors that the model makes.

In the regression setting we need to define the primal problem differently. We again want to define a regularized risk functional, but we need to use a loss suited for regression. In the SVM for regression, the goal is to find a function such that the points are at a distance smaller than a given $\epsilon \in \mathbb{R}$. That is, given a training sample

$$D = \{(x_i, t_i), i = 1, \ldots, m\},$$

with $t_i \in \mathbb{R}$ being the target values, the hard constraints to express this would be

$$|t_i - (\langle w, x_i \rangle + b)| \leq \epsilon.$$

However, it is possible that there does not exist such function, so we introduce slack variables $\xi_i \geq 0$ again as

$$|t_i - (\langle w, x_i \rangle + b)| \leq \epsilon + \xi_i,$$

That is, the slack variables are defined as

$$\xi_i = [t_i - (\langle w, x_i \rangle + b)]_\epsilon.$$

This is equivalent to using the $\epsilon$-insenstitive loss presented in (1.5). Getting rid of the absolute value, these constraints can be decomposed as

$$\begin{aligned}\langle w, x_i\rangle + b &\geq t_i - \epsilon - \xi_i,\\ \langle w, x_i\rangle + b &\leq t_i + \epsilon + \hat{\xi}_i.\end{aligned}$$

Observe here that

$$\xi_i + \hat{\xi}_i = \left[\langle w^*, x_i\rangle + b^* - t_i\right]_\epsilon,$$

so that

- $\xi_i = 0, \hat{\xi}_i = 0$: these are points inside the tube.
- $\xi_i = 0, \hat{\xi}_i > 0$: these are points above the tube.
- $\xi_i > 0, \hat{\xi}_i = 0$: these are points below the tube.
- $\xi_i > 0, \hat{\xi}_i > 0$: this is an impossible situation, since at any time a point satisfies one of the two constraints.

Therefore, we can also assert that $\xi_i\hat{\xi}_i = 0$. Now, we can write the regression SVM primal problem as follows.

**Definition 1.36** (Primal Problem of SVM for Regression)**.** Given a binary classification sample

$$D = \{(x_i, t_i),\ i = 1, \ldots, m\},$$

where $x_i \in \mathbb{R}^d$ are the feature vectors and $t_i \in \mathbb{R}$ are the target values, the primal problem of the SVM for regression is:

$$\begin{aligned}\min_{w,b,\boldsymbol{\xi},\hat{\boldsymbol{\xi}}} \quad & C\sum_{i=1}^{m}\left(\xi_i + \hat{\xi}_i\right) + \frac{1}{2}\|w\|^2\\ \text{s.t.} \quad & \langle w, x_i\rangle + b \geq t_i - \epsilon - \xi_i,\\ & \langle w, x_i\rangle + b \leq t_i + \epsilon + \hat{\xi}_i,\\ & \xi_i \geq 0, \hat{\xi}_i \geq 0,\end{aligned} \tag{1.28}$$

where $\boldsymbol{\xi} = (\xi_1, \ldots, \xi_m)^\intercal$ and analogously for $\hat{\boldsymbol{\xi}}$, and $w, b, \boldsymbol{\xi}, \hat{\boldsymbol{\xi}}$ are the primal variables.

We have again defined a regularized risk functional, where now the selected loss is the $\epsilon$-insensitive one. Again, $C$ is the tradeoff hyperparameter, but in this case we also have $\epsilon$ as the hyperparameter that defines the width of the tube insensitive to errors.

In both cases, classification and regression, we have defined the corresponding optimization problems, which can be also interpreted as regularized risk functionals, that present some similarities. In fact, it is possible to express both problems using a unifying formulation presented in Lin (2001).

**Definition 1.37** (SVM Primal Problem)**.** Given a sample

$$D = \{(x_i, y_i, p_i),\ i = 1, \ldots, n\},$$

where $x_i \in \mathbb{R}^d$ are the feature vectors and $p_i \in \mathbb{R}$, the primal problem is defined as

$$\begin{aligned} \min_{w,b,\boldsymbol{\xi}} \quad & C\sum_{i=1}^{n}\xi_i + \frac{1}{2}\|w\|^2 \\ \text{s.t.} \quad & y_i(\langle w, x_i\rangle + b) \geq p_i - \xi_i, \\ & \xi_i \geq 0, \end{aligned} \tag{1.29}$$

which can be interpreted as the classification primal SVM problem (1.27) when $n = m$ and

$$p_i = 1, \quad \text{for } i = 1, \ldots, m;$$

and can be interpreted as the regression primal SVM problem (1.28) when $n = 2m$ and

$$\begin{aligned} p_i &= t_i - \epsilon, \quad & y_i &= 1 \quad & \text{for } i &= 1, \ldots, m; \\ p_i &= -t_i - \epsilon, \quad & y_i &= -1 \quad & \text{for } i &= m+1, \ldots, 2m. \end{aligned}$$

For the rest of this work we will use this formulation whenever it is not necessary to make a comment regarding the specific classification or regression characteristics.

### 1.5.3 Dual Formulation

The problem defined in (1.29) is an optimization primal problem with a convex and differentiable objective function, and with affine constraint functions; then, we know from the strong duality theorem, presented in Theorem 1.34, that we can use its corresponding dual problem to find the solution. Recall that to obtain the dual problem, defined in Definition 1.30, we need the Lagrangian function. In this case, the Lagrangian corresponding to (1.29) is

$$\mathcal{L}(w, b, \boldsymbol{\xi}, \boldsymbol{\alpha}, \boldsymbol{\beta}) = C\sum_{i=1}^{n}\xi_i + \frac{1}{2}\|w\|^2 - \sum_{i=1}^{n}\alpha_i\left[y_i(\langle w, x_i\rangle + b) - p_i + \xi_i\right] - \sum_{i=1}^{n}\beta_i\xi_i,$$

where $\boldsymbol{\alpha}, \boldsymbol{\beta} \geq 0$ are the Lagrange multipliers. Although in the general optimization problem explanations we used $\boldsymbol{\beta}$ for equality constraints, here we use it to discriminate between the two kinds of constraints in (1.29). Now, the dual problem is defined as

$$\Theta(\boldsymbol{\alpha}, \boldsymbol{\beta}) = \inf_{w,b,\boldsymbol{\xi}} \mathcal{L}(w, b, \boldsymbol{\xi}, \boldsymbol{\alpha}, \boldsymbol{\beta}).$$

Observe that $\mathcal{L}(w, b, \boldsymbol{\xi}, \boldsymbol{\alpha}, \boldsymbol{\beta})$ is convex in all the primal variables, $w, b$ and $\boldsymbol{\xi}$, because the objective function is convex and the constraints are affine. Moreover, it is also differentiable in the primal variables, so to find the minimum of the Lagrangian in the primal variables is sufficient to take derivatives and select those points where these derivatives are zero; we then get:

$$\nabla_w \mathcal{L}(w, b, \boldsymbol{\xi}, \boldsymbol{\alpha}, \boldsymbol{\beta}) = 0 \implies w = \sum_{i=1}^{n} y_i\alpha_i x_i, \tag{1.30}$$

$$\nabla_b \mathcal{L}(w, b, \boldsymbol{\xi}, \boldsymbol{\alpha}, \boldsymbol{\beta}) = 0 \implies \sum_{i=1}^{n} y_i\alpha_i = 0, \tag{1.31}$$

$$\nabla_{\xi_i} \mathcal{L}(w, b, \boldsymbol{\xi}, \boldsymbol{\alpha}, \boldsymbol{\beta}) = 0 \implies C - \alpha_i - \beta_i = 0, \tag{1.32}$$

If we plug these results in the Lagrangian, we get a simpler formula. Using (1.31) the terms with $b$ disappear, and with (1.32) all the terms with $\xi_i$ vanish as well. Then, replacing $w$ according to (1.30), we get

$$\begin{aligned}\inf_{w,b,\boldsymbol{\xi}} \mathcal{L}(w, b, \boldsymbol{\xi}, \boldsymbol{\alpha}, \boldsymbol{\beta}) &= \frac{1}{2}\left\langle \sum_{i=1}^{n} y_i \alpha_i x_i, \sum_{i=1}^{n} y_i \alpha_i x_i \right\rangle \\ &\quad - \sum_{i=1}^{n} \alpha_i \left[ y_i \left\langle \sum_{j=1}^{n} y_j \alpha_j x_j, x_i \right\rangle \right] + \sum_{i=1}^{n} \alpha_i p_i \\ &= -\frac{1}{2} \sum_{i=1}^{n} \sum_{j=1}^{n} y_i y_j \alpha_i \alpha_j \left\langle x_i, x_j \right\rangle + \sum_{i=1}^{n} \alpha_i p_i.\end{aligned}$$

We observe here that $\boldsymbol{\beta}$ disappears from the dual objective function $\Theta$. However, we still have to take into account that we have used (1.31) and (1.32). From (1.32), since $\beta_i \geq 0$, we get that $\alpha_i \leq C$. Then, the dual problem can be defined as follows.

**Definition 1.38** (SVM Dual Problem)**.** The dual problem corresponding to problem (1.29) is

$$\begin{aligned} \min_{\boldsymbol{\alpha}} \quad & \frac{1}{2} \sum_{i=1}^{n} \sum_{j=1}^{n} y_i y_j \alpha_i \alpha_j \left\langle x_i, x_j \right\rangle - \sum_{i=1}^{n} \alpha_i p_i \\ \text{s.t.} \quad & \sum_{i=1}^{n} y_i \alpha_i = 0, \\ & 0 \leq \alpha_i \leq C, \end{aligned} \tag{1.33}$$

where the first constraint is named the equality constraint and the rest are the box constraints.

The result is a constrained quadratic problem where the first term is quadratic and the second is linear. Algorithms, such as the Sequential Minimal Optimization (SMO) (Keerthi et al., 2001), have been developed to solve this dual problem efficiently. Moreover, optimized implementations of SMO are available, for example, `LIBSVM` (Chang and Lin, 2011).

Observe that (1.33) is a convex problem, so it has a single value $\boldsymbol{\alpha}^*$ that is a solution. Therefore, by solving the dual problem, using algorithms such as SMO, we obtain this optimal $\boldsymbol{\alpha}^*$; then, using (1.30) we can recover the optimal primal variable $w^*$ as

$$w^* = \sum_{i=1}^{n} y_i \alpha_i^* x_i.$$

Observe that the primal problem (1.29) has a convex, differentiable objective function and affine constraints, which satisfy the conditions from Theorem 1.33; thus, the KKT conditions are necessary. Concretely, the complementary slackness conditions (1.25) in our case can be expressed as

$$\alpha_i^* \left[ y_i (\langle w^*, x_i \rangle + b^*) - p_i + \xi_i^* \right] = 0, \tag{1.34}$$

$$\beta_i^* \xi_i^* = 0, \tag{1.35}$$

TABLE 1.1: Classification of the vectors in terms of the value of $\alpha_i^*$. The first three rows illustrate the support vectors, while the last one illustrates those instances that are not support vectors.

| $\alpha_i^*$ | $\beta_i^*$ | $\xi_i^*$ | $y_i(\langle w^*, x_i\rangle + b^*)$ |
|---|---|---|---|
| Support Vectors | | | |
| $0 < \alpha_i^* < C$ | $\beta^* > 0$ | $\xi_i^* = 0$ | $y_i(\langle w^*, x_i\rangle + b^*) = p_i$ |
| $\alpha_i^* = C$ | $\beta^* = 0$ | $0 < \xi_i^* < p_i$ | $0 < y_i(\langle w^*, x_i\rangle + b^*) < p_i$ |
| $\alpha_i^* = C$ | $\beta^* = 0$ | $p_i \leq \xi_i^*$ | $y_i(\langle w^*, x_i\rangle + b^*) \leq 0$ |
| Non-Support Vectors | | | |
| $\alpha_i = 0$ | – | – | – |

for all $i = 1, \ldots, n$. Now we realize from (1.34) that when $y_i(\langle w^*, x_i\rangle + b^*) - p_i + \xi_i^* > 0$, the corresponding dual variable must be $\alpha_i = 0$. Therefore, the primal solution $w^*$ is

$$w^* = \sum_{i \in \mathcal{I}} y_i \alpha_i^* x_i,$$

where the elements of the set $\mathcal{I} = \{i : \alpha_i > 0\}$ receive the name of support vectors.

We still have to see how we can recover the other primal variables $\boldsymbol{\xi}^*$ and $b^*$. For this goal, the KKT conditions will be useful. Recall that $\alpha_i$ and $\beta_i$ are connected through (1.32), such that

$$\alpha_i^* + \beta_i^* = C.$$

Therefore, when $0 < \alpha_i < C$, and, thus, $\beta_i \neq 0$, we have from (1.35) that $\xi_i^* = 0$ and, in consequence from (1.34),

$$y_i(\langle w^*, x_i\rangle + b^*) - p_i = 0.$$

That is, we can recover $b^*$ from the set of indices $\{i : 0 < \alpha_i < C\}$. After obtaining $b^*$, we focus on the indices $i$ such that $\alpha_i^* = C$, so $\beta_i^* = 0$ and, since (1.35) is satisfied, $\xi_i^*$ is not necessarily 0. Then, we can apply (1.34) to get the solutions $\xi_i^*$ from

$$y_i(\langle w^*, x_i\rangle + b^*) - p_i - \xi_i^* = 0.$$

Actually, we can sum this up and use these necessary KKT conditions, equations (1.34) and (1.35), to establish a classification of the support vectors according to their corresponding $\alpha_i^*$, which we give in Table 1.1. To study the meaning of this table, we will consider first the binary classification setting, where $p_i = 1$ for all $i = 1, \ldots, n$. The first row corresponds to points where $y_i(\langle w^*, x_i\rangle + b^*) = 1$, that is, points that are correctly classified and are just on the margin. The points from the second row are also points that are correctly classified, but they lie inside the margin. Finally, the points in the third row are misclassified. Therefore, the model of the SVM in the classification settings depend only on those points that lie over the margin border, inside the margin, or are misclassified. The points that are correctly classified but outside the margin do not play any role in the final model.

In the case of regression, recall that we have two slack variables: $\xi_i$ for when de points are below the tube and $\hat{\xi}_i$ for points above it; however, in the unifying formulation we double the variables $\xi_i$ such that $\xi_i = \xi_i$ and $\hat{\xi}_i = \xi_{n+1}$. Then, we can observe the

following property:

$$\xi_i + \hat{\xi}_i = \xi_i + \xi_{n+i} = \left[\langle w^*, x_i\rangle + b^* - t_i\right]_\epsilon ,$$

where $\xi_i \xi_{n+i} = 0$. Consider the first row of Table 1.1; if $\xi_i = 0$, then $\langle w^*, x_i\rangle + b^* = t_i - \epsilon$, which corresponds to a point in the lower border of the $\epsilon$-tube, and if $\xi_{i+n} = 0$, we have that $\langle w^*, x_i\rangle + b^* = t_i + \epsilon$, which corresponds to a point in the upper border of the $\epsilon$-tube. In the second and third rows, if $\xi_i > 0$, then $\xi_{i+n} = 0$, and we have points such that $\langle w^*, x_i\rangle + b^* < t_i - \epsilon$, that is, they are below the tube. Analogously, if $\xi_{i+n} > 0$, we have points above the tube.

### 1.5.4 Kernel Extension

Observe that the dual problem (1.33) depends only on the inner product between the feature vectors $x_i$. Also, observe that, using the result from (1.30) we can write the prediction for a new instance $\tilde{x}$ as

$$\langle w^*, \tilde{x}\rangle + b^* = \sum_{i=1}^{n} y_i \alpha_i^* \langle x_i, \tilde{x}\rangle + b^*.$$

This result is similar to that of the Representer Theorem, presented in Theorem 1.14. In fact, consider the following model $f(\cdot) = \langle w, \phi(\cdot)\rangle + b$ where we are using the transformation $\phi : \mathbb{R}^d \to \mathcal{H}$ and $\mathcal{H}$ is some RKHS with reproducing kernel $k$, such that for all $x, \tilde{x} \in \mathbb{R}^d$,

$$\langle \phi(x), \phi(\tilde{x})\rangle = k(x, \tilde{x}).$$

Now, we are not looking for a maximum margin hyperplane in the original feature space $\mathbb{R}^d$, but in a possibly infinite-dimensional space $\mathcal{H}$. The primal problem for the kernel extension using the unified formulation is

$$\begin{aligned} \min_{w,b,\boldsymbol{\xi}} \quad & C\sum_{i=1}^{n} \xi_i + \frac{1}{2}\|w\|^2 \\ \text{s.t.} \quad & y_i(\langle w, \phi(x_i)\rangle + b) \geq p_i - \xi_i, \\ & \xi_i \geq 0. \end{aligned} \tag{1.36}$$

Its corresponding Lagrangian is then

$$\mathcal{L}(w, b, \boldsymbol{\xi}, \boldsymbol{\alpha}, \boldsymbol{\beta}) = \sum_{i=1}^{n} \xi_i + \frac{1}{2}\|w\|^2 - \sum_{i=1}^{n} \alpha_i \left[y_i(\langle w, \phi(x_i)\rangle + b) - p_i + \xi_i\right] - \sum_{i=1}^{n} \beta_i \xi_i,$$

and the optimal value for the variable $w$, which in the linear case was given in (1.30), is now

$$\nabla_w \mathcal{L}(w, b, \boldsymbol{\xi}, \boldsymbol{\alpha}, \boldsymbol{\beta}) = 0 \implies w^* = \sum_{i=1}^{n} y_i \alpha_i \phi(x_i).$$

Since $k$ is a reproducing kernel, we can also express this as

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

as the Representer Theorem indicates for a regularized risk functional like (1.36). Repeating the procedure that we have followed in the linear case, we can obtain the corresponding dual problem, which is the following.

$$\begin{aligned} \min_{\boldsymbol{\alpha}} \quad & \frac{1}{2}\sum_{i=1}^{n}\sum_{j=1}^{n} y_i y_j \alpha_i \alpha_j k(x_i, x_j) - \sum_{i=1}^{n} \alpha_i p_i \\ \text{s.t.} \quad & \sum_{i=1}^{n} y_i \alpha_i = 0, \\ & 0 \leq \alpha_i \leq C, \end{aligned}$$

where we use that $k(x_i, x_j) = \langle \phi(x_i), \phi(x_j) \rangle$. We can express the dual problem using a matrix formulation as

$$\begin{aligned} \min_{\boldsymbol{\alpha}} \quad & \frac{1}{2}\boldsymbol{\alpha}^\intercal Q \boldsymbol{\alpha} - \boldsymbol{\alpha}^\intercal \boldsymbol{p} \\ \text{s.t.} \quad & \sum_{i=1}^{n} y_i \alpha_i = 0, \\ & 0 \leq \alpha_i \leq C, \end{aligned}$$

with $\boldsymbol{p} = (p_1, \ldots, p_n)^\intercal$. Here, $Q$ is called the kernel matrix and is defined as

$$Q_{ij} = y_i y_j k(x_i, x_j)$$

in the case of the kernelized case, or as

$$Q_{ij} = y_i y_j \langle x_i, x_j \rangle$$

in the linear one.

## 1.6 SVM Variants

The SVM has experienced a great popularity for many years, and several modifications and extensions have been proposed. In this work we will consider two relevant variants: the L2 and LS-SVM, and we will also refer to the standard SVM as L1-SVM to differentiate it from these variants. For compactness, we will consider the non-linear models to present these variants, namely

$$f(\cdot) = \langle w, \phi(\cdot) \rangle + b,$$

which can be reduced to the linear case just by selecting $\phi$ as the identity function.

### 1.6.1 L2-SVM

The idea motivating the L2-SVM variant (Burges, 1998) is replacing the loss functions used in the L1-SVM for their squared counterparts. The hinge loss, used for classification and defined in (1.2), is replaced by the squared hinge loss, as defined in (1.3). The $\epsilon$-insensitive loss defined as (1.5), which is used in the regression L1-SVM, is replaced by its squared version as defined in (1.6).

We can give then the definitions of the L2-SVM problems for classification and regression.

**Definition 1.39** (Primal Problem of L2-SVM for Classification)**.** Given a binary classification sample

$$D = \{(x_i, y_i),\ i = 1, \dots, m\},$$

where $x_i \in \mathbb{R}^d$ are the feature vectors and $y_i \in \{-1, 1\}$ are the class labels, the primal problem of the L2-SVM for binary classification is:

$$\begin{aligned} \min_{w,b,\boldsymbol{\xi}} \quad & \frac{C}{2}\sum_{i=1}^{m}(\xi_i)^2 + \frac{1}{2}\|w\|^2 \\ \text{s.t.} \quad & y_i(\langle w, \phi(x_i)\rangle + b) \geq 1 - \xi_i, \end{aligned} \tag{1.37}$$

where $\boldsymbol{\xi} = (\xi_1, \dots, \xi_m)^\intercal$, and $w, b, \boldsymbol{\xi}$ are the primal variables.

**Definition 1.40** (Primal Problem of L2-SVM for Regression)**.** Given a regression sample

$$D = \{(x_i, t_i),\ i = 1, \dots, m\},$$

where $x_i \in \mathbb{R}^d$ are the feature vectors and $t_i \in \mathbb{R}$ are the target values, the primal problem of the L2-SVM for regression is:

$$\begin{aligned} \min_{w,b,\boldsymbol{\xi},\hat{\boldsymbol{\xi}}} \quad & \frac{C}{2}\sum_{i=1}^{m}\left((\xi_i)^2 + (\hat{\xi}_i)^2\right) + \frac{1}{2}\|w\|^2 \\ \text{s.t.} \quad & \langle w, \phi(x_i)\rangle + b \geq t_i - \epsilon - \xi_i, \\ & \langle w, \phi(x_i)\rangle + b \leq t_i + \epsilon + \hat{\xi}_i, \end{aligned} \tag{1.38}$$

where $\boldsymbol{\xi} = (\xi_1, \dots, \xi_m)^\intercal$ and analogously for $\hat{\boldsymbol{\xi}}$, and $w, b, \boldsymbol{\xi}, \hat{\boldsymbol{\xi}}$ are the primal variables.

Observe in both cases that, since the slack variables are squared in the objective function, the constraints to enforce that these slack variables are non-negative are no longer needed. As with the L1-SVM, we will use a formulation to unify both problems and give a development that is valid for both cases.

**Definition 1.41** (L2-SVM Primal Problem)**.** Given a sample

$$D = \{(x_i, y_i, p_i),\ i = 1, \dots, n\},$$

where $x_i \in \mathbb{R}^d$ are the feature vectors, $y_i = \pm 1$ and $p_i \in \mathbb{R}$, the primal problem is defined as

$$\begin{aligned} \min_{w,b,\boldsymbol{\xi}} \quad & \frac{C}{2}\sum_{i=1}^{n}(\xi_i)^2 + \frac{1}{2}\|w\|^2 \\ \text{s.t.} \quad & y_i(\langle w, x_i\rangle + b) \geq p_i - \xi_i, \end{aligned} \tag{1.39}$$

which can be interpreted as the classification primal L2-SVM problem (1.37) when $n = m$ and

$$p_i = 1, \quad \text{for } i = 1, \dots, m;$$

and can be interpreted as the regression primal L2-SVM problem (1.38) when $n = 2m$ and

$$\begin{aligned} p_i &= t_i - \epsilon, \quad & y_i &= 1 \quad & \text{for } i &= 1, \dots, m; \\ p_i &= -t_i - \epsilon, \quad & y_i &= -1 \quad & \text{for } i &= m+1, \dots, 2m. \end{aligned}$$

We follow an analogous procedure to the one we have used for the standard SVM. The Lagrangian corresponding to the primal problem (1.39) is

$$\mathcal{L}(w, b, \boldsymbol{\xi}, \boldsymbol{\alpha}) = \frac{C}{2}\sum_{i=1}^{n}(\xi_i)^2 + \frac{1}{2}\|w\|^2 - \sum_{i=1}^{n}\alpha_i\left[y_i(\langle w, \phi(x_i)\rangle + b) - p_i + \xi_i\right],$$

where $\boldsymbol{\alpha} \geq 0$ are the Lagrange multipliers. Again, the dual problem is defined as

$$\Theta(\boldsymbol{\alpha}) = \inf_{w,b,\boldsymbol{\xi}} \mathcal{L}(w, b, \boldsymbol{\xi}, \boldsymbol{\alpha}),$$

and $\mathcal{L}(w, b, \boldsymbol{\alpha})$ is convex in all the primal variables, $w, b$ and $\boldsymbol{\xi}$. Therefore, to find the minimum of the Lagrangian in the primal variables is sufficient to take derivatives and make them zero:

$$\nabla_w \mathcal{L}(w, b, \boldsymbol{\xi}, \boldsymbol{\alpha}) = 0 \implies w = \sum_{i=1}^{n} y_i\alpha_i\phi(x_i), \tag{1.40}$$

$$\nabla_b \mathcal{L}(w, b, \boldsymbol{\xi}, \boldsymbol{\alpha}) = 0 \implies \sum_{i=1}^{n} y_i\alpha_i = 0, \tag{1.41}$$

$$\nabla_{\xi_i} \mathcal{L}(w, b, \boldsymbol{\xi}, \boldsymbol{\alpha}) = 0 \implies C\xi_i - \alpha_i = 0. \tag{1.42}$$

If we apply these results in the Lagrangian, we get a simpler formula. Using (1.41) and (1.42), all the terms with $b$ or $\xi_i$ vanish. Then, we plug (1.40) to replace $w$ to get

$$\begin{aligned}\inf_{w,b,\boldsymbol{\xi}} \mathcal{L}(w, b, \boldsymbol{\xi}, \boldsymbol{\alpha}) &= \frac{1}{2C}\sum_{i=1}^{n}\alpha_i^2 + \frac{1}{2}\left\langle \sum_{i=1}^{n} y_i\alpha_i\phi(x_i), \sum_{i=1}^{n} y_i\alpha_i\phi(x_i)\right\rangle \\ &\quad - \sum_{i=1}^{n}\alpha_i\left[y_i\left\langle \sum_{j=1}^{n} y_j\alpha_j\phi(x_j), \phi(x_i)\right\rangle\right] - \frac{1}{C}\sum_{i=1}^{n}\alpha_i^2 + \sum_{i=1}^{n}\alpha_i p_i \\ &= -\frac{1}{2}\sum_{i=1}^{n}\sum_{j=1}^{n} y_i y_j\alpha_i\alpha_j\left\langle \phi(x_i), \phi(x_j)\right\rangle - \frac{1}{2C}\sum_{i=1}^{n}\alpha_i^2 + \sum_{i=1}^{n}\alpha_i p_i.\end{aligned}$$

We observe here that now the partial derivative with respect to $\xi$, given in (1.42), does not establish an upper bound for $\alpha_i$, but a new term with $\alpha_i^2$ appears in the dual objective function. We can define now the L2-SVM dual problem.

**Definition 1.42** (L2-SVM Dual Problem)**.** The dual problem corresponding to problem (1.29) is

$$\begin{aligned}\min_{\boldsymbol{\alpha}} \quad & \frac{1}{2}\boldsymbol{\alpha}^\intercal\left(Q + \frac{1}{C}I_n\right)\boldsymbol{\alpha} - \boldsymbol{\alpha}^\intercal\boldsymbol{p} \\ \text{s.t.} \quad & \sum_{i=1}^{n} y_i\alpha_i = 0, \\ & \alpha_i \geq 0.\end{aligned}$$

Observe that we again obtain a quadratic optimization problem, as in the L1-SVM case, where we do no longer have the box constraints, but we have a diagonal matrix that is added to the kernel matrix. That is, there is not an upper bound for $\alpha_i$, but the diagonal term can be interpreted as a soft constraint for the size of $\boldsymbol{\alpha}$.

### 1.6.2 LS-SVM

The LS-SVM (Suykens and Vandewalle, 1999) replaces the losses of the L1-SVM for the squared loss, even in classification, where the aim is to regress to the values 1 and $-1$ the positive and negative classes. To do this, the inequality constraints are replaced by equality ones. We give the definitions of the LS-SVM problems for classification and regression next.

**Definition 1.43** (Primal Problem of LS-SVM for Classification)**.** Given a binary classification sample

$$D = \{(x_i, y_i),\ i = 1, \ldots, n\},$$

where $x_i \in \mathbb{R}^d$ are the feature vectors and $y_i \in \{-1, 1\}$ are the class labels, the primal problem of the LS-SVM for binary classification is:

$$\begin{aligned} \min_{w,b,\boldsymbol{\xi}} \quad & \frac{C}{2}\sum_{i=1}^{n}(\xi_i)^2 + \frac{1}{2}\|w\|^2 \\ \text{s.t.} \quad & y_i(\langle w, \phi(x_i)\rangle + b) = 1 - \xi_i, \end{aligned} \tag{1.43}$$

where $\boldsymbol{\xi} = (\xi_1, \ldots, \xi_m)^\intercal$, and $w, b, \boldsymbol{\xi}$ are the primal variables.

**Definition 1.44** (Primal Problem of LS-SVM for Regression)**.** Given a regression sample

$$D = \{(x_i, t_i),\ i = 1, \ldots, n\},$$

where $x_i \in \mathbb{R}^d$ are the feature vectors and $t_i \in \mathbb{R}$ are the target values, the primal problem of the LS-SVM for regression is:

$$\begin{aligned} \min_{w,b,\boldsymbol{\xi}} \quad & \frac{C}{2}\sum_{i=1}^{n}(\xi_i)^2 + \frac{1}{2}\|w\|^2 \\ \text{s.t.} \quad & \langle w, \phi(x_i)\rangle + b = t_i - \xi_i, \end{aligned} \tag{1.44}$$

where $\boldsymbol{\xi} = (\xi_1, \ldots, \xi_m)^\intercal$, and $w, b, \boldsymbol{\xi}$ are the primal variables.

As with the other variants, we want to use a unifying formulation for the regression and classification problems.

**Definition 1.45** (LS-SVM Primal Problem)**.** Given a sample

$$D = \{(x_i, y_i, p_i),\ i = 1, \ldots, n\},$$

where $x_i \in \mathbb{R}^d$ are the feature vectors, $y_i = \{-1, 1\}$ and $p_i \in \mathbb{R}$, the primal problem is defined as

$$\begin{aligned} \min_{w,b,\boldsymbol{\xi}} \quad & \frac{C}{2}\sum_{i=1}^{n}(\xi_i)^2 + \frac{1}{2}\|w\|^2 \\ \text{s.t.} \quad & y_i(\langle w, x_i\rangle + b) = p_i - \xi_i, \end{aligned} \tag{1.45}$$

which can be interpreted as the classification primal LS-SVM problem (1.43) when

$$p_i = 1, \quad \text{for } i = 1, \ldots, n;$$

and can be interpreted as the regression primal LS-SVM problem (1.44) when

$$p_i = t_i,\ y_i = 1 \text{ for } i = 1, \ldots, n.$$

The corresponding Lagrangian to the primal problem (1.45) is

$$\mathcal{L}(w, b, \boldsymbol{\xi}, \boldsymbol{\alpha}) = \frac{C}{2}\sum_{i=1}^{n}(\xi_i)^2 + \frac{1}{2}\|w\|^2 - \sum_{i=1}^{n}\alpha_i\left[y_i(\langle w, \phi(x_i)\rangle + b) - p_i + \xi_i\right],$$

where $\boldsymbol{\alpha} \in \mathbb{R}$ are the Lagrange multipliers. Observe that we do no longer need the non-negativity of these variables, because they correspond to equality instead of inequality constraints. Now we use KKT conditions to find the solutions to the optimization problem. Recall that the KKT are sufficient conditions, and for problems with convex, differentiable objective function with affine constraints, they can be expressed as:

$$\nabla_w \mathcal{L}(w, b, \boldsymbol{\xi}, \boldsymbol{\alpha}) = 0 \implies w = \sum_{i=1}^{n} y_i\alpha_i\phi(x_i), \tag{1.46}$$

$$\nabla_b \mathcal{L}(w, b, \boldsymbol{\xi}, \boldsymbol{\alpha}) = 0 \implies \sum_{i=1}^{n} y_i\alpha_i = 0,$$

$$\nabla_{\xi_i} \mathcal{L}(w, b, \boldsymbol{\xi}, \boldsymbol{\alpha}) = 0 \implies C\xi_i - \alpha_i = 0, \tag{1.47}$$

$$\nabla_{\alpha_i} \mathcal{L}(w, b, \boldsymbol{\xi}, \boldsymbol{\alpha}) = 0 \implies y_i(\langle w, \phi(x_i)\rangle + b) - p_i + \xi_i = 0. \tag{1.48}$$

Observe that we do not have inequality constraints here, so $\alpha_i$ is used for the equality constraints, and we do not need the complementary slackness condition. Using now (1.46) and (1.47) to replace $w$ and $\xi_i$ in (1.48) we get

$$y_i\left(\left\langle\sum_{j=1}^{n} y_j\alpha_j\phi(x_j), \phi(x_i)\right\rangle + b\right) + \frac{1}{C}\alpha_i = p_i$$

for $i = 1, \ldots, n$. Therefore, we have a system of equations, which can be expressed as

$$\left[\begin{array}{c|c} 0 & \boldsymbol{y}^\intercal \\ \hline \boldsymbol{y} & Q + \frac{1}{C}I_n \end{array}\right]\begin{bmatrix} b \\ \boldsymbol{\alpha} \end{bmatrix} = \begin{bmatrix} 0 \\ \boldsymbol{p} \end{bmatrix},$$

where $Q$ is the same kernel matrix we have used in the other variants. Therefore, instead of being a constrained convex problem, by solving this linear system of equations, we obtain the optimal dual variable $\boldsymbol{\alpha}^*$, and thus, the solutions $w^*, b^*$ and $\boldsymbol{\xi}^*$ as well.

## 1.7 Conclusions

In this chapter, we have given definitions and concepts that we will use in the rest of this work. In Section 1.1 we have described the RKHSs, spaces with a reproducing kernel; we have seen how in these spaces we can apply the kernel trick to implicitly use non-linear transformations, hence, considering a much richer class of functions.

Next, in Section 1.2, we have given some of the most common loss functions and defined the empirical and expected risks. Also, we have presented the concept of regularization and regularized risk, which, in the case of kernel methods, has a closed form solution. To motivate the regularization risk, and to better understand the differences between empirical and expected risks, we have introduced in Section 1.3 the learning theory. Here, we have described how, by minimizing the empirical risk, we can expect to approximate a function that minimizes the expected risk. For these results, the VC-dimension is defined, which measures the capacity of a space of functions. Thus, as a simplification,

the smaller the VC-dimension is, the better the function minimizing the empirical risk generalizes to other samples. These concepts will be important for our discussion of Multi-Task Learning (MTL) advantages in Chapter 2.

To solve the regularized risk minimization problems, we have presented the optimization theory in Section 1.4, where we have described the concept of duality. In particular, given a convex objective function with affine constraints, there exists a unique solution, which we can obtain by solving the corresponding dual problem. The optimization problem related to the training of the SVM (also called L1-SVM) is one of such cases, as we have seen in Section 1.5. Here, we have shown how in this particular case we get the dual problem, and we have also analyzed the properties of the final model. Finally, in Section 1.6, we have presented two relevant variants of the standard SVM, the L2 and LS-SVM. In Chapters 3 and 4 we will present MTL models based on the L1, L2 and LS-SVM, where we will apply the optimization results given in this chapter.

Chapter 2

# Multi-Task Learning

In Machine Learning (ML), we typically try to minimize some loss metric that defines the performance on a single task. Given a data sample, we choose a model, select its hyperparameters and optimize the parameters of the model to achieve a minimal sample-dependent error. However, this process may seem too focused on the task at hand, since it ignores any other information that could be useful in the learning process, such as that of related tasks. Multi-Task Learning (MTL) aims at solving different related tasks simultaneously, so that the information of each task can leverage the learning process in the rest, thus achieving an overall greater generalization ability. This goal requires selecting which tasks should be learned together, that is, defining Multi-Task (MT) problems, and also designing algorithms that can benefit from the presence of different tasks. On the early stages of MTL (Baxter, 2000; Caruana, 1997), one motivation was data scarcity, since by combining different sources of information, this problem could be solved. Nowadays, in the age of "big data" this is not the main motivation, but other benefits can be extracted from MTL, such as bias mitigation, domain adaptation or the avoidance of overfitting.

The concept of an MT problem is not too precise because different definitions have been given 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. Multiple kind of problems can be faced with an MTL approach. The most common definition of MT 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 input space and $\mathcal{Y}$ is the output space, which can be $\mathbb{R}$ in the regression case or $\{0, 1\}$ in the binary classification one, and each task $r$ 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 only consider 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 MT problems. these are problems where each task has been sampled from a possibly different distribution, so we have different samples $(X_r, Y_r)$. Here, we can find the following cases:

  - MT single regression: This is the case where each task is a regression problem with a single target, e.g. $\mathcal{Y} = \mathbb{R}$.
  - MT binary classification: This is the case where each task is a binary classification problem, e.g. $\mathcal{Y} = \{0, 1\}$.
  - MT multi-target regression: 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-class classification: This is the case where each task is a multi-class classification problem, e.g. $\mathcal{Y} = \{1, \dots, C\}$, where $C$ is the number of classes.

- Non-pure MT problems: these are problems that can be seen as MT 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. $(X_r, Y_r) = (X, Y_r)$. Examples of such problems are:

  - multi-target regression: 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-class classification: 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.

In Table C.1 we give some of the most popular MTL problems, along with their characteristics and the main works that use them.

Even when we have an MT problem with tasks that are related and, hence, could be beneficial in the learning process of the others, it is still necessary to design algorithms that can exploit this additional information. The first approaches (Caruana, 1997) were focused on sharing a common representation, and, therefore, the models based on a Neural Network (NN) were the most commonly used. Other approaches with linear models present techniques based on matrix regularization, where some coupling between the parameters of each task is enforced. Kernel methods, however, are not as adaptable to the regularizers crafted for MTL, so different techniques, typically based on combinations of common and specific parts, are used. Independently of the method considered to exploit the information of different tasks, the mechanisms of the advantage obtained by using MTL have also been studied. These results show how incorporating different tasks can improve the generalization ability of the learning processes, and under what circumstances this is possible.

In this chapter, in Section 2.1, we present the fundamentals of MTL theory. We introduce the concept of Learning to Learn (LTL), a notion of task relatedness, and review some of the most relevant theoretical works in this area. In Section 2.2 we present a survey with some of the most important MTL approaches, and we also update the taxonomy presented in Zhang and Yang (2017) defining three main groups:

- feature-based methods,
- parameter-based methods, and
- combination-based methods.

Given the importance of Neural Networks (NNs) and kernel methods, we also present specific surveys of the MTL strategies in Section 2.3 and Section 2.4, respectively. Finally, related to MTL with kernel methods, specially the Support Vector Machine (SVM), in Section 2.5 we present the Learning Using Privileged Information (LUPI) paradigm and connect it with the MTL framework.

## 2.1 Why does Multi-Task Learning work?

To give an answer to how does MTL obtain its leverage, we present some theoretical results that provide a better insight. Typically, in ML 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 or classification. This expectation is taken using an unknown probability distribution that expresses how the data points and targets are distributed in the joint space (the product of the input and output spaces). 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), described in Section 1.2, 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 NN or the SVM, each considering a different space of hypotheses from which to choose a candidate. The definition of such space determines the bias for these models 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 to the original problem. In other words, in a single-task scenario there is no difference between learning the optimal hypothesis space (bias learning), and the 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. When we have a fixed known number of scenarios, this is called MTL, while the case where different unknown tasks can be sampled and the goal is to learn a good hypothesis space is called 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 a theoretical point of view, how combining multiple tasks can be helpful in the learning process. Finally, other important theoretical results on MTL are surveyed for completeness.

### 2.1.1 Multi-Task Learning and Learning to Learn

As explained below, in Baxter (2000) an effort is made to define the concepts needed to construct the theory about bias learning or LTL, which can be seen as a generalization of strict MTL. The problem of bias learning (or LTL) consists on finding not only the hypothesis in some hypothesis family that minimizes an empirical risk, but also learning

the best hypothesis space, which defines the bias of the problem, that can be taken for this purpose. Baxter defines 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, which have been presented in Chapter 1. In the ordinary statistical learning scenario, the following elements are 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* $\mathcal{H} = \{h(\cdot, \alpha), \alpha \in A\}$, where $A$ is a non-empty set, 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 $D = \{(x_1, y_1), \ldots, (x_n, y_n)\}$ of samples drawn from $P$. Then, the approach is to apply the ERM inductive principle, minimizing the empirical risk

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

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 $\hat{R}_D$ 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. As described in Chapter 1, 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 $\hat{R}_D$, 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 the convergence for any $\alpha \in A$ of

$$\lim_{n \to \infty} P\left(\sup_{h \in \mathcal{H}} \left(R_P(h) - \hat{R}_D(h)\right) > \epsilon\right) = 0.$$

This is the one-sided convergence, which is a necessary and sufficient condition for the consistency of the learning process, namely

$$\inf_{h \in \mathcal{H}} \hat{R}_D(h) \xrightarrow[n \to \infty]{P} \inf_{h \in \mathcal{H}} R_P(h).$$

Under some general conditions, as presented in (1.16), he proves that for $\delta > 0$, with probability $1 - \delta$, we have that

$$R_P(h) \leq \hat{R}_D(h) + \sqrt{\frac{8}{n}\left(\text{VCdim}\left(\mathcal{H}\right)\left(\log\left(\frac{n}{\text{VCdim}\left(\mathcal{H}\right)}\right) + 1\right) + \log\frac{4}{\delta}\right)}.$$

for all $h \in \mathcal{H}$. Observe that the right-most term is a non-decreasing function of $\text{VCdim}\left(\mathcal{H}\right)/n$, where $\text{VCdim}\left(\mathcal{H}\right)$ is the VC-dimension of $\mathcal{H}$. 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 of the hypothesis space $\mathcal{H}$, $\text{VCdim}\left(\mathcal{H}\right)$, which is desirable that it be small. This term is not intuitive and is the most important term in Vapnik's theory.

Recall that the VC-dimension measures the capacity of a set of hypotheses $\mathcal{H}$. In the case of a set of indicator functions, it is the maximum number of vectors $x_1, \ldots, x_{\text{VCdim}(\mathcal{H})}$ that can be shattered (in two classes) by functions of this set, and in the case of real functions, it is defined as the VC-dimension of the following set of indicator functions $I(x, \alpha, \beta) = \mathbf{1}_{\{h(x,\alpha)-\beta\}}$. If the capacity of the set $\mathcal{H}$ is too large, we may find a hypothesis $h\left(x, \alpha^*\right)$ that minimizes $\hat{R}_D$ but does not generalize well and, therefore, does not even remotely minimize $R_P$. This is the overfitting problem. On the other side, if we use a too simple $\mathcal{H}$, with low capacity, we could be in a situation where there is not a good hypothesis $h\left(x, \alpha\right) \in \mathcal{H}$, so the empirical risk $\inf_{\alpha \in A} R_P$ is too large. This is the underfitting problem.

In Baxter (2000) the goal is not to learn the optimal hypothesis $h\left(\cdot, \alpha^*\right)$ 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 or LTL 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* $\ell\left(\cdot, \cdot\right) : \mathcal{Y} \times \mathcal{Y} \to \mathbb{R}$; 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 LTL expected risk, defined as

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

for all $\beta \in B$. Observe that this risk is not a function of a specific hypothesis, but it depends on the selection of an entire 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 $D_r = \{(x_1^r, y_1^r), \ldots, (x_{m_r}^r, y_{m_r}^r)\}$ 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 $\boldsymbol{D} = \{(x_i^r, y_i^r), r = 1, \ldots, T,\ i = 1, \ldots, m\}$, with $m$ examples from the $T$ learning tasks, which can be expressed as

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

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

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

which is essentially a sum 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 $\hat{R}_{D_r}(h)$. Moreover, we cannot ensure the convergence of the empirical risk to the expected risk using the Chernoff inequality (Chernoff, 1952) as in the single-task case, so new results have to be developed to check this convergence. 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 $\hat{R}_{\boldsymbol{D}}(\mathcal{H}_\beta)$ 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 $\hat{R}_{\boldsymbol{D}}(\mathcal{H}_\beta) - R_Q(\mathcal{H}_\beta)$ for any $\beta$ using the capacity of a family of hypothesis spaces. If possible, finding also a bound for $\inf_{\beta \in B} \hat{R}_{\boldsymbol{D}}(\mathcal{H}_\beta) - \inf_{\beta \in B} R_Q(\mathcal{H}_\beta)$.

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 $\beta$ and write just $\mathcal{H}$ for simplicity.

### 2.1.2 VC-Dimension for Multi-Task Learning

Two pseudometrics are defined in the work of Baxter (2000) in order to derive notions of MTL capacities. Recall that given a space $\mathcal{X}$, a pseudometric of $\mathcal{X}$ is a real-valued function $d : \mathcal{X} \times \mathcal{X} \to \mathbb{R}_{\geq 0}$ such that

- $d(x, x) = 0$,
- $d(x, y) = d(y, x)$,
- $d(x, y) \leq d(x, z) + d(z, y)$.

Notice that, unlike the case of metrics, when we use pseudometrics $d(x, y) = 0$ does not imply $x = y$.

Going back to the pseudometrics of Baxter (2000), in first place a *sample-driven* pseudometric 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 2.1** (*Sample-driven* Pseudometric)**.** Given a $(T, 1)$-sample

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

considering the set of sequences of $T$ hypotheses

$$\mathcal{H}^T = \left\{h^T = (h_1, \ldots, h_T), h_1, \ldots, h_T \in \mathcal{H}\right\};$$

we can define then the set of $(T, 1)$-empirical risks, with one sample per task, as

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

Observe that in each element $h_\ell^T \in \mathcal{H}_\ell^T$ we use all the hypothesis $h_1, \ldots, h_T$ from the same hypothesis space $\mathcal{H}$. Combining the sets of risks for each hypothesis space we define $\mathbb{H}_\ell^T = \bigcup_{\mathcal{H}\in\mathbb{H}} \mathcal{H}_\ell^T$, and for any pair $h_\ell^T \in \mathcal{H}_\ell^T$ and $\widetilde{h_\ell^T} \in \widetilde{\mathcal{H}}_\ell^T$, 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 $\mathbb{H}_\ell^T$.

That is, this *sample-driven* pseudometric measures the expected difference between the $(T, 1)$-empirical risks obtained with $\mathcal{H}$ and $\widetilde{\mathcal{H}}$.

Then, a *distribution-driven* pseudometric is defined.

**Definition 2.2** (*Distribution-Driven* Pseudometric)**.** Given a distribution $P$ on $\mathcal{X} \times \mathcal{Y}$. Consider the infimum expected risk for each $\mathcal{H}$:

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

The set of such values is denoted as $\mathbb{H}_\ell^* = \{R_P^*(\mathcal{H}), \mathcal{H} \in \mathbb{H}\}$. Then, for $R_P^*(\mathcal{H}), R_P^*(\widetilde{\mathcal{H}}) \in \mathbb{H}_\ell^*$,

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

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

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

**Definition 2.3** ($\epsilon$-cover)**.** Given a pseudometric $d_\mathcal{X}$ in a space $\mathcal{X}$, a set of $l$ elements $x_1, \ldots, x_L \in \mathcal{X}$ is an $\epsilon$-cover of $\mathcal{X}$ if $\forall x \in \mathcal{X}$, $d_\mathcal{X}(x, x_i) \leq \epsilon$ for some $i = 1, \ldots, L$.

Let $\mathcal{U}(\epsilon, \mathcal{X}, d_{\mathcal{X}})$ denote the size $L$ of the smallest $\epsilon$-cover. Then, we can define the following capacities of a family space $\mathbb{H}$:

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

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 $0 < \eta < 1$, the LTL expected risk $R_Q(\mathcal{H})$ can be bounded as

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

with probability $1 - \eta$, where $\hat{R}_{\boldsymbol{D}}$ is the LTL empirical risk, given sufficiently large $T$ and $m$, namely

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

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 essentially inversely proportional to $T$, so the more tasks there are, the less data is necessary in each task.

The previous result is for pure bias learning or LTL, 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 $(\mathcal{X} \times \mathcal{Y})$. The goal is not learning a hypothesis space $\mathcal{H}$ but a sequence of hypotheses

$$\boldsymbol{h} = (h_1, \ldots, h_T),\ h_1, \ldots, h_T \in \mathcal{H}.$$

Thus, the MTL expected risk is

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

and the MTL empirical risk is defined as

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

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

$$R_{\boldsymbol{P}}(\boldsymbol{h}) \leq \hat{R}_{\boldsymbol{D}}(\boldsymbol{h}) + \epsilon$$

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

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

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.

To see how these bounds can be helpful, consider the MTL approach based on feature learning, where a common space of features is shared by all the tasks. The most popular example are NNs, 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 $\mathcal{F} = \{f : \mathcal{X} \to \mathcal{V}\}$. Using these features, models defining functions $g \in \mathcal{G}$ (which are typically 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 $\mathbb{H} = \{\mathcal{H}_f, f \in \mathcal{F}\}$. With the above definitions, Baxter also considers the functions

$$g_\ell : \mathcal{V} \times \mathcal{Y} \to \mathbb{R},$$

such that $g_\ell(f(x), y) = \ell(g(f(x)), y)$, and the class of such functions is called $\mathcal{G}_\ell$. The capacity of such space is defined as $C(\epsilon, \mathcal{G}_\ell) = \sup_P \mathcal{U}(\epsilon, \mathcal{G}_\ell, d_P)$. To define the capacity of $\mathcal{F}$, Baxter gives first the definition of the pseudo-metric

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

which "pulls back" the metric on $\mathbb{R}$ through $\mathcal{G}_\ell$. Thus, the capacity of $\mathcal{F}$ is defined as $C_{\mathcal{G}_\ell}(\epsilon, \mathcal{F}) = \sup_P \mathcal{U}(\epsilon, \mathcal{F}, d_{[P,\mathcal{G}_\ell]})$, the size of the smallest $\epsilon$-cover of $\mathcal{F}$.

Now, the bias learning problem is the problem of finding a good mapping $f$, that is, the selection of $f$ defines hypothesis space $\mathcal{H}_f$. 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

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

with $\epsilon = \epsilon_1 + \epsilon_2$. This means that the sample-driven capacity depends both on the capacity of the feature mapping functions $f$ and the decision functions $g$; however, the distribution-driven capacity can be bounded just with the capacity of the space of the feature mapping functions. Thus, the capacity of the space corresponding to the full models $g \circ f$ can be splitted in two parts, one corresponding to the feature mapping and other to the decision functions. These results, alongside those presented for bias learning, is useful to establish concrete bounds for feature learning models like NNs.

The results presented until now rely on the development of two capacities of a family of hypothesis spaces $\mathbb{H}$ to establish bounds in the difference $R_{\boldsymbol{P}}(\boldsymbol{h}) - \hat{R}_{\boldsymbol{D}}(\boldsymbol{h})$, that is, the probability of deviations between the MTL empirical and expected risks for a given hypothesis sequence $h$. 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 2.4.** 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, in Baxter (2000) is it shown that

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

and also that

$$\overline{d}(\mathbb{H}) \geq d_{\mathbb{H}}(T) \geq \underline{d}(\mathbb{H}). \tag{2.1}$$

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

**Theorem 2.5.** *Let $\boldsymbol{P} = (P_1, \dots, P_T)$ be a sequence on $(\mathcal{X} \times \{0,1\})^T$, and a let $\boldsymbol{D}$ be a sample from this distribution. Consider also a sequence $\boldsymbol{h} = (h_1, \dots, h_T)$ of Boolean hypothesis $h_i \in \mathcal{H}$. Then for every $\epsilon > 0$ and $0 < \nu < 1$*

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

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

$$m \geq \frac{88}{\epsilon^2} \left[ 2 d_{\mathbb{H}}(T) \log \frac{22}{\epsilon} + \frac{1}{T} \log \frac{4}{\eta} \right]. \tag{2.2}$$

Here, since $d_{\mathbb{H}}(T) \geq d_{\mathbb{H}}(T+1)$, it is easy to see that as the number of tasks $T$ 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

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

With these results we can see that the condition (2.2) has some important properties:

- It is a computable bound, assuming 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 MT Learners, i.e. uniform convergence of the best empirical risk to the best expected risk.

### 2.1.3 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, that is, the hypothesis of all tasks are elements of the same hypothesis space $\mathcal{H}$, and the bias learning consists on selecting the best $\mathcal{H}$ from a family of hypothesis spaces $\mathbb{H}$. In Ben-David and Borbely (2008), nevertheless, 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 MT 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.

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

**Definition 2.6** ($\mathcal{F}$-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 $\Omega$. Consider a probability distribution $P$ over this product space and a function $f:\mathcal{X}\to\mathcal{X}$, then we define the distribution $f[P]$ such that for any $S\in\mathcal{A}\times\wp(\{0,1\})$,

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

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 consider 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 $\mathcal{H}_{\mathcal{F}}$ is a hypothesis space family, since it is a set of equivalence classes $[h]$, which are sets of hypotheses. Using the formulation from the previous section, $\mathcal{H}_{\mathcal{F}} = \mathbb{H}$ and $[h] \in \mathbb{H}$.

This equivalence classes are useful to divide the learning process in two stages; this is what the authors name the MT ERM. Also, here our goal is to find bounds for one specific task, the target one, while the rest of the tasks help in the learning process of such target task. Consider the samples $D_1, \ldots, D_T$ from $T$ different tasks; then we can proceed as follows:

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

$$[h^{\mathcal{F}}] = \min_{[h] \in \mathcal{H}_{\mathcal{F}}} \inf_{h_1, \ldots, h_T \in [h]} \frac{1}{T} \sum_{r=1}^{T} \hat{R}_{D_r}(h_r). \tag{2.3}$$

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

$$h^{\diamond} = \inf_{h \in [h^{\mathcal{F}}]} \hat{R}_{D_1}(h).$$

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 $f \in \mathcal{F}$ generates 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 $P$ 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 $P$. 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.

With this MT ERM the transfer learning between tasks can be found in the selection of the hypothesis class $[h^{\mathcal{F}}]$; this is a bias learning problem where a hypothesis space that is optimal for all tasks, $[h^{\mathcal{F}}]$, is selected in the first stage. Next, from this space the hypothesis optimal for the target task is taken. We will see below how this strategy can improve the learning process; in short, with MT ERM the bound for the difference between the expected and empirical risks can be tighter than that obtained with standard ERM.

### 2.1.4 Bounds for Related Tasks

The results of Theorem 2.5 can be applied to the hypothesis quotient space of equivalent classes $\mathcal{H}_{\mathcal{F}}$. However, we first need the following result. Let $P_1, P_2$ be $\mathcal{F}$-related distributions; then, for any hypothesis space $\mathcal{H}$, it can be proved (Ben-David and Borbely,

2008, Lemma 2) that

$$\inf_{h\in\mathcal{H}} R_{P_1}(h) = \inf_{h\in\mathcal{H}} R_{P_2}(h). \tag{2.4}$$

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 2.7.** *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 $\boldsymbol{P} = (P_1, \dots, P_T)$ a sequence of $\mathcal{F}$-related distributions over $\mathcal{X} \times \{0, 1\}$, and $\boldsymbol{D} = (D_1, \dots, D_T)$ the corresponding sequence of samples where $D_i$ is sampled using $P_i$. Then for every $[h] \in \mathcal{H}_{\mathcal{F}}$ and $\epsilon > 0$ and $0 < \eta < 1$,*

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

*with probability at least $1 - \eta$ if the number of samples from each distribution satisfies*

$$m_r \geq \frac{88}{\epsilon^2}\left[2d_{\mathcal{H}_{\mathcal{F}}}(T)\log\frac{22}{\epsilon} + \frac{1}{T}\log\frac{4}{\eta}\right], \tag{2.5}$$

*where $m_r$ is the size of the sample $D_r$.*

Note that, in contrast to Theorem 2.5, this result bounds the expected risk of a single task, not the average risk. This is the consequence of applying Theorem 2.5 and replacing the average empirical error using the result from (2.4) for $\mathcal{F}$-related tasks. 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 MT ERM principle is given (Ben-David and Borbely, 2008, Theorem 3).

**Theorem 2.8.** *Define $\mathcal{F}$ and $\mathcal{H}$ as in the previous theorem. Consider also the previous sequence of distributions $(P_1, \dots, P_T)$ and corresponding samples $(D_1, \dots, D_T)$. Let $\underline{d}(\mathcal{H}_{\mathcal{F}}) = \max_{h\in\mathcal{H}} \mathrm{VCdim}([h])$, and $h^\diamond$ be the hypothesis selected using the MT ERM principle. Then, for every $\epsilon_1, \epsilon_2 > 0$ and $0 < \eta < 1$:*

$$\hat{R}_{D_1}(h^\diamond) \leq \inf_{h'\in\mathcal{H}} R_{P_1}(h') + 2(\epsilon_1 + \epsilon_2)$$

*with probability greater than $1 - \eta$ if*

$$m_1 \geq \frac{64}{\epsilon^2}\left[2\underline{d}(\mathcal{H}_{\mathcal{F}})\log\frac{12}{\epsilon} + \frac{1}{T}\log\frac{8}{\eta}\right], \tag{2.6}$$

*and for $r \neq 1$*

$$m_r \geq \frac{88}{\epsilon^2}\left[2d_{\mathcal{H}_{\mathcal{F}}}(T)\log\frac{22}{\epsilon} + \frac{1}{T}\log\frac{8}{\eta}\right]. \tag{2.7}$$

*Here we have considered, without loss of generality, that the first task is the target task.*

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 MT ERM principle, we select the hypothesis class $[h^{\mathcal{F}}]$ as in (2.3), such that for all $h \in \mathcal{H}$ it satisfies

$$\inf_{h_1,\dots,h_T\in[h^{\mathcal{F}}]} \frac{1}{T}\sum_{r=1}^{T} \hat{R}_{D_r}(h_r) \leq \inf_{h_1,\dots,h_T\in[h]} \frac{1}{T}\sum_{r=1}^{T} \hat{R}_{D_r}(h_r).$$

According to Theorem 2.7, with probability greater than $1-\eta/2$ we have that

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

and also applying Theorem 2.7, with probability greater than $1-\eta/2$,

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

Combining these two inequalities we get

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

when $m_r$ is large enough for every task, as in the condition (2.5). This bounds the risk of the hypothesis space given by the equivalence class of $h^{\mathcal{F}}$ and establishes the inequality (2.7) as a requirement.

Once we select $[h^{\mathcal{F}}]$ as our hypothesis space, the second stage is just standard ERM using this hypothesis space. According to Vapnik (1982), given any hypothesis space $\mathcal{H}$, we have the bound

$$\inf_{h\in\mathcal{H}} R_{D_1}(h) \leq \inf_{h\in\mathcal{H}} R_{P_1}(h) + 2\epsilon_2$$

if the size of the sample $D_1$ satisfies

$$m_1 \geq \frac{64}{\epsilon_2^2}\left[2\,\mathrm{VCdim}\left(\mathcal{H}\right)\log\frac{12}{\epsilon_2} + \frac{1}{T}\log\frac{8}{\eta}\right].$$

In this second stage of the MT ERM, the hypothesis space is $\mathcal{H} = [h^{\mathcal{F}}]$, so we have that

$$\inf_{h\in[h^{\mathcal{F}}]} R_{P_1}(h) \leq \inf_{h\in[h^{\mathcal{F}}]} R_{D_1}(h) + 2\epsilon_2$$

if

$$m_1 \geq \frac{64}{\epsilon_2^2}\left[2\,\mathrm{VCdim}\left([h^{\mathcal{F}}]\right)\log\frac{12}{\epsilon_2} + \frac{1}{T}\log\frac{8}{\eta}\right].$$

Recall that, according to Definition 2.4,

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

thus, by definition we have that

$$\mathrm{VCdim}\left([h^{\mathcal{F}}]\right) \leq \underline{d}(\mathcal{H}_{\mathcal{F}}). \tag{2.8}$$

That is, with the inequality (2.8), we can bound $m_1$ as in (2.6).

The advantage of using multiple tasks is then illustrated in this bound, and it will be defined by the gap between $\mathrm{VCdim}\left(\mathcal{H}\right)$ and $\underline{d}(\mathcal{H}_{\mathcal{F}})$. Observe that using standard ERM with no information about the tasks relations, a learner would have to select the hipothesis from the entire space $\mathcal{H}$; however, in this MT ERM we restrict our search to the space $[h^{\mathcal{F}}]$. If $\underline{d}(\mathcal{H}_{\mathcal{F}})$ is smaller than $\mathrm{VCdim}\left(\mathcal{H}\right)$, the number of samples needed to solve the target task will also be smaller.

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 entire set of hypotheses, is what marks the difference between single task and MTL.

As we have seen in Theorem 2.8, the definitions $\text{VCdim}(\mathcal{H})$, $\underline{d}(\mathcal{H}_{\mathcal{F}})$ and $d_{\mathcal{H}_{\mathcal{F}}}(T)$, given in Definition 2.4, are crucial for stating the advantage of MTL over STL. To understand better how these concepts interact, Ben-David *et al.* gave some theoretical results. Recall that, given a hypothesis space $\mathcal{H}$, we have defined $\mathcal{H}_{\mathcal{F}}$ as the family of hypothesis spaces composed by the hypothesis spaces $[h], h \in \mathcal{H}$. That is, our hypothesis space family is now $\mathbb{H} = \mathcal{H}_{\mathcal{F}}$, and the elements of this family are the hypothesis spaces $\mathcal{H} = [h]$. Therefore, the definitions of Theorem 2.8 can be expressed in this case as:

$$\begin{aligned} d_{\mathcal{H}_{\mathcal{F}}}(T) &= \max_{\left\{m, \Pi_{\mathcal{H}_{\mathcal{F}}} = 2^{Tm}\right\}} m, \\ \underline{d}(\mathcal{H}_{\mathcal{F}}) &= \max_{[h] \in \mathcal{H}_{\mathcal{F}}} \text{VCdim}\left([h]\right), \\ \overline{d}(\mathcal{H}_{\mathcal{F}}) &= \text{VCdim}\left(\bigcup_{[h] \in \mathcal{H}_{\mathcal{F}}} [h]\right) = \text{VCdim}\left(\mathcal{H}\right). \end{aligned}$$

Using the result from (2.1) we observe that

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

That is, the best we can hope when bounding $m_r$ as given in (2.7) of Theorem 2.8 is $\underline{d}(\mathcal{H}_{\mathcal{F}}) = d_{\mathcal{H}_{\mathcal{F}}}(T)$. Ben-David *et al.* presented evidence that, with some restrictions on $\mathcal{H}$, this lower bound can be achieved (Ben-David and Borbely, 2008, Theorem 4).

**Theorem 2.9.** *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 restrictive, it can also be proved that the upper limit of $d_{\mathcal{H}_{\mathcal{F}}}(T)$, that is, $\text{VCdim}(\mathcal{H})$, under some conditions on $\mathcal{F}$ is not achieved. The following result (Ben-David and Borbely, 2008, Theorem 6) shows this.

**Theorem 2.10.** *If $\mathcal{F}$ is finite and $\frac{T}{\log(T)} \geq \text{VCdim}(\mathcal{H})$, then*

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

This inequality indicates that, given a finite set of transformations $\mathcal{F}$, there are scenarios when $\text{VCdim}(\mathcal{H})$ is arbitrarily large but $d_{\mathcal{H}_{\mathcal{F}}}(T)$ is bounded, and therefore the right-hand side of inequality (2.7) is also bounded. That is, the MT bound, which substitutes $\text{VCdim}(\mathcal{H})$ by $d_{\mathcal{H}_{\mathcal{F}}}(T)$, is a better one in these cases.

## 2.2 Multi-Task Learning Methods: An Overview

The theory of MTL serves as motivation and gives some clues on how to use the data of different, related tasks to improve the learning process. However, it is necessary to design specific algorithms that implement mechanisms to exploit the information of multiple tasks. In this section a general overview of MTL methods will be given, categorizing some of the most relevant approaches. Recall that in this work we define an MTL sample as

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

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

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

is minimized, where $\Omega$ is some regularizer of the hypotheses. We propose a taxonomy for MTL algorithms, which enforce a coupling between tasks, with three main groups: feature-based, parameter-based and combination-based strategies. The feature-based approaches define the hypotheses $h_r$ as the composition of a common feature-building function $f$ and task-specific functions $g_r$, 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 approaches consider 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 frameworks, MTL with neural networks and MTL with kernel methods will be treated separately in their specific subsections.

### 2.2.1 Feature-based Multi-Task Learning

As explained above, 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.

#### 2.2.1.1 Feature Learning

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 $\Theta$ 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 Section 2.3.

With linear or kernel models, unlike the case with NNs, the function $f$ is modeled as linear combinations of fixed features defined by $\phi$, for example, $f(x_i^r) =$

$(u_1^\intercal \phi(x_i^r), \dots, u_k^\intercal \phi(x_i^r))$ for some $k \in \mathbb{N}_{>0}$, where $\phi$ is the implicit transformation, which is the identity in the case of linear models. Then, for $r = 1, \dots, T$, $h_r(x_i^r) = a_r^\intercal(U^\intercal \phi(x_i^r))$, 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 these models as $h_r(x_i^r) = w_r^\intercal x_i^r$, with $w_r = Ua_r$ and $U = (u_1, \dots, u_k)$. This idea, named MT 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 $W = (w_1, \dots, w_T)$ 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 $u_1, \dots, u_T$ capture different information and only a subset of them is necessary for each task. The minimization problem to obtain the optimal model parameters is

$$\underset{\substack{U\in\mathbb{R}^{d\times d}\\ A\in\mathbb{R}^{d\times T}}}{\arg\min} \sum_{r=1}^{T}\sum_{i=1}^{m_r} \ell(y_i^r, \langle Ua_r, x_i^r\rangle) + \lambda \|A\|_{2,1}^2 \text{ s.t. } U^\intercal U = I, \tag{2.9}$$

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 (2.9) 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

$$\begin{aligned}&\underset{W\in\mathbb{R}^{d\times T}, D\in\mathbb{R}^{d\times d}}{\arg\min} \sum_{r=1}^{T}\sum_{i=1}^{m_r} \ell(y_i^r, \langle w_r, x_i^r\rangle) + \lambda \sum_{r=1}^{T} \langle w_r, D^{-1} w_r\rangle \\ &\text{s.t. } D \succeq 0,\ \operatorname{tr}(D) \le 1.\end{aligned} \tag{2.10}$$

If $(A^*, U^*)$ is an optimal solution of (2.9), then

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

where $a^i$ are the rows of $A$, is an optimal solution of problem (2.10); conversely, given an optimal solution $(W^*, D^*)$ of (2.10), if $U^*$ has as columns an orthonormal basis of eigenvectors of $D^*$ and $A^* = (U^*)^\intercal W^*$, $(A^*, U^*)$ is an optimal solution of (2.9). To obtain an optimal solution $(W^*, D^*)$ the authors, for a better stability, use a modified problem

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

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

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

where the identity matrix, which improves the stability, is a consequence of the addition of the term $\operatorname{tr}\left(D^{-1}\right)$ in the objective function. It is interesting to observe that the regularizer of (2.10) can be expressed as $\operatorname{tr}\left(W^\intercal D^{-1} W\right)$ and by plugging $D^*$ in this formula, when $\mu = 0$, we obtain the squared-trace norm regularizer for $W$:

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

Here, $\|W\|_* = \operatorname{tr}\left((W^\intercal W)^{\frac{1}{2}}\right)$ denotes the trace norm, also known as nuclear norm. The trace norm can be seen as the continuous envelope of the rank since it can be expressed as

$$\|W\|_* = \sum_{i=1}^{\min(d,T)} \lambda_i,$$

where $\lambda_i$ are the eigenvalues of $W$. Therefore, by penalizing the trace norm, we favor low-rank solutions of $W$. That is, the initial problem (2.9) 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 $A = V^\intercal \operatorname{diag}(\lambda_1, \ldots, \lambda_d) V$ is given in terms of a scalar function $f$ that is applied over its eigenvalues:

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

Then, since the trace is invariant under cyclic permutations, a generalized regularizer for problem (2.10) can be expressed as

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

It is easy to see that problem (2.10) is a particular case of this spectral regularization where $f(\lambda) = \lambda^{-1}$. In Maurer (2009) some bounds on the excess risks are given for this MT 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 parameters $w_r$ lie in a manifold $\mathcal{M}$. The resultant optimization problem to train the model is

$$\underset{W, \mathcal{M}, \boldsymbol{b}}{\arg\min} \sum_{r=1}^{T} \sum_{i=1}^{m_r} \ell(y_i^r, \langle w_r, x_i^r \rangle) + \lambda \sum_{r=1}^{T} \mathcal{P}_{\mathcal{M}}(w_r), \tag{2.11}$$

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 use the definition $w_r = U a_r$ as in (2.9), the manifold corresponding to the linear subspace $\operatorname{span}(u_1, \ldots, u_k)$ would be at zero distance to $w_r$ for all $r = 1, \ldots, T$. Thus, in (2.9) we have a hard constraint with a linear manifold and in (2.11) we use a soft constaint with a non-linear manifold. Again, an approximation of (2.11) is considered to obtain a convex problem, and it is solved with 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 applied for MTL. Recall that we use $d$ for the dimension of the original feature space; then the

problem presented by Maurer *et al.* is

$$\underset{D\in\mathcal{D}_k, A\in\mathbb{R}^{k\times T}}{\arg\min} \sum_{r=1}^{T}\sum_{i=1}^{m_r} \ell(y_i^r, \langle Da_r, x_i^r\rangle) + \lambda \|D\|_{2,\infty} + \mu \|A\|_{1,\infty} . \tag{2.12}$$

Here, $\mathcal{D}_k$ is a set of $k$-dimensional dictionaries and every $D \in \mathcal{D}_k$ is a linear map $D : \mathbb{R}^d \to \mathcal{H}$ 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 (2.9) and (2.12) share a similar form, there are crucial differences. The matrix $U$ in (2.9) is an orthogonal square matrix, while the matrix $D$ of (2.12) is overcomplete with $k$ columns and $k > d$. Also, in problem (2.9) non-linear features can be used, as we will see later when we review MTL with kernel methods, while problem (2.12) is linear. A problem very similar to (2.12) is presented in Kumar and Daumé (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 $A$:

$$\underset{D\in\mathcal{D}_k, A\in\mathbb{R}^{k\times T}}{\arg\min} \sum_{r=1}^{T}\sum_{i=1}^{m_r} \ell(y_i^r, \langle Da_r, x_i^r\rangle) + \lambda \|D\|_{2,2} + \mu \|A\|_{1,1} . \tag{2.13}$$

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

#### 2.2.1.2 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 MT Feature Selection use an $L_{p,q}$ regularization of the weight matrix $W$. The first work using this idea is Obozinski et al. (2006), where the authors solve the problem

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

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) $L_{\infty,1}$ 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 $\theta$ enables a more flexible regularization: with small values of $\theta$, the smallest rows of $W$ are pushed towards zero since the rows with norms larger than $\theta$ do not dominate the sum. When $\theta$ grows to infinity, this penalty will converge 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 $\Theta = \operatorname{diag}\left(\theta^1, \ldots, \theta^d\right)$, they define the problem

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

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 (2.12) but with $D$, the "feature building" matrix, being limited to a diagonal matrix since it is acting here as a selection matrix:

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

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.

### 2.2.2 Parameter-based Multi-Task Learning

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 $\phi$ is a fixed, non-learnable transformation. Since we have one vector $w_r$ for each task, we can consider the matrix $W$ whose columns are $w_r$ for $r = 1, \ldots, T$. Some approaches rely on the assumption that the MT 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 sum of multiple matrices. We summarize each approach below.

#### 2.2.2.1 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

$$\underset{\Theta \in \mathbb{R}^{k \times d}, \boldsymbol{u}, \boldsymbol{v}}{\arg\min} \sum_{r=1}^{T} \sum_{i=1}^{m_r} \ell(y_i^r, \langle u_r + \Theta^\intercal v_r, x_i^r \rangle) + \lambda \sum_{r=1}^{T} \|u_r\|^2 \text{ s.t. } \Theta\Theta^\intercal = I_k, \tag{2.14}$$

where $\boldsymbol{u}^\intercal = (u_1^\intercal, \ldots, u_T^\intercal)$ and analogously for $\boldsymbol{v}$. Observe that this problem shares some similarities with (2.9). However, this is a more flexible approach, since the vectors $u_r$

allow for deviations of the task parameters from the shared subspace. Problem (2.14) can be reformulated as

$$\underset{\Theta\in\mathbb{R}^{k\times d},\boldsymbol{w},\boldsymbol{v}}{\arg\min} \sum_{r=1}^{T}\sum_{i=1}^{m_r} \ell(y_i^r, \langle w_r, x_i^r\rangle) + \lambda \sum_{r=1}^{T} \|w_r - \Theta^\intercal v_r\|^2 \text{ s.t. } \Theta\Theta^\intercal = I_k. \tag{2.15}$$

Here the terms $\|w_r - \Theta^\intercal v_r\|^2$ enforce the similarity across tasks by bringing them closer to the shared subspace. Problem (2.15) is solved using a two-step optimization, iterating between minimizing in $\{\Theta, \boldsymbol{v}\}$ and minimizing in $\boldsymbol{w}$. In Chen et al. (2009) the following extension is proposed:

$$\underset{\Theta\in\mathbb{R}^{k\times d},\boldsymbol{w},\boldsymbol{v}}{\arg\min} \sum_{r=1}^{T}\sum_{i=1}^{m_r} \ell(y_i^r, \langle w_r, x_i^r\rangle) + \lambda \sum_{r=1}^{T} \|w_r - \Theta^\intercal v_r\|^2 + \mu \sum_{r=1}^{T} \|w_r\|^2 \text{ s.t. } \Theta\Theta^\intercal = I_k. \tag{2.16}$$

Moreover, it is shown that (2.16), 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, indirectly 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 $\theta$ and also converges to the trace norm for large enough $\theta$.

#### 2.2.2.2 Task-Relation Learning approaches

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

One of the first works with the goal of explicitly modelling the pairwise task-relations is Bonilla et al. (2007), where an MT Gaussian Process (GP) formulation is presented. Since the Bayesian formulations are not the main focus of this work, we will just present a general idea of how these are used in the MTL framework. In one of the first works of Bayesian MTL, 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:

$$\boldsymbol{f} \sim N\left(\mathbf{0}_{dT}, K^f \otimes K_\theta^{\mathcal{X}}\right), \tag{2.17}$$

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, that is

$$A \otimes B = \begin{bmatrix} a_{11} & \ldots & a_{1m} \\ \vdots & \ddots & \vdots \\ a_{n1} & \ldots & a_{nm} \end{bmatrix} \otimes B = \begin{bmatrix} a_{11}B & \ldots & a_{1m}B \\ \vdots & \ddots & \vdots \\ a_{n1}B & \ldots & a_{nm}B \end{bmatrix},$$

so $\text{Cov}(f_r(x), f_s(x')) = K^f_{rs} k^{\mathcal{X}}_\theta(x, x')$. 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-feature covariance and inter-task 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 MT GP, they consider linear models $f(x^r_i) = \langle w_r, x^r_i \rangle + b_r$ and the prior on matrix $W = (w_1 \ldots, w_T)$ is defined as

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

where $\mathcal{M}_{\mathcal{N}}(M, A \otimes B)$ denotes the matrix-variate normal distribution with mean $M$, row covariance matrix $A$ and column covariance matrix $B$. Thus, in this case the feature covariance matrix is the identity matrix, while the task-covariance matrix is given by $OMEGA$. It is shown that the problem of selecting the maximum a posteriori estimation of $W$ and the maximum likelihood estimations of $\Omega$ and biases $b_r$, $r = 1, \ldots, T$, is a regularized minimization problem that, when relaxing the restrictions on $\Omega$, can be expressed as

$$\begin{aligned} &\underset{\Omega\in\mathbb{R}^{d\times T}, W, \boldsymbol{b}}{\arg\min} \sum_{r=1}^{T}\sum_{i=1}^{m_r} \ell(y^r_i, \langle w_r, x^r_i\rangle + b_r) + \lambda \sum_{r=1}^{T} \|w_r\|^2 + \mu \sum_{r,s=1}^{T} \left(\Omega^{-1}\right)_{rs} \langle w_r, w_s\rangle \\ &\text{s.t. } \Omega \succeq 0, \operatorname{tr}\Omega = 1. \end{aligned}$$

Since these are kernel methods, we will come back to them later when we review the MTL methods with kernels.

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

$$\underset{W,\boldsymbol{b}}{\arg\min} \sum_{r=1}^{T}\sum_{i=1}^{m_r} \ell(y^r_i, \langle w_r, x^r_i\rangle + b_r) + \mu \sum_{r,s=1}^{T} (L)_{rs} \langle w_r, w_s\rangle, \tag{2.18}$$

where $\mu$ is a tuning parameter. Here the regularization can be also written using the adjacency matrix $A$ of the graph 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 (2.18) can be solved in the dual

space using a Multi-Task kernel

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

For the second step, in Zhang and Yeung (2010) the authors propose the use of the trace norm to enforce a low-rank task-covariance matrix $\Omega$, 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:

$$\begin{aligned} &\arg\min_{\boldsymbol{\alpha}, L} \boldsymbol{\alpha}^\intercal K_L \boldsymbol{\alpha} + \nu \boldsymbol{\alpha}^\intercal \boldsymbol{y} + \operatorname{tr}(L + \lambda I)^{-1} \\ &\text{s.t. } 0 \preceq L,\ (L + \lambda I)_{\text{off}} \leq 0,\ (L + \lambda I)^{-1} \mathbf{1}_n = \frac{1}{\lambda} \mathbf{1}_n, \end{aligned} \tag{2.19}$$

where $A_{\text{off}}$ denotes the off diagonal entries of $A$ and $\mathbf{1}_n$ is the vector of $n$ ones. The restrictions of problem (2.19) are to ensure that $L$ is a valid graph Laplacian: it is positive definite, with non-positive terms outside of the 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 $\boldsymbol{\alpha}$ 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

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

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.

#### 2.2.2.3 Task Clustering Approaches

The task clustering approaches try 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

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

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, that is $\boldsymbol{\omega}_r^* = \arg\min_{\boldsymbol{\omega}_r} A_r(\boldsymbol{\omega}_r)$ with

$$A_r(\boldsymbol{\omega}_r) = \sum_{i=1}^{m_r} (y_i^r y_i^r) \operatorname{dist}_{\boldsymbol{\omega}_r}(x_i^r, x_i^r),$$

where $y_i^r \in \{-1, 1\}$ is the class of pattern $x_i^r$. 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 $B_\kappa$ 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 $C \leq T$ must be selected beforehand. That is, the clusters are selected using the results of independently trained tasks but using distances chosen to be optimal for other tasks. After grouping the tasks the clusters, all the tasks in the same cluster are fitted together using a distance parametrized by

$$\boldsymbol{\omega}_{B_\kappa} = \arg\min_{\boldsymbol{\omega}} \sum_{r \in B_\kappa} A_r(\omega),$$

which is optimal for the entire cluster.

Taking a different perspective, 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 $R = \frac{1}{T}\mathbf{1}\mathbf{1}^\intercal$, $E$ the $T \times C$ cluster assignment binary matrix, and defining the adjacency matrix $M = E(E^\intercal E)^{-1}E^\intercal$, the problem is

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

where $\Omega_{\mathrm{m}} = \operatorname{tr}(WRW^\intercal)$ is the mean regularization, $\Omega_{\mathrm{b}} = \operatorname{tr}(W(M-R)W^\intercal)$ is the inter-cluster variance regularization and $\Omega_{\mathrm{w}} = \operatorname{tr}(W(I-M)W^\intercal)$ 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 $Q_\kappa$ are $T \times T$ binary, diagonal matrices where the $r$-th element of the diagonal indicates whether task $r$ corresponds to cluster $\kappa$. They consider the problem

$$\begin{aligned} \arg\min_{W, Q_1, \ldots, Q_C, \boldsymbol{b}} \quad & \sum_{r=1}^{T} \sum_{i=1}^{m_r} \ell(y_i^r, \langle w_r, x_i^r \rangle + b_r) + \lambda \sum_{\kappa=1}^{C} \|WQ_\kappa\|_*^2 \\ \text{s.t.} \quad & \sum_{\kappa=1}^{C} Q_\kappa = I \text{ with } Q_{\kappa t} \in \{0, 1\}. \end{aligned} \tag{2.20}$$

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 MT feature learning (Argyriou et al., 2006, 2008), shown in Equation (2.9), but instead of assuming that all tasks share the same subspace, only the tasks in the same cluster do. The optimization of matrices $Q_\kappa$ on problem (2.20) is done by relaxing the binary nature

of matrices $Q_\kappa$ such that $0 \leq Q_{\kappa t} \leq 1$, 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 $f_\kappa$ are fitted using the pooled data from tasks in cluster $\kappa$. Secondly each task $r$ is assigned to the cluster $\kappa$ whose function $f_\kappa$ 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 (2012). In this work the weight 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 $\|G_r\|^2 = 1$, where $G_r$ is the $r$-th column of matrix $G$. The corresponding optimization problem is

$$\begin{aligned} &\underset{D \in \mathbb{R}^{d \times C}, G \in \mathbb{R}^{C \times T}}{\arg\min} \sum_{r=1}^{T} \sum_{i=1}^{m_r} \ell(y_i^r, \langle Dg_r, x_i^r \rangle) + \lambda \|D\|_{2,1} \\ &\text{s.t. } G_{\kappa r} \in [0,1], \ \|G_r\|^2 = 1, \end{aligned} \tag{2.21}$$

where the constraints on $G_{\kappa r}$ have been relaxed to be in the $[0,1]$ interval in order to make problem (2.21) convex. Observe that (2.21) is similar to (2.13) but different restrictions are used to ensure that $G$ can be seen as a clustering assignment matrix.

#### 2.2.2.4 Decomposition Approaches

The decomposition approaches consider too restrictive for real world scenarios 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. The idea 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

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

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, namely $g(U) = \|U\|_*$ 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 the sum of $L$ matrices, that is, $W = \sum_{l=1}^{L} W_l$; then, the problem to solve has the form

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

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.

### 2.2.3 Combination-based Multi-Task Learning

The combination-based methods use a combination of task-specific parts 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 Chapters 3 and 4. The original problem presented in Evgeniou and Pontil (2004), which is linear, is

$$\underset{w,V}{\arg\min}\, C \sum_{r=1}^{T}\sum_{i=1}^{m_r} \ell(y_i^r, \langle w + v_r, x_i^r\rangle) + \frac{\mu}{2}\|w\|^2 + \frac{1}{2}\sum_{r=1}^{T}\|v_r\|^2, \tag{2.22}$$

where $\mu$ is a trade-off parameter to leverage the common and task-specific information, and $C$ is the hyperparameter to control the relevance of the error term against the regularization terms. When the dual problem of (2.22) is obtained, the result is a problem where the kernel encodes this combination of common and shared parts; that is, the kernel function is

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

where $\delta_{rs}$ is 1 if $r = s$ and 0 otherwise, so it encodes the specific part of the model. Observe that both the common and task-specific parts are linear. This is equivalent to a feature extension strategy, like the domain adaptation scheme proposed in Daumé (2007) with only target and source domains, but that can be easily adapted to MTL.

## 2.3 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 (Krizhevsky et al., 2012), Generative Adversarial Networks for generative models (Goodfellow et al., 2014) or Transformers for Natural Language Processing (Vaswani et al., 2017). 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 MTL 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 the matrix $A^{-1}$:

$$w_r(\tau + 1) = w_r(\tau) - \nu y_i^s A_{rs}^{-1} x_i^s, \tag{2.23}$$

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

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

where $\mathcal{T}_{\mathcal{N}}$ 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 $\ell$ is the loss function, we can observe some similarities with the update (2.23), 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 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 1 and 2, these values are combined as

$$\begin{bmatrix} \tilde{h}_1^l \\ \tilde{h}_2^l \end{bmatrix} = A^l \begin{bmatrix} h_1^l \\ h_2^l \end{bmatrix} = \begin{bmatrix} \alpha_{11}^l & \alpha_{12}^l \\ \alpha_{21}^l & \alpha_{22}^l \end{bmatrix} \begin{bmatrix} h_1^l \\ h_2^l \end{bmatrix}, \tag{2.24}$$

where the $2 \times 2$ 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 $\alpha_{rs}^l$ 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: the 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 uses 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 (2.24), $A$ is a $4 \times 4$ matrix, because each $\alpha_{i,j}^l$ is a $2 \times 2$ 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 $L_{2,2}$ norm is used, so the parameters matrices $W_{r,a}^l$ and $W_{r,b}^l$ span orthogonal spaces; observe that this regularization forces the inner products $\left\langle (W_{r,a}^l)_i, (W_{r,b}^l)_j \right\rangle$ to go towards zero, with $(W_{r,a}^l)_i$ being the $i$-th column of the matrix $(W_{r,a}^l)$. 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 $\beta_r$ 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 $d_1^l \times d_2^l$; we denote each of these matrices as $L_{:,:,\kappa}$ for $\kappa = 1, \ldots, K$. Here $:$ denote that we select all elements in a given dimension, for example, if $A$ is a matrix, $A_{:,j}$ is used for all the rows and $j$-th column. Now, consider $T$ different linear combinations of such collection of $K$ matrices, expressed as the columns of a $K \times T$ matrix $S$, then

each matrix $W_r^l$ for $r = 1, \ldots, T$, can be expressed as

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

Thus, 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 (2009) and shown in (2.12). Since all the strategies to build $\mathcal{W}$ from other pieces are based on tensor products, the entire process is differentiable and all those pieces can be learned with gradient descent. Another 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. 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 layers 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 $L \times T$ policy matrix $B$ with the $T$ policies that determines which layers are used for each task. The problem is then a bi-level optimization

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

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

## 2.4 Multi-Task Learning with Kernel Methods

Kernel methods are central in ML, as they have been successful in many areas, and their principal appeal is that the original features are implicitly transformed to a kernel space, possibly infinite-dimensional, where the problems of classification or regression 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. Also, operating in a general RKHS $\mathcal{H}$, where there exists a kernel function such that

$$k(x, \hat{x}) = \langle \phi(x), \phi(\hat{x}) \rangle \text{ , for every } \phi \in \mathcal{H},$$

requires to be more careful than when using linear models, specially when different kinds of regularizers are used to enforce a coupling between tasks, as shown in Section 2.2. These characteristics make it more difficult to design MTL algorithms with kernel methods. Instead, other proposals have been made; in this subsection we review the most relevant ones, such as learning vector-valued functions, graph Laplacian regularization or joint learning of common and task-specific parts, among others.

The task relation learning, a parameter-based approach, is a common one in MTL with kernel methods, with a special focus on GPs, as we have seen in Section 2.2. Using a GP formulation, the coupling between tasks is enforced using the covariance matrix, usually defining a prior as the one in (2.17), with an inter-task covariance matrix and a covariance matrix among patterns; then, different strategies to learn the task-covariance matrix are proposed as in Bonilla et al. (2007); Lawrence and Platt (2004).

A feature-based strategy was also presented in Argyriou et al. (2008), where each task parameter $w_r$ is expressed as $w_r = Ua_r$, that is, $U$ contains the elements that will be combined linearly, according to the weights $a_r$, to form the parameters $w_r$. In the linear case, the optimization problem is given in (2.9), and the kernel extension is the following one:

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

where $\phi$ 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

$$\underset{W\in\mathbb{R}^{d\times T}}{\arg\min} \sum_{r=1}^{T}\sum_{i=1}^{m_r} \ell(y_i^r, \langle w_r, \phi(x_i^r)\rangle) + \lambda \|W\|_*^2, \tag{2.25}$$

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 (2.25) using this result, where they use an alternating optimization procedure. This method, however, is computationally challenging, because it requires finding an orthonormal basis of the $n \times n$ kernel matrix, with $n = \sum_{r=1}^{T} m_r$; then, at each iteration of the alternating minimization, a full optimization problem has to be solved. That is, if we use an SVM as our base model for classification, by selecting the hinge loss as the loss function, at each iteration we have a cost $O(n^2)$. A clusterized extension, with similar computational limitations, is proposed in Kang et al. (2011), whose corresponding optimization problem we have given in (2.20).

Other 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

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

where $W$ is a matrix with $T$ columns, one for each task, and $\boldsymbol{b}$ is the vector of corresponding biases. Finally, $\Omega$ 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. A commonly

used loss function $\ell$ for this approach in regression problems is

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

In Micchelli and Pontil (2004, 2005), the authors 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 the targets or labels are not necessarily a vector; that is, for each data $x_i^r$ there is a single scalar $y_i^r$. Then, the MT regularized risk using this formulation has to be expressed as

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

where $A$ is a binary matrix with the same sparsity pattern that $Y$ has, and $\odot$ is the element-wise multiplication. By using this formulation, we are not being as efficient as we could, since we consider a vector-valued target for each pattern, where we really have a scalar one.

Other strategy that seems better suited for kernel methods is a combination-based one, first presented in Evgeniou and Pontil (2004), as can be seen in (2.22) for the linear case, and then extended in Cai and Cherkassky (2009, 2012). The kernelized optimization problem is

$$\arg\min_{w,V} C \sum_{r=1}^{T} \sum_{i=1}^{m_r} \ell(y_i^r, \langle w, \phi(x_i^r)\rangle + \langle v_r, \phi_r(x_i^r)\rangle) + \mu \|w\|^2 + \operatorname{tr}(V^\intercal V).$$

With this formulation, it is possible to use different implicit transformations for the common part $\phi$ and for each specific part $\phi_r$. This can be done because the coupling between tasks is not made using restrictions with some regularizer, but using a common model for all tasks. Similary to the representer theorem, it can be shown that

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

where the dual coefficients $\alpha_i^r$ 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. This approach has a strong motivation given its connection with the Learning Using Privileged Information (LUPI) paradigm, which we later describe in Section 2.5.

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. Consider $\boldsymbol{w} = \operatorname{vec}(W)$ for the vectorized matrix $W$, i.e. the vector $(W_1^\intercal, \ldots, W_T^\intercal)^\intercal$ where $W_1, \ldots, W_T$ are the columns of $W$; then, given a linear problem

$$\sum_{r=1}^{T} \sum_{i=1}^{m} \ell(y_i^r, \langle w_r, x_i^r\rangle) + \mu \boldsymbol{w}^\intercal (E \otimes I) \boldsymbol{w},$$

where $E$ is a $T \times T$ matrix and $\otimes$ is the Kronecker product between matrices. The solution to this problem can be expressed as

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

with $B_r$ being the columns of a matrix $B$ such that $E = (B^\intercal B)^{-1}$. Then, Evgeniou *et al.* show this to be equivalent to using a kernel

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

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

$$\sum_{r,s=1}^{T} (\Theta)_{rs} \langle w_r, w_s \rangle = \boldsymbol{w}^\intercal (\Theta \otimes I) \boldsymbol{w},$$

where $\Theta$ is the inter-task relationship matrix, and they propose different strategies to learn such matrix.

## 2.5 Learning Using Privileged Information and Multi-Task Learning

Another important motivation for MTL can be found in the Learning Using 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 $\hat{R}_D$ given a set of training samples. Vapnik is one of the main contributors to the theory of statistical learning, see Vapnik (2000). Within this theory several important results are provided: necessary and sufficient conditions for the consistency of learning processes and bounds for the rate of convergence. Some of these results, which use the notion of VC-dimension, have been presented in Section 1.3. A new inductive principle, Structural Risk Minimization (SRM), and an algorithm, Support Vector Machine (SVM), that make use of this notion to improve the learning process are given, as explained in Chapter 1.

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 state 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.

### 2.5.1 Privileged Information and Convergence Rates

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 LUPI.

In this paradigm, given a set of i.i.d. triplets

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

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

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

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 $x^\diamond$ of a space $\mathcal{X}^\diamond$, which is different from $\mathcal{X}$. The goal of the Teacher is, given a pair $(x_i, y_i)$, to provide an information $x_i^* \in \mathcal{X}^\diamond$ given some probability $P\left(x^\diamond \mid x\right)$. That is, the "intelligence" of the Teacher is defined by the choice of the space $\mathcal{X}^\diamond$ and the conditional probability $P\left(x^\diamond \mid x\right)$. To understand better this paradigm consider the following example given in Vapnik and Izmailov (2015). 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 aggressive proliferation of cells of Type B". These commentaries are the elements $x^\diamond$ of a certain space $\mathcal{X}^\diamond$ and the Teacher also chooses the probability $P\left(x^\diamond \mid x\right)$, which defines when to express this additional information.

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 $\hat{R}_D$ and the VC-dimension of the set of hypotheses $\mathcal{H}$. In the case of classification, where $\mathcal{Y} = \{-1, 1\}$, and the loss $\ell(h\left(x, \alpha\right), y) = \mathbf{1}_{yh(x,\alpha)\leq 0}$, the risks can be expressed as

$$R_P(\alpha) = \int \mathbf{1}_{yh(x,\alpha)\leq 0} dP(x, y),$$
$$\hat{R}_{D_n}(\alpha) = \frac{1}{n}\sum_{i=1}^{n} \mathbf{1}_{y_i h(x_i,\alpha)\leq 0}.$$

Therefore, in the case of the classification, with the loss $\ell(h\left(x, \alpha\right), y) = \mathbf{1}_{yh(x,\alpha)\leq 0}$, we can interpret the expected risk $R_P(\alpha)$ as the probability of error, and the empirical risk $\hat{R}_D(\alpha)$ as the frequency of errors in the training sample. Consider now the rule $h(x, \alpha_n)$ that minimizes the frequency of errors, that is, $\alpha_n = \arg\min_{\alpha\in A} \hat{R}_{D_n}(\alpha)$; then, in Vapnik (1982, Theorem 6.8), the following bound for the rate of convergence is given with probability $1 - \eta$:

$$R_P(\alpha_n) \leq \hat{R}_{D_n}(\alpha_n) + \frac{d\log\left(\frac{2n}{d}\right) - \log\left(\frac{\eta}{12}\right)}{n}\left(1 + \sqrt{\frac{\hat{R}_{D_n}(\alpha_n) n}{d\left(\log\left(\frac{2n}{d}\right) + 1\right) - \log\left(\frac{\eta}{12}\right)}}\right). \tag{2.26}$$

That is, the bound is controlled by the ratio $d/n$, where $d$ is the $\mathrm{VCdim}\left(\mathcal{H}\right)$. 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 $\hat{R}_{D_n}(\alpha_n) = 0$. In this case, the following bound for the rate of converge holds

$$R_P(\alpha_n) \leq O\left(\frac{d \log\left(\frac{2n}{d}\right) - \log \eta}{n}\right).$$

- **Non-Separable case:** the training data cannot be classified in two groups without errors. That is, for all $\alpha \in A$, there exists $i \in \{1, \ldots, n\}$, such that $y_i h(x_i, \alpha) \leq 0$, and thus $\hat{R}_{D_n}(\alpha_n)$. Observe in (2.26) that in this case we have one term that depends essentially $d/n$ and other on $\sqrt{d/n}$. Given a finite $d$, as $n$ grows, the term depending on $\sqrt{d/n}$ will go to zero at a lower rate than that depending on $d/n$; thus, the following bound for the rate of converge holds

$$R_P(\alpha_n) \leq \hat{R}_{D_n}(\alpha_n) + O\left(\sqrt{\frac{d \log\left(\frac{2n}{d}\right) - \log \eta}{n}}\right). \tag{2.27}$$

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

### 2.5.2 From Oracle SVM to SVM+

Vapnik *et al.* illustrate the difference in convergence rates between the separable and non-separable cases by looking at the SVM. Recall that 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

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

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 $\xi_1, \ldots, \xi_n$) have to be estimated with $n$ examples.

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

$$(x_1, \xi_1^*, y_1), \ldots, (x_n, \xi_n^*, y_n)$$

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

$$\xi_i^* = \max\left(0, 1 - h\left(x, \alpha^*\right)\right), \ \forall i = 1, \ldots, n,$$

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

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

subject to the constraints

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

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

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

That is, when the privileged information encoded in $\xi^*$ is available, we can recover the rate with an order of magnitude $d/n$.

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 $\mathcal{X}^\diamond$ and a set of functions $\{f^\diamond(x^\diamond, \alpha^\diamond), \alpha^\diamond \in A^\diamond\}$, then models the slack variables as

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

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

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

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

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

subject to the constraints

$$y_i h\left(x_i, \alpha\right) \geq -f^\diamond(x_i^\diamond, \alpha^\diamond).$$

Here we are not considering a margin in the contraints, that is, the margin is 0. Now, let $f(x^\diamond, \alpha_n^\diamond)$, $h(x, \alpha_n)$ be the solutions that minimize this empirical risk; then, in Vapnik and Vashist (2009, Proposition 2) the following results for the convergence bound are given:

$$R_P(n) \leq P(f^\diamond(x^\diamond, \alpha_n^\diamond) \geq 0) + O\left(\frac{(d + d^\diamond) \log\left(\frac{2n}{(d+d^\diamond)}\right) - \log \eta}{n}\right), \tag{2.28}$$

where $d^\diamond$ is the VC-dimension of the space of hypothesis $\{f(x^\diamond, \alpha^\diamond) \in A^\diamond\}$. This result shows that, to maintain the best convergence rate $d/n$, we need to estimate the rate of convergence of $P(f^\diamond(x^\diamond, \alpha_n^\diamond) \geq 0)$, where we can use the standard rate for non-separable problems, i.e. $\sqrt{d^\diamond/n}$.

We can now observe the differences between the standard bound for non-separable problems, given in (2.27), and the bound (2.28) obtained with the LUPI paradigm; looking at the second terms we see that we have replaced the order of magnitude $\sqrt{d/n}$ by $d + d^\diamond/n$, which was our goal. However, the price that we have paid for this is that we have also replaced the empirical risk $\hat{R}_{D_n}(\alpha_n)$, which is a known quantity, by a probability $P(f^\diamond(x^\diamond, \alpha_n^\diamond))$, which converges at a rate $\sqrt{d^\diamond/n}$. Nevertheless, observe that $\mathcal{X}^\diamond$ is the space suggested by the Teacher, which hopefully has a lower capacity, and thus, since $d^\diamond$ is smaller, the convergence will be faster in this space.

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

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

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

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

The minimization problem is the following:

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

Here $\phi$ and $\phi^\diamond$ are two transformations that can be different. Next, to replace the positive part in the con $[\cdot]_+$ in the constraints, Vapnik *et al.* propose a relaxation of this problem where the idea is to model the slack variables $\xi$ as

$$\xi(x^\diamond, y) = y(w^\diamond \phi^\diamond(x^\diamond) + b^\diamond) + \zeta(x^\diamond, y),$$

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

$$\begin{aligned} \underset{w,w^\diamond,b,b^\diamond,\zeta_i}{\arg\min} \quad & C \sum_{i=1}^{n} ([y_i(\langle w^\diamond, \phi^\diamond(x_i^*)\rangle + b^\diamond)] + \zeta_i) + \hat{C} \sum_{i=1}^{n} \zeta_i \\ & \quad + \frac{1}{2}\langle w, w\rangle + \frac{\mu}{2}\langle w^\diamond, w^\diamond\rangle \\ \text{s.t.} \quad & y_i(\langle w, \phi(x_i)\rangle + b) \geq 1 - [y_i(\langle w^\diamond, \phi^\diamond(x_i^\diamond)\rangle + b^\diamond) + \zeta_i], \\ & y_i(\langle w^\diamond, \phi^\diamond(x_i^\diamond)\rangle + b^\diamond) + \zeta_i \geq 0, \\ & \zeta_i \geq 0, \\ \text{for} \quad & i = 1, \ldots, n, \end{aligned} \tag{2.29}$$

where $C$ and $\hat{C}$ are hyperparameters. Problem (2.29) is convex and, as shown in Vapnik and Izmailov (2015), the corresponding dual problem can be expressed as

$$\begin{aligned}
\underset{\alpha_i,\delta_i}{\arg\min} \quad & \frac{1}{2}\sum_{i,j=1}^{n} y_i y_j \alpha_i \alpha_j k(x_i, x_j) + \frac{1}{2\mu}\sum_{i,j=1}^{n} y_i y_j (\alpha_i - \delta_i)(\alpha_j - \delta_i) k^*(x_i^\diamond, x_j^\diamond) - \sum_{i=1}^{n} \alpha_i \\
\text{s.t.} \quad & 0 \leq \delta_i \leq C \\
& 0 \leq \alpha_i \leq \hat{C} + \delta_i, \\
& \sum_{i=1}^{n} \delta_i y_i = 0, \ \sum_{i=1}^{n} \alpha_i y_i = 0, \\
\text{for} \quad & i = 1, \ldots, n,
\end{aligned} \tag{2.30}$$

where we use the kernel functions

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

We can observe in Problem (2.30) that the LUPI paradigm exerts a similarity control, correcting the similarity in space $\mathcal{X}$ with the similarity in the privileged space $\mathcal{X}^\diamond$. For that reason, $\mathcal{X}$ and $\mathcal{X}^\diamond$ are named Decision Space and Correction Space, respectively.

### 2.5.3 Connection between SVM+ and MTL

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

$$\begin{aligned}
\underset{w,b,v_r,b_r,\xi_i^r}{\arg\min} \quad & C\sum_{r=1}^{T}\sum_{i=1}^{m_r} \xi_i^r + \frac{1}{2}\langle w, w \rangle + \sum_{r=1}^{T} \frac{\mu}{2} \langle v_r, v_r \rangle \\
\text{s.t.} \quad & y_i^r(\langle w, \phi(x_i^r) \rangle + b + \langle v_r, \phi_r(x_i^r) \rangle + b_r) \geq 1 - \xi_i^r, \\
& \xi_i^r \geq 0, \\
\text{for} \quad & r = 1, \ldots, T; \ i = 1, \ldots, m_r.
\end{aligned} \tag{2.31}$$

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 $\phi$ and the task-independent ones $\phi_r$ can be different. The dual problem corresponding to (2.31) is

$$\begin{aligned}
\arg\min_{\alpha_i} \quad & \frac{1}{2}\sum_{r,s=1}^{T}\delta_{rs}\sum_{i,j=1}^{m_r,m_s} y_i^r y_j^s \alpha_i^r \alpha_j^s k(x_i^r, x_j^s) + \frac{1}{2\mu}\sum_{r,s=1}^{T}\sum_{i,j=1}^{m_r,m_s} y_i^r y_j^s \alpha_i^r \alpha_j^r k_r(x_i^r, x_j^r) \\
& - \sum_{r=1}^{T}\sum_{i=1}^{m_r}\alpha_i^r \\
\text{s.t.} \quad & 0 \leq \alpha_i^r \leq C, \\
& \sum_{i=1}^{m_r}\alpha_i^r y_i^r = 0, \\
\text{for} \quad & r = 1, \ldots, T;\ i = 1, \ldots, m_r,
\end{aligned} \tag{2.32}$$

where $\delta_{rs} = 1$ only if $r = s$ and $\delta_{rs} = 0$ otherwise. Later, in Chapter 3 we will describe with more detail this MTL SVM, and we will explain our proposed modification of it.

The connection between the MTL SVM and the SVM+ is not trivial, but we can observe that there exist similarities, some of which are pointed out in Liang and Cherkassky (2008). Problem (2.31) can be regarded as an adaptation of (2.29) to solve MTL problems, where different tasks are incorporated and multiple correcting spaces are defined using the transformations $\phi_r$. If we consider the problem (2.31) with a single task, it is a modification of the SVM+ problem (2.29) where the slack variables are modeled as

$$\xi(x, y) = y(v_r\phi_r(x) + b_r).$$

That is, it is a relaxation of the original problem (2.29), 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 $\zeta_i$ are no longer required, this is reflected in a simpler dual form (2.32), where only $n$ dual variables have to be estimated, instead of the $2n$ dual variables of (2.30). The MT part in (2.32) resides in the $\delta_{rs}$ 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 MTL SVM and SVM+. The results for the improved rate of convergence with an Intelligent Teacher may not be valid for MTL SVM, since we are not modelling the slack variables $\xi$ adequately. It is still a work in progress to study the rate of convergence of MTL SVM and to establish clearer links with SVM+.

## 2.6 Conclusions

In this Chapter we have introduced the concept of MTL, we try to motivate it, present its advantages and describe multiple ways of implementing it. In Section 2.1 we have presented some of the foundational works on the theory of MTL. First, we describe the work of (Baxter, 2000), which sets the foundation for MTL and LTL theory, establishing necessary conditions for the consistency of MTL. Following the work of Vapnik *et al.*, reviewed in Chapter 1, this is done by bounding the expected MTL risk in terms of the empirical risk and using concepts related with the capacity of function spaces. Next, we present the work of Ben-David and Schuller (2003), which gives a definition for related tasks. Combining this task relatedness concept with the results from Baxter (2000), the

authors derive new bounds for the expected MTL risk. The work of Ben-David *et al.* is particularly useful when there is a target task and the rest are auxiliary ones.

After the description of the theoretical works, in Section 2.2 we have reviewed multiple approaches for MTL and categorize them in three main groups: feature-based, parameter-based and combination-based methods. The feature-based methods are more common in linear models or NNs, the parameter-based ones typically use linear models as well, and the combination-based, which is the less common one, can be used also with kernel methods.

To complete this review, we consider a particular analysis of MTL with NNs and kernel methods, which are two of the most successful models in the two last decades. In Section 2.3, a survey of the MTL methods based on NNs has been given, where it can be observed that they are mainly focused on finding a shared representation beneficial to all tasks. In Chapter 3 we will present an alternative combination-based MTL method with NNs. Also, an overview of MTL with kernel methods is given in Section 2.4, where there are fewer works than for NNs or linear models, especially for SVMs. Other important theoretical motivation for MTL that we have presented is its connection with the LUPI paradigm, presented in Section 2.5. A kernel method that is central in this work is the MTL SVM, which combines a common and task-specific parts, and we also present in Section 2.5 its connection with the LUPI paradigm.

In summary, in this chapter, we have covered some of the most relevant works in MTL. We have observed that there are some theoretical reasons that support the MTL, and we have reviewed multiple approaches that apply it. We remark two facts from this survey. First, although there are many MTL strategies, few of them are applicable for kernel methods. Also, most approaches using NNs are based on a feature-based strategy, but we have seen that there are other two main groups: parameter-based and combination-based. In the next chapters we try to cover these gaps. In Chapters 3 and 4 we describe novel approaches for MTL using kernel methods, particularly for the L1, L2, and LS-SVM. In Chapter 4 we present a GL approach, where different task-specific parts, possibly in an infinite-dimensional kernel space, are coupled together using a specific graph regularizer. Also, in Chapter 3 we present a convex formulation for combination-based MTL, which is not only valid for kernel methods, but also for NNs.

Chapter 3

# A Convex Formulation for Multi-Task Learning

As we have seen in Chapter 2, the Multi-Task Learning (MTL) proposals can be categorized as feature-based, parameter-based and combination-based approaches. Feature-based proposals follow 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 task-specific ones. In this chapter we will present a convex formulation for combination-based MTL. With this formulation, a general task-specialized model is defined as

$$h_r(\cdot) = \lambda_r g(\cdot) + (1 - \lambda_r) g_r(\cdot). \tag{3.1}$$

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-Task Learning (CTL) approach, while $\lambda_r = 0$ leads to an Independent-Task Learning (ITL) one. All the values $\lambda_r \in (0, 1)$ correspond to pure MTL models, being closer to a common model when $\lambda_r$ is close to 1 and more specific when $\lambda_r$ is close to 0. Given an MTL sample

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

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

$$\hat{R}_D(g, g_1, \ldots, g_T) = C \sum_{r=1}^{T} \sum_{i=1}^{m_r} \ell(\lambda_r g(x_i^r) + (1 - \lambda_r) g_r(x_i^r), y_i^r) + \Omega(g) + \sum_{r=1}^{T} \Omega_r(g_r),$$

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 $\hat{R}_D$ is optimized jointly in all parts. In this chapter these characteristics will be presented in specific models using this convex MTL formulation. In particular, we will propose

convex MTL formulations for kernel methods, particularly the L1, L2 or LS-Support Vector Machine (SVM), and also for Neural Networks (NNs).

Moreover, we also consider a natural alternative to convex MTL, which 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

$$\hat{R}_D(\lambda_1, \ldots, \lambda_T) = C \sum_{r=1}^{T} \sum_{i=1}^{m_r} \ell(\lambda_r g^*(x_i^r) + (1 - \lambda_r) g_r^*(x_i^r), y_i^r).$$

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

The chapter is structured as follows. In Section 3.1 we present the convex MTL formulation for kernel methods, and in Section 3.2 we do it for NNs. Finally, in Section 3.3 we present the alternative method of combining pre-trained models in a convex manner, proposing methods to select the optimal combination parameter for different loss functions.

## 3.1 Convex Multi-Task Learning with Kernel Methods

As explained in the previous chapters, kernel methods 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, such as the L1, L2 or LS-SVM, in the case of STL, a regularized risk problem like the following one is solved,

$$\hat{R}_D(w) = \sum_{i=1}^{n} \ell(\langle w, \phi(x_i) \rangle + b, y_i) + \mu \Omega(w), \tag{3.2}$$

where $D$ 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 (3.2) $\phi$ 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, $\phi(\cdot)$ is an element of an RKHS, where $k$ is the reproducing kernel. The Representer Theorem (Schölkopf et al., 2001), which we give in Theorem 1.14, states that the optimal solution of problem (3.2) 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 (SRM) 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.

MTL with kernel methods requires 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 methods, since the (implicit) transformation functions $\phi$ used are not learned but fixed, and determined by the choice of the kernel function. Therefore, other strategies have to be developed. One of the first approaches to MTL with kernel methods was presented in Evgeniou and Pontil (2004), and later extended in Cai and Cherkassky (2009, 2012), where they use an L1-SVM formulation and the model for each task is 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 then

$$\hat{R}_D(w, v_1, \ldots, v_T) = C \sum_{r=1}^{T} \sum_{i=1}^{m_r} \ell(\langle w, \phi(x_i^r)\rangle + \langle v_r, \phi_r(x_i^r)\rangle + b_r, y_i^r) + \mu \|w\|^2 + \sum_{r=1}^{T} \|v_r\|^2 ,$$

where $\mu$ is the hyperparameter to control the regularization of the common and specific parts, respectively. Here, observe also that different transformations are used: the transformation $\phi(\cdot)$ corresponds to the common part of the model, while $\phi_r(\cdot)$ is task-specific. This is a joint learning approach that is developed for the L1-SVM, to which the authors give the name of *Regularized MTL*, but we will refer to it as the additive MTL approach. The influence of the hyperparameters $C$ and $\mu$ is not straightforward, and the behaviour of the solutions is asymptotical. For example, when $\mu \to \infty$, the common part $w$ disappears, and we end up with independent models for each task. Also, with $C$ large enough and $\mu \to 0$, the result tendency is the contrary as it approaches a common model for all tasks.

To obtain a more interpretable formulation, here we propose to use our convex MTL framework of (3.1) and define the common part as $g(\cdot) = \langle w, \phi(\cdot)\rangle + b$ and the task-specific parts as $g_r(\cdot) = \langle v_r, \phi_r(\cdot)\rangle + d_r$, so the MTL models are

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

where $\lambda_r \in [0, 1]$ are hyperparameters, and the corresponding regularized risk is

$$\begin{aligned}\hat{R}_D(w, v_1, \ldots, v_T) =& C \sum_{r=1}^{T} \sum_{i=1}^{m_r} \ell(\lambda_r \left\{\langle w, \phi(x_i^r)\rangle + b\right\} + (1 - \lambda_r) \left\{\langle v_r, \phi_r(x_i^r)\rangle + d_r\right\}, y_i^r) \\ &+ \|w\|^2 + \sum_{r=1}^{T} \|v_r\|^2 .\end{aligned}$$

We will name this approach convex, in contrast to the additive approach of the original formulation. Here, the interpretation is simpler. The model with $\lambda_r = 1$ for $r = 1, \ldots, T$, is equivalent to learning a single common model for all tasks, that is $w_r = w$, while with $\lambda_r = 0$, the models for each task are completely independent, i.e. $w_r = v_r$.

Moreover, we show that the two formulations, additive and convex, are equivalent within an L1-SVM setting. Finally, we also present Ruiz et al. (2021b) the extensions for the L2 and LS-SVM, where we apply the convex MTL formulation.

### 3.1.1 Additive Multi-Task Learning SVM

The additive MTL primal problem formulation, presented in Evgeniou and Pontil (2004) and extended for multiple kernel functions and task-specific biases in Cai and Cherkassky (2012), is the following one,

$$\begin{aligned} \underset{w,\boldsymbol{v},\boldsymbol{b},\xi}{\arg\min} \quad & J(w,\boldsymbol{v},\boldsymbol{b},\boldsymbol{\xi}) = C\sum_{r=1}^{T}\sum_{i=1}^{m_r}\xi_i^r + \frac{1}{2}\sum_{r=1}^{T}\|v_r\|^2 + \frac{\mu}{2}\|w\|^2 \\ \text{s.t.} \quad & y_i^r(\langle w,\phi(x_i^r)\rangle + \langle v_r,\phi_r(x_i^r)\rangle + b_r) \geq p_i^r - \xi_i^r, \\ & \xi_i^r \geq 0;\ i=1,\ldots,m_r,\ r=1,\ldots,T. \end{aligned} \tag{3.3}$$

Here, we are using the unified formulation for regression and classification where we use the sample

$$D = \{(x_i^r, y_i^r, p_i^r), i=1,\ldots,m_r,\ r=1,\ldots,T\}.$$

Recall that, as explained in Section 1.5, the problem (3.3) is equivalent to a classification problem when we select $p_i^r = 1$ and $y_i^r \in \{-1,1\}$ as the corresponding labels for all $i=1,\ldots,m_r$, $r=1,\ldots,T$. Also, the problem (3.3) is equivalent to a regression problem if for each task $r=1,\ldots,T$, we duplicate the number of instances and select $y_i^r = 1, p_i^r = -t_i^r - \epsilon$ for $i=1,\ldots,m_r$ and $y_i^r = -1, p_i^r = -t_{i-m_r}^r - \epsilon$ for $i=m_r+1,\ldots,2m_r$, where $t_i^r$, $i=1,\ldots,m_r$ are the regression target values. 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 $\phi$ is used for the common part, while the transformation $\phi_r$ is task-specific.

In (3.3) there are hyperparameters: $C$ and $\mu$, which, in combination, balance the different parts of the objective function. The hyperparameter $C$ plays the same role that in the standard L1-SVM: it balances the trade-off between the loss incurred by the model, represented by the slack variables $\xi_i^r$, and its complexity, 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 $C$ indirectly 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 $\mu$, in combination with $C$, balances how common or specific the models are. Large values of $\mu$ penalize the common part, resulting in more specific models; while small values of $\mu$, alongside small values of $C$, result in an objective function where the specific norms are the terms more strongly penalized, so the task-specific parts vanish and common models are promoted. Hence, we can distinguish the following cases:

- Reduction to an ITL approach:

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

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

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

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

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

- Pure MTL approach:

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

  The models combine a common and task-specific parts.

Observe that (3.3) 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 compute the Lagrangian of problem (3.3),

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

where $\alpha_i^r, \beta_i^r \geq 0$ are the Lagrange multipliers. Here we use the vectors $\boldsymbol{\alpha}$ and $\boldsymbol{\beta}$, defined as

$$\boldsymbol{\alpha} = \left(\alpha_1^1, \ldots, \alpha_{m_1}^1, \ldots, \alpha_1^T, \ldots, \alpha_{m_T}^T\right)^\intercal$$

and analogously for $\boldsymbol{\beta}$. For compactness, we also use

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

Recall that the KKT conditions for convex differentiable problems, like this one, are those corresponding to the saddle point in the primal variables, namely

$$\begin{aligned}
\nabla w \mathcal{L}(w, \boldsymbol{v}, \boldsymbol{b}, \boldsymbol{\xi}, \boldsymbol{\alpha}, \boldsymbol{\beta}) &= \mathbf{0}, \\
\nabla \boldsymbol{v} \mathcal{L}(w, \boldsymbol{v}, \boldsymbol{b}, \boldsymbol{\xi}, \boldsymbol{\alpha}, \boldsymbol{\beta}) &= \mathbf{0}, \\
\nabla \boldsymbol{b} \mathcal{L}(w, \boldsymbol{v}, \boldsymbol{b}, \boldsymbol{\xi}, \boldsymbol{\alpha}, \boldsymbol{\beta}) &= \mathbf{0}, \\
\nabla \boldsymbol{\xi} \mathcal{L}(w, \boldsymbol{v}, \boldsymbol{b}, \boldsymbol{\xi}, \boldsymbol{\alpha}, \boldsymbol{\beta}) &= \mathbf{0},
\end{aligned}$$

the corresponding to the saddle point in the dual variables

$$\begin{aligned}
\nabla \boldsymbol{\alpha} \mathcal{L}(w, \boldsymbol{v}, \boldsymbol{b}, \boldsymbol{\xi}, \boldsymbol{\alpha}, \boldsymbol{\beta}) &= \mathbf{0}, \\
\nabla \boldsymbol{\beta} \mathcal{L}(w, \boldsymbol{v}, \boldsymbol{b}, \boldsymbol{\xi}, \boldsymbol{\alpha}, \boldsymbol{\beta}) &= \mathbf{0},
\end{aligned}$$

and the vanishing dual-gap conditions, which, in our case are

$$\alpha_i^r \left[ y_i^r (\langle w, \phi(x_i^r)\rangle + \langle v_r, \phi_r(x_i^r)\rangle + b_r) - p_i^r + \xi_i^r \right] = 0. \tag{3.4}$$

For this problem, the dual function is

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

Here, we will use the strong duality theorem, which we have given in Theorem 1.34. We can apply it since we have a convex differentiable objective function and affine constraints. Thus, by maximizing $\Theta(\boldsymbol{\alpha},\boldsymbol{\beta})$ we can get an optimal solution. To define the dual problem we need the optimal primal variables, which we obtain by taking derivatives and making them 0 as:

$$\begin{aligned}\nabla_w \mathcal{L} = \mathbf{0} &\implies \mu w - \sum_{r=1}^{T}\sum_{i=1}^{m_r} \alpha_i^r \left\{y_i^r \phi(x_i^r)\right\} = \mathbf{0}, \\ \nabla_{v_r} \mathcal{L} = \mathbf{0} &\implies v_r - \sum_{i=1}^{m_r} \alpha_i^r \left\{y_i^r \phi_r(x_i^r)\right\} = \mathbf{0}, \\ \partial_{b_r} \mathcal{L} = 0 &\implies \sum_{i=1}^{m_r} \alpha_i^r y_i^r = 0, \qquad (3.5) \\ \partial_{\xi_i^r} \mathcal{L} = 0 &\implies C - \alpha_i^r - \beta_i^r = 0. \qquad (3.6)\end{aligned}$$

Using these results and substituting back in the Lagrangian and rearranging the terms we obtain

$$\begin{aligned}&\mathcal{L}(w^*,\boldsymbol{v}^*,\boldsymbol{b}^*,\boldsymbol{\xi}^*,\boldsymbol{\alpha},\boldsymbol{\beta}) \\ &= \frac{1}{2}\sum_{r=1}^{T}\left\|\sum_{i=1}^{m_r} \alpha_i^r \left\{y_i^r \phi_r(x_i^r)\right\}\right\|^2 + \frac{\mu}{2}\left\|\frac{1}{\mu}\sum_{r=1}^{T}\sum_{i=1}^{m_r} \alpha_i^r \left\{y_i^r \phi(x_i^r)\right\}\right\|^2 \\ &\quad - \sum_{r=1}^{T}\sum_{i=1}^{m_r} \alpha_i^r \left\{ y_i^r \left[ \left\langle \frac{1}{\mu}\sum_{s=1}^{T}\sum_{j=1}^{m_s} \alpha_j^s \left\{y_j^s \phi(x_j^s)\right\}, \phi(x_i^r) \right\rangle \right] \right\} \\ &\quad - \sum_{r=1}^{T}\sum_{i=1}^{m_r} \alpha_i^r \left\{ y_i^r \left[ \left\langle \sum_{j=1}^{m_r} \alpha_j^r \left\{y_j^r \phi_r(x_j^r)\right\}, \phi_r(x_i^r) \right\rangle \right] \right\} \\ &\quad - \sum_{r=1}^{T}\sum_{i=1}^{m_r} \alpha_i^r \left\{-p_i^r\right\},\end{aligned}$$

which is equivalent to

$$\begin{aligned}&\mathcal{L}(w^*,\boldsymbol{v}^*,\boldsymbol{b}^*,\boldsymbol{\xi}^*,\boldsymbol{\alpha},\boldsymbol{\beta}) \\ &= -\frac{1}{2\mu}\left\langle \sum_{r=1}^{T}\sum_{i=1}^{m_r} \alpha_i^r \left\{y_i^r \phi(x_i^r)\right\}, \sum_{r=1}^{T}\sum_{i=1}^{m_r} \alpha_i^r \left\{y_i^r \phi(x_i^r)\right\} \right\rangle \\ &\quad - \frac{1}{2}\sum_{r=1}^{T}\left\langle \sum_{i=1}^{m_r} \alpha_i^r \left\{y_i^r \phi_r(x_i^r)\right\}, \sum_{i=1}^{m_r} \alpha_i^r \left\{y_i^r \phi_r(x_i^r)\right\} \right\rangle \\ &\quad + \sum_{r=1}^{T}\sum_{i=1}^{m_r} \alpha_i^r p_i^r.\end{aligned}$$

Observe that $\boldsymbol{\beta}$ has disappeared from the Lagrangian, however, it plays a role in the feasibility of $\boldsymbol{\alpha}$; combining the condition (3.6) with $\alpha_i^r, \beta_i^r \geq 0$, we get that $0 \leq \alpha_i^r \leq C$. Then, with this feasibility condition, the dual problem is

$$\max_{0 \leq \boldsymbol{\alpha} \leq C} \mathcal{L}(w^*, \boldsymbol{v}^*, \boldsymbol{b}^*, \boldsymbol{\xi}^*, \boldsymbol{\alpha}),$$

but we will consider instead the equivalent formulation $\min_{0 \leq \boldsymbol{\alpha} \leq C} \Theta(\boldsymbol{\alpha})$ where

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

Therefore, the dual problem is the following one:

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

Here there are $T$ task-specific equality constraints that have their origin in the derivatives with respect to the specific biases $b_r$ given in (3.5). Using the kernel trick, we can write the dual problem with a matrix formulation:

$$\begin{aligned}
\min_{\boldsymbol{\alpha}} \quad & \Theta(\boldsymbol{\alpha}) = \frac{1}{2} \boldsymbol{\alpha}^\intercal \left( \frac{1}{\mu} Q + K \right) \boldsymbol{\alpha} - \boldsymbol{p}\boldsymbol{\alpha} \\
\text{s.t.} \quad & 0 \leq \alpha_i^r \leq C; \ i = 1, \ldots, m_r, \ \ r = 1, \ldots, T, \\
& \sum_{i=1}^{m_r} \alpha_i^r y_i^r = 0; \ r = 1, \ldots, T.
\end{aligned} \tag{3.7}$$

In this dual problem, $Q$ and $K$ are the common and specific kernel matrices, respectively. The matrix $Q$ is generated using the common kernel, defined as

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

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

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

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

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

so the corresponding matrix is

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

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

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

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 (3.7) is very similar to the CTL one 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 (3.5), are consequence of the specific biases used in the primal problem (3.3). In Cai and Cherkassky (2012) the authors develop a Generalized SMO algorithm to account for these multiple equality constraints. To get the optimal bias $b_r$, we use (3.4) for the support vectors, that is where $\alpha_i^r \neq 0$ and, hence, $y_i^r(\langle w, \phi(x_i^r) \rangle + \langle v_r, \phi_r(x_i^r) \rangle + b_r) - p_i^r + \xi_i^r = 0$.

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 specific of this MTL formulation, which is $\mu$, scales the common matrix $Q$. As with the primal formulation, we can define three different cases:

- Reduction to an ITL approach:

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

  Thus, the kernel matrix is $\hat{Q} \approx K$, a block-diagonal one, 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):

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

  Here, the kernel matrix is $\hat{Q} \approx Q$; this is the standard matrix for CTL.

- Pure MTL approach:

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

  Here the kernel matrix combines the common and specific matrices, that is $\hat{Q} = (1/\mu)Q + K$, where neither of the two terms is negligible.

### 3.1.2 Convex Multi-Task Learning SVM

We present a convex formulation for the MTL L1-SVM, which we have proposed in Ruiz et al. (2019). This formulation offers a better interpretability because the influence of the common and task-specific parts is more clear. Here, instead of the biases $b_r$ that we have used in the additive formulation, we will consider the following convex combination of biases:

$$\lambda b + (1 - \lambda) d_r.$$

Then, the primal problem for the convex MTL L1-SVM is

$$\begin{aligned} \underset{w, \boldsymbol{v}, b, \boldsymbol{d}, \boldsymbol{\xi}}{\arg\min} \quad & J(w, \boldsymbol{v}, b, \boldsymbol{d}, \boldsymbol{\xi}) = C \sum_{r=1}^{T} \sum_{i=1}^{m_r} \xi_i^r + \frac{1}{2} \sum_{r=1}^{T} \|v_r\|^2 + \frac{1}{2} \|w\|^2 \\ \text{s.t.} \quad & y_i^r \left(\lambda_r \left\{\langle w, \phi(x_i^r)\rangle + b\right\} + (1 - \lambda_r) \left\{\langle v_r, \phi_r(x_i^r)\rangle + d_r\right\}\right) \geq p_i^r - \xi_i^r, \\ & \xi_i^r \geq 0, \ i = 1, \ldots, m_r, \ r = 1, \ldots, T. \end{aligned} \tag{3.8}$$

Here, the former hyperparameter $\mu$ used in the regularization is replaced by the hyperparameters $\lambda_r \in [0, 1]$, 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

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

for classification.

With this convex formulation, the roles of hyperparameters $C$ and $\lambda_r$ are independent. Hyperparameter $C$ regulates the trade-off between the loss and the margin size of each

task-specialized model $h_r$, while $\lambda_r$ 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 (3.8), depending on the value of hyperparameters $C$, $\lambda_1, \ldots, \lambda_T$, we can highlight the following situations:

- Reduction to an ITL approach:

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

- Reduction to a CTL approach:

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

- Pure MTL approach:

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

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 $\lambda_r$ calibrate the specifity of the models. The Lagrangian of problem (3.8) is

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

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

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

Again, we use the strong duality theorem, given in Theorem 1.34, to get the solution. 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

$$\begin{aligned}
\nabla_w \mathcal{L} = \mathbf{0} &\implies w^* - \sum_{r=1}^{T} \lambda_r \sum_{i=1}^{m_r} \alpha_i^r \left\{ y_i^r \phi(x_i^r) \right\} = \mathbf{0}, \\
\nabla_{v_r} \mathcal{L} = \mathbf{0} &\implies {v_r}^* - (1-\lambda_r) \sum_{i=1}^{m_r} \alpha_i^r \left\{ y_i^r \phi_r(x_i^r) \right\} = \mathbf{0}, \\
\partial_b \mathcal{L} = 0 &\implies \sum_{r=1}^{T} \sum_{i=1}^{m_r} \alpha_i^r y_i^r = 0, \\
\partial_{d_r} \mathcal{L} = 0 &\implies \sum_{i=1}^{m_r} \alpha_i^r y_i^r = 0, \\
\partial_{\xi_i^r} \mathcal{L} = 0 &\implies C - \alpha_i^r - \beta_i^r = 0.
\end{aligned}$$

Using these results and substituting back in the Lagrangian we obtain:

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

As with the additive formulation, the $\beta_i^r$ multipliers disappear, and the dual problem is $\min_{\boldsymbol{\alpha}} \Theta(\boldsymbol{\alpha})$ with

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

where $\boldsymbol{\alpha}$ is feasible. Therefore, the dual problem corresponding to the convex MTL problem (3.8) is the following one:

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

And the matrix formulation using the kernel matrices is

$$\begin{aligned} \min_{\boldsymbol{\alpha}} \quad & \Theta(\boldsymbol{\alpha}) = \frac{1}{2}\boldsymbol{\alpha}^{\intercal}\left(\Lambda Q \Lambda + (I_n - \Lambda)\, K\, (I_n - \Lambda)\right)\boldsymbol{\alpha} - \boldsymbol{p}\boldsymbol{\alpha} \\ \text{s.t.} \quad & 0 \leq \alpha_i^r \leq C;\ i = 1, \ldots, m_r,\ r = 1, \ldots, T, \\ & \sum_{i=1}^{m_r} \alpha_i^r y_i^r = 0;\ r = 1, \ldots, T, \end{aligned} \tag{3.9}$$

where

$$\Lambda = \operatorname{diag}(\overbrace{\lambda_1, \ldots, \lambda_1}^{m_1}, \ldots, \overbrace{\lambda_T, \ldots, \lambda_T}^{m_T}) \tag{3.10}$$

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

$$\widehat{Q} = \Lambda Q \Lambda + (I_n - \Lambda)\, K\, (I_n - \Lambda)\,, \tag{3.11}$$

whose corresponding multi-task kernel function can be expressed as

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

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 $r = 1, \ldots, T$. Here, to get the biases $b$ and $d_r$, we use again the support vectors, with $\alpha_i^r \neq 0$; hence

$$y_i^r(\lambda_r \langle w, \phi(x_i^r) \rangle + \lambda_r d + (1 - \lambda_r) \langle v_r, \phi_r(x_i^r) \rangle + (1 - \lambda_r) b_r) - p_i^r + \xi_i^r = 0.$$

However, we can only obtain the values for $\lambda_r b + (1 - \lambda_r) d_r$; observe that this does not change the prediction. Therefore, we can assume for $\lambda_r \neq 1$ that $b = 0$ and solve for $d_r$, while for $\lambda_r = 1$ we solve for $b$.

This dual problem is very similar to the one shown in (3.7) where there are also $T$ equality constraints, but the multi-task kernel matrix $\widehat{Q}$ is defined differently, dropping the $\mu$ hyperparameter and incorporating the $\lambda_r$ ones. Studying the influence of $\lambda_r$ hyperparameters in the dual problem we can describe the following cases:

- Reduction to an ITL approach:

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

  Here, the kernel matrix is $\widehat{Q} = K$, a block diagonal matrix.

- Reduction to a CTL approach:

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

  Thus, the kernel matrix is $\widehat{Q} = Q$, the one used in the standard CTL approach.

- Pure MTL approach:

  $$0 < \lambda_r < 1,\ r = 1, \ldots, T \implies$$
  $$h_r(\hat{x}) = \sum_{s=1}^{T} \lambda_s^2 \sum_{i=1}^{m_s} \{\alpha_i^s y_i^s k(x_i^s, \hat{x})\} + (1-\lambda_r)^2 \sum_{i=1}^{m_s} \{\alpha_i^s k_s(x_i^s, \hat{x})\} + \lambda_r b + (1-\lambda_r) d_r.$$

  Here, as described above, the kernel matrix is $\widehat{Q} = \Lambda Q \Lambda + (I_n - \Lambda)\, K\, (I_n - \Lambda)$ where none of the terms, either with $Q$ or $K$, are negligible.

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

### 3.1.3 Equivalence between Convex and Additive MTL SVM

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 $\mu$ in favor of those defining the convex combination of models, the hyperparameters $\lambda_r$. 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 provided a proposition, which we present next, to show the equivalence between additive and convex MTL SVM formulations where a common $\lambda$ is used, that is $\lambda_1 = \ldots = \lambda_T = \lambda$.

**Proposition 3.1.** *The additive MTL-SVM primal problem with parameters $C_{add}$ and $\mu$ (and possibly $\epsilon$), that is,*

$$\begin{aligned} \underset{w,\boldsymbol{v},\boldsymbol{b},\xi}{\arg\min} \quad & J(w, \boldsymbol{v}, \boldsymbol{b}, \boldsymbol{\xi}) = C_{add} \sum_{r=1}^{T} \sum_{i=1}^{m_r} \xi_i^r + \frac{1}{2} \sum_{r=1}^{T} \|v_r\|^2 + \frac{\mu}{2} \|w\|^2 \\ s.t. \quad & y_i^r (\langle w, \phi(x_i^r) \rangle + \langle v_r, \phi_r(x_i^r) \rangle + b_r) \geq p_i^r - \xi_i^r, \\ & \xi_i^r \geq 0,\ i = 1, \ldots, m_r,\ r = 1, \ldots, T, \end{aligned} \tag{3.12}$$

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

$$\begin{aligned} \underset{\hat{w},\hat{\boldsymbol{v}},\hat{b},\hat{\boldsymbol{d}},\boldsymbol{\xi}}{\arg\min} \quad & J(\hat{w}, \hat{\boldsymbol{v}}, \hat{b}, \hat{\boldsymbol{d}}, \boldsymbol{\xi}) = C_{conv} \sum_{r=1}^{T} \sum_{i=1}^{m_r} \xi_i^r + \frac{1}{2} \sum_{r=1}^{T} \|\hat{v}_r\|^2 + \frac{1}{2} \|\hat{w}\|^2 \\ s.t. \quad & y_i^r \left( \lambda \left\{ \langle \hat{w}, \phi(x_i^r) \rangle + \hat{b} \right\} + (1-\lambda) \left\{ \langle \hat{v}_r, \phi_r(x_i^r) \rangle + \hat{d}_r \right\} \right) \geq p_i^r - \xi_i^r, \\ & \xi_i^r \geq 0;\ i = 1, \ldots, m_r,\ r = 1, \ldots, T, \end{aligned} \tag{3.13}$$

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

*Proof.* Making the change of variables $w = \lambda\hat{w}$, $v_r = (1-\lambda)\hat{v}_r$ and $b_r = \lambda\hat{b} + (1-\lambda)\hat{d}_r$ in the convex primal problem (3.13), we can write it as

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

Multiplying now the objective function by $(1-\lambda)^2$ we obtain the additive MTL-SVM primal problem (3.12) 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. □

The previous proposition shows the equivalence between the primal problems when we use the same values for all the hyperparameters $\lambda_r$, that is $\lambda_1 = \ldots = \lambda_T = \lambda$. However, the full convex formulation, with possibly different values for each $\lambda_r$, is more general than the additive one, and it allows us to define different degrees of specifity in each task.

Anyway, the equivalence result can also be obtained from the dual problems. Consider the dual problem of the convex formulation (3.9) with $\lambda_1 = \ldots = \lambda_T = \lambda$; multiplying the objective function by $\frac{1}{(1-\lambda)^2} > 0$ we get

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

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

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

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

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

which is the dual problem of the additive formulation.

Finally, observe that the results in Proposition 3.1 are only valid for $\lambda$ 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. This possibility of recovering the CTL and ITL approaches is not present in the additive formulation, where the behaviour is asymptotical.

### 3.1.4 Convex Multi-Task Learning L2-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 ignored is kept, but the errors which are larger than such margin are penalized using their squared value.

We present a convex MTL L2-SVM, which we proposed in Ruiz et al. (2021b). To the best of our knowledge, this is the first combination-based MTL approach for the L2-SVM. The primal problem is the following:

$$\begin{aligned} \underset{w,\boldsymbol{v},b,\boldsymbol{d},\xi}{\arg\min} \quad & J(w, \boldsymbol{v}, b, \boldsymbol{d}, \boldsymbol{\xi}) = \frac{C}{2} \sum_{r=1}^{T} \sum_{i=1}^{m_r} (\xi_i^r)^2 + \frac{1}{2} \sum_{r=1}^{T} \|v_r\|^2 + \frac{1}{2} \|w\|^2 \\ \text{s.t.} \quad & y_i^r \left(\lambda_r \left\{\langle w, \phi(x_i^r)\rangle + b\right\} + (1 - \lambda_r) \left\{\langle v_r, \phi_r(x_i^r)\rangle + d_r\right\}\right) \geq p_i^r - \xi_i^r, \end{aligned} \tag{3.14}$$

where $i = 1, \ldots, m_r$ and $r = 1, \ldots, T$. Here, 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

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

for classification.

As in the standard variant, the hyperparameter $C$ regulates the trade-off between the loss and the margin size of each task-specialized model $h_r$, while $\lambda_r$ indicates how specific or common these models are in the range of $[0, 1]$. Again, 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, albeit with task-specific biases.

To obtain the dual problem we define first the Lagrangian of problem (3.14):

$$\begin{aligned} &\mathcal{L}(w, \boldsymbol{v}, b, \boldsymbol{d}, \boldsymbol{\xi}, \boldsymbol{\alpha}) \\ &= \frac{C}{2} \sum_{r=1}^{T} \sum_{i=1}^{m_r} (\xi_i^r)^2 + \frac{1}{2} \sum_{r=1}^{T} \|v_r\|^2 + \frac{1}{2} \|w\|^2 \\ &\quad - \sum_{r=1}^{T} \sum_{i=1}^{m_r} \alpha_i^r \left\{y_i^r \left(\lambda_r \left\{\langle w, \phi(x_i^r)\rangle + b\right\} + (1 - \lambda_r) \left\{\langle v_r, \phi_r(x_i^r)\rangle + d_r\right\}\right) - p_i^r + \xi_i^r\right\}, \end{aligned} \tag{3.15}$$

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 $\boldsymbol{\alpha}$. To find the solutions of problem (3.15), we use the strong duality theorem, given in Theorem 1.34. The dual objective function is defined as

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

To obtain $\Theta(\boldsymbol{\alpha})$, we need to compute the gradients with respect to the primal variables and make them zero:

$$\begin{aligned}\nabla_w \mathcal{L} = \mathbf{0} &\implies w^* - \sum_{r=1}^{T} \lambda_r \sum_{i=1}^{m_r} \alpha_i^r \left\{y_i^r \phi(x_i^r)\right\} = \mathbf{0}, \\ \nabla_{v_r} \mathcal{L} = \mathbf{0} &\implies {v_r}^* - (1-\lambda_r) \sum_{i=1}^{m_r} \alpha_i^r \left\{y_i^r \phi_r(x_i^r)\right\} = \mathbf{0}, \\ \partial_b \mathcal{L} = 0 &\implies \sum_{r=1}^{T}\sum_{i=1}^{m_r} \alpha_i^r y_i^r = 0, \\ \partial_{d_r} \mathcal{L} = 0 &\implies \sum_{i=1}^{m_r} \alpha_i^r y_i^r = 0, \\ \partial_{\xi_i^r} \mathcal{L} = 0 &\implies C\xi_i^r - \alpha_i^r = 0.\end{aligned}$$

Using these results and substituting back in the Lagrangian we obtain

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

which can be shown to be equal to

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

The corresponding dual problem can then be expressed as

$$\begin{aligned}
\min_{\boldsymbol{\alpha}} \quad \Theta(\boldsymbol{\alpha}) &= \frac{1}{2}\frac{1}{C}\sum_{r=1}^{T}\sum_{i=1}^{m_r}(\alpha_i^r)^2 \\
&\quad + \frac{1}{2}\left\langle \sum_{r=1}^{T}\lambda_r \sum_{i=1}^{m_r}\alpha_i^r y_i^r \phi(x_i^r), \sum_{r=1}^{T}\lambda_r \sum_{i=1}^{m_r}\alpha_i^r y_i^r \phi(x_i^r)\right\rangle \\
&\quad + \frac{1}{2}\sum_{r=1}^{T}\left\langle (1-\lambda_r)\sum_{i=1}^{m_r}\alpha_i^r y_i^r \phi_r(x_i^r), (1-\lambda_r)\sum_{i=1}^{m_r}\alpha_i^r y_i^r \phi_r(x_i^r)\right\rangle \\
&\quad - \sum_{r=1}^{T}\sum_{i=1}^{m_r}\alpha_i^r p_i^r \\
\text{s.t.} \quad & 0 \le \alpha_i^r,\ i = 1, \ldots, m_r,\ r = 1, \ldots, T, \\
& \sum_{i=1}^{m_r}\alpha_i^r y_i^r = 0,\ r = 1, \ldots, T.
\end{aligned}$$

Using the kernel trick, we can write the dual problem with a matrix formulation in the following way:

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

As in the convex MTL L1-SVM, we are using the matrix

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

and the $n \times n$ identity matrix $I_n$. Note that instead of having the upper bound for the dual coefficients $\alpha_i^r$ we add a diagonal term to the MTL kernel matrix $\widehat{Q}$, as defined in (3.11), so we use the matrix $\widehat{Q} + \frac{1}{C}I_n$. Again, we use the support vectors, with $\alpha_i^r \neq 0$, to get the optimal biases. The support vectors must satisfy the equation

$$y_i^r(\lambda_r \langle w, \phi(x_i^r)\rangle + \lambda_r d + (1-\lambda_r)\langle v_r, \phi_r(x_i^r)\rangle + (1-\lambda_r)b_r) - p_i^r + \xi_i^r = 0,$$

and, for $\lambda \neq 1$, we assume that $b = 0$ to solve for $d_r$, while for $\lambda = 1$ we solve for $b$.

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 $\xi_i^r$ 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.

### 3.1.5 Convex Multi-Task Learning LS-SVM

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. A combination-based MTL approach using the additive formulation was given in Xu et al. (2014). As with the L1-SVM, we present an alternative convex formulation, which we proposed in Ruiz et al. (2021b). The primal problem for the convex MTL LS-SVM is the following one:

$$\begin{aligned} \underset{w,\boldsymbol{v},b,\boldsymbol{d},\xi}{\arg\min} \quad & J(w,\boldsymbol{v},b,\boldsymbol{d},\boldsymbol{\xi}) = \frac{C}{2}\sum_{r=1}^{T}\sum_{i=1}^{m_r}(\xi_i^r)^2 + \frac{1}{2}\sum_{r=1}^{T}\|v_r\|^2 + \frac{1}{2}\|w\|^2 \\ \text{s.t.} \quad & y_i^r\left(\lambda_r\left\{\langle w,\phi(x_i^r)\rangle + b\right\} + (1-\lambda_r)\left\{\langle v_r,\phi_r(x_i^r)\rangle + d_r\right\}\right) = p_i^r - \xi_i^r, \end{aligned} \tag{3.16}$$

where $i = 1,\ldots,m_r$ and $r = 1,\ldots,T$. Observe that it differs with the L1-SVM in (3.8) and the L2-SVM in (3.14) because the inequalities in the constraints are replaced by equalities. Anyway, 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

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

for classification.

As in the standard variant, the hyperparameter $C$ regulates the trade-off between the loss and the margin size, while $\lambda_r$, 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 with task-specific biases.

The Lagrangian of problem (3.16) is

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

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 of the slack variables, and $\boldsymbol{\alpha}$ that

of the Lagrange multipliers. The dual objective function is defined as

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

The gradients with respect to the primal variables are

$$\nabla_w \mathcal{L} = \mathbf{0} \implies w^* - \sum_{r=1}^{T} \lambda_r \sum_{i=1}^{m_r} \alpha_i^r y_i^r \phi(x_i^r) = \mathbf{0}, \tag{3.17}$$

$$\nabla_{v_r} \mathcal{L} = \mathbf{0} \implies {v_r}^* - (1-\lambda_r) \sum_{i=1}^{m_r} \alpha_i^r y_i^r \phi_r(x_i^r) = \mathbf{0}, \tag{3.18}$$

$$\partial_b \mathcal{L} = 0 \implies \sum_{r=1}^{T} \sum_{i=1}^{m_r} \alpha_i^r y_i^r = 0, \tag{3.19}$$

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

$$\partial_{\xi_i^r} \mathcal{L} = 0 \implies C\xi_i^r - \alpha_i^r = 0. \tag{3.20}$$

By applying the strong duality theorem, it is necessary to maximize the dual problem to find a solution for the primal one. Since all $\alpha_i^r \in \mathbb{R}$ are feasible and the Lagrangian is differentiable, to define the dual problem we can write

$$\frac{\partial \mathcal{L}}{\partial \alpha_i^r} = 0 \implies y_i^r \left(\lambda_r \left\{\langle w, \phi(x_i^r)\rangle + b\right\} + (1-\lambda_r)\left\{\langle v_r, \phi_r(x_i^r)\rangle + d_r\right\}\right) = p_i^r - \xi_i^r,$$

and, using (3.17) and (3.18) to substitute $w$ and $v_r$, and (3.20) to replace $\xi_i^r$, we can express this condition as

$$y_i^r \left(\lambda_r \left\{\left\langle \sum_{s=1}^{T} \lambda_s \sum_{j=1}^{m_s} \alpha_j^s y_j^s \phi(x_j^s), \phi(x_i^r)\right\rangle + b\right\}\right) + y_i^r \left((1-\lambda_r)\left\{\left\langle (1-\lambda_r) \sum_{j=1}^{m_r} \alpha_j^r y_j^r \phi_r(x_j^r), \phi_r(x_i^r)\right\rangle + d_r\right\}\right) = p_i^r - \frac{1}{C}\alpha_i^r$$

and hence

$$\left(\lambda_r \left\{\sum_{s=1}^{T} \lambda_s \sum_{j=1}^{m_s} \alpha_j^s y_i^r y_j^s k(x_j^s, x_i^r) + y_i^r b\right\}\right) + \left((1-\lambda_r)\left\{(1-\lambda_r) \sum_{j=1}^{m_r} \alpha_j^r y_i^r y_j^r k_r(x_j^r, x_i^r) + y_i^r d_r\right\}\right) + \frac{1}{C}\alpha_i^r = p_i^r.$$

These equations, alongside those corresponding to the conditions (3.19), allows us to express the dual problem as the following system of equations

$$\left[\begin{array}{c|c|c} 0 & \mathbf{0}_T^\intercal & \boldsymbol{y}^\intercal \Lambda \\ \hline \mathbf{0}_T & \mathbf{0}_{T\times T} & A^\intercal Y (I_n - \Lambda) \\ \hline \boldsymbol{y} & YA & \widehat{Q} + \frac{1}{C} I_n \end{array}\right] \begin{pmatrix} b \\ d_1 \\ \vdots \\ d_T \\ \boldsymbol{\alpha} \end{pmatrix} = \begin{pmatrix} 0 \\ \mathbf{0}_T \\ \boldsymbol{p} \end{pmatrix}, \tag{3.21}$$

where $\mathbf{0}_T$ is the zero vector of length $T$, $\mathbf{0}_{T\times T}$ is the $T \times T$ zero matrix, $\Lambda$ is defined in (3.10), and we use the matrices

$$\underset{T\times N}{A^\intercal} = \begin{pmatrix} \overbrace{1\dots1}^{m_1} & \dots & \overbrace{0\dots0}^{m_T} \\ \ddots & \ddots & \ddots \\ 0\dots0 & \dots & 1\dots1 \end{pmatrix}, \quad \underset{N\times N}{Y} = \begin{pmatrix} y_1^1 & 0 & \dots & 0 \\ 0 & y_2^1 & \dots & 0 \\ \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & \dots & y_{m_T}^T \end{pmatrix}.$$

Again, $\widehat{Q}$ is the convex MTL kernel matrix defined in (3.11) and, as in the L2-SVM, a diagonal term $\frac{1}{C} I_n$ is added. Observe that in (3.21) the first equation is a linear combination of the rest, so it does not give any extra information. This is because, as in the previous cases, we cannot know the individual values of $b$ and $d_r$, but the values of the combination $\lambda_r b + (1 - \lambda_r) d_r$. As before, when $\lambda_r \neq 1$ we can assume that $b = 0$ and solve for $d_r$, that is, removing the first equation from the system.

In contrast to the L1 and L2-SVM, here, in the primal problem, the inequalities are replaced by equalities, which remove the bounds of the dual coefficients, thus leading to a problem that can be solved as a linear system of equations.

## 3.2 Convex Multi-Task Learning with Neural Networks

The convex MTL formulation is easily applicable and interpretable, so it has good properties not only 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 makes 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, and Deep Neural Networks (DNNs) 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 NNs.

In particular, in Chapter 2 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 Section 2.3 some of the most relevant approaches to MTL with NNs. 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 3.1; 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

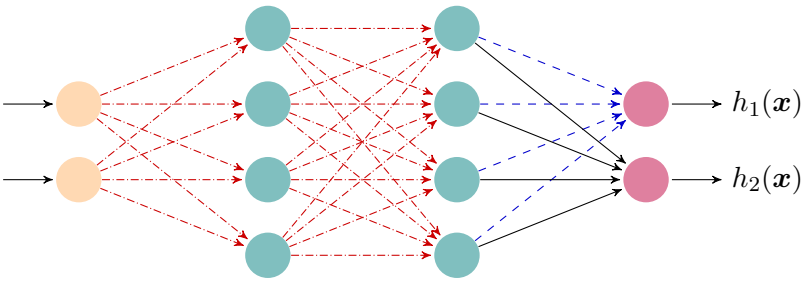



FIGURE 3.1: *Hard Sharing* Neural Network for two tasks and a two-dimensional input. The input neurons are shown in orange, the hidden ones in cyan and the output ones in magenta. Assuming a sample belonging to task 1 is used, the shared weights updated in training are represented in red, and in blue the updated specific weights.

close in some sense; see Long and Wang (2015); Yang and Hospedales (2017b). However, to the best of our knowledge, we proposed 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, that we present next.

### 3.2.1 Model Definition

Using the formulation of (3.1), we use neural networks to model the common part

$$g(\cdot; w, \Theta) = w^\intercal f(\cdot; \Theta) + b,$$

and the task-specific parts

$$g_r(\cdot; w_r, \Theta_r) = w_r^\intercal f_r(\cdot; \Theta_r) + d_r.$$

Here $\Theta$ and $\Theta_r$ are the sets of hidden weights, $w$ and $w_r$ are the output weights of the common and specific networks, respectively, and $b$ and $d_r$ the output biases. Observe that the feature transformations $f(\cdot; \Theta)$ and $f_r(\cdot; \Theta_r)$ are not fixed like the transformations $\phi(\cdot)$ and $\phi_r(\cdot)$ in the kernel methods; instead, here, they are automatically learned in the training process. The full MTL models are then

$$\begin{aligned} h_r(\cdot) &= \lambda_r g(\cdot; w, \Theta) + (1 - \lambda_r) g_r(\cdot; w_r, \Theta_r) \\ &= \lambda_r \{w^\intercal f(\cdot; \Theta) + b\} + (1 - \lambda_r)\{w_r^\intercal f_r(\cdot; \Theta_r) + d_r\}. \end{aligned} \tag{3.22}$$

This formulation offers a large flexibility 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 NNs, such as fully connected and convolutional, can be combined depending on the characteristics of each task. This combination of NNs can also be interpreted as an implementation of the Learning Using Privileged Information (LUPI) paradigm (Vapnik and Izmailov, 2015) presented in Section 2.5, i.e., the common network captures the privileged information for each of the tasks, since it can learn from more sources.

The regularized risk corresponding to the convex MTL NN is

$$\hat{R}_D = \sum_{r=1}^{T} \sum_{i=1}^{m_r} \ell(h_r(x_i^r), y_i^r) + \frac{\mu}{2} \left( \|w\|^2 + \sum_{r=1}^{T} \|w_r\|^2 + \Omega(\Theta) + \Omega(\Theta_r) \right).$$

Here, $h_r$ is defined as in equation (3.22), and $\Omega(\Theta)$ and $\Omega(\Theta_r)$ represents the regularization of the set of hidden weights of the common and specific networks, such as the $L_2$ regularization.

Regarding the training, 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 $\mathcal{P}$:

$$\nabla_{\mathcal{P}} \ell(h_t(x_i^t), y_i^t) = \frac{\partial}{\partial \hat{y}_i^t} \ell(\hat{y}_i^t, y_i^t)|_{\hat{y}_i^t = h_t(x_i^t)} \nabla_{\mathcal{P}} h_t(x_i^t).$$

Recall that we are using the formulation presented in (3.22), where we make a distinction between output weights $w, w_t$ and hidden parameters $\Theta, \Theta_t$. Then, the corresponding gradients of $h_t$ needed to compute the loss gradients are

$$\begin{aligned}
&\nabla_w h_t(x_i^t) = \lambda_t f(x_i^t, \Theta), && \nabla_\Theta h_t(x_i^t) = \lambda_t w^\intercal \nabla_\Theta f(x_i^t, \Theta; \\
&\nabla_{w_t} h_t(x_i^t) = (1-\lambda_t) f_t(x_i^t, \Theta), && \nabla_{\Theta_t} h_t(x_i^t) = (1-\lambda_t) w^\intercal \nabla_{\Theta_t} f_t(x_i^t, \Theta_t; \\
&\nabla_{w_r} h_t(x_i^t) = 0, && \nabla_{\Theta_r} h_t(x_i^t) = 0, \text{ for } r \neq t.
\end{aligned} \tag{3.23}$$

Putting all together, the gradient of the loss with respect to, for example, $w$ 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 backpropagation as the loss gradients with respect to the common parameters are scaled by $\lambda$, 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 backpropagation 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

$$\begin{aligned}
w^{\tau+1} &\leftarrow w^\tau - \eta \left[ \frac{\partial}{\partial \hat{y}_i^t} \ell(\hat{y}_i^t, y_i^t)|_{\hat{y}_i^t = h_t(x_i^t)} \lambda_t f(x_i^t, \Theta) + \mu w^\tau \right], \\
\Theta^{\tau+1} &\leftarrow \Theta^\tau - \eta \left[ \frac{\partial}{\partial \hat{y}_i^t} \ell(\hat{y}_i^t, y_i^t)|_{\hat{y}_i^t = h_t(x_i^t)} \lambda_t w^\intercal \nabla_\Theta f(x_i^t, \Theta + \mu \nabla_\Theta \Omega(\Theta) \right],
\end{aligned} \tag{3.24}$$

while the update rules for the $t$-th task network parameters would be

$$\begin{aligned}
w_t^{\tau+1} &\leftarrow w_t^\tau - \eta_t \left[ \frac{\partial}{\partial \hat{y}_i^t} \ell(\hat{y}_i^t, y_i^t)|_{\hat{y}_i^t = h_t(x_i^t)} (1-\lambda_t) f(x_i^t, \Theta) + \mu w_t^\tau \right], \\
\Theta_t^{\tau+1} &\leftarrow \Theta_t^\tau - \eta_t \left[ \frac{\partial}{\partial \hat{y}_i^t} \ell(\hat{y}_i^t, y_i^t)|_{\hat{y}_i^t = h_t(x_i^t)} (1-\lambda_t) w^\intercal \nabla_{\Theta_t} f(x_i^t, \Theta_t + \mu \nabla_{\Theta_t} \Omega(\Theta_t) \right],
\end{aligned} \tag{3.25}$$

and the parameters from the rest of task-specific networks are not updated. That is, no specific algorithm has to be developed for training the convex MTL NN. In (3.24) and (3.25) we have shown the update rules for vanilla SGD, but any other algorithm, e.g., Adam (Kingma and Ba, 2015), can be used scaling properly the loss gradients.

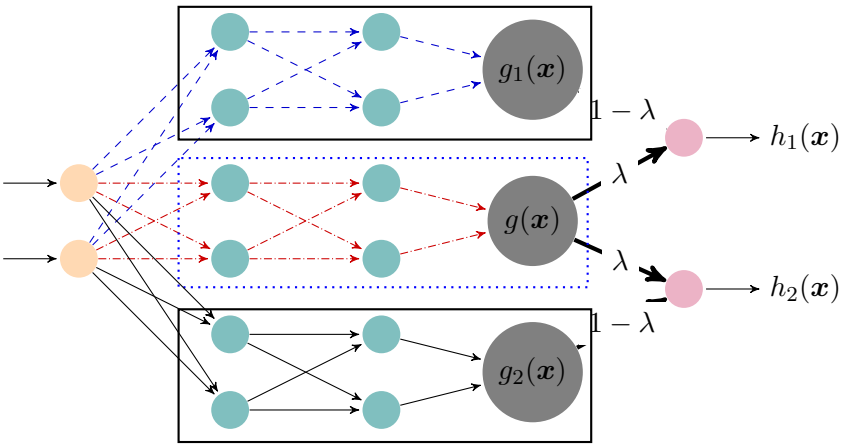



FIGURE 3.2: Convex MTL neural network for two tasks and a two-dimensional input. Assuming a sample belonging to task 1 is used, the updated shared weights are represented in red, and in blue the updated specific weights. Specific networks are framed in black boxes and the common one in a blue box. The 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 $\lambda$ and $1-\lambda$ of the convex combination.

**Algorithm 1:** Forward pass for Convex MTL neural network.

**Input:** $X_{\text{mb}}, t_{\text{mb}}$ `// Minibatch data and task labels`
**Output:** $f$ `// Forward pass for the minibatch`
**Data:** $\lambda$ `// 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**

### 3.2.2 Implementation Details

Our implementation of the convex MTL NN is based on `PyTorch` (Paszke et al., 2019). Although we include the gradient expressions in equation (3.23), the `PyTorch` package implements automatic differentiation, so we do not have to explicitly implement the gradients. 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[1] is a general framework to combine in a convex manner any `Pytorch` modules and select which ones should be

[1] https://github.com/carlosruizp/convexMTLPyTorch

used and updated in each training iteration, as well as using the corresponding modules for the prediction stage.

In Algorithm 1 we show the pseudocode 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.

## 3.3 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 $r = 1, \ldots, T$ 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 $\lambda_r, r = 1, \ldots, T$. The goal is to select the parameters $\lambda_r \in [0, 1]$ that minimize the training error

$$\hat{R}_D(\lambda_1, \ldots, \lambda_T) = \sum_{r=1}^{T} \sum_{i=1}^{m_r} \ell(\lambda_r g(x_i^r) + (1 - \lambda_r) g_r(x_i^r), y_i^r),$$

which depends on the choice of the loss function $\ell$. In Ruiz et al. (2021b) we consider the training error with four popular loss functions:

- Hinge loss (classification):

$$\hat{R}_D(\lambda_1, \ldots, \lambda_T) = \sum_{r=1}^{T} \sum_{i=1}^{m_r} \left[1 - y_i^r \left\{\lambda_r g(x_i^r) + (1 - \lambda_r) g_r(x_i^r)\right\}\right]_+ . \tag{3.26}$$

- Squared hinge loss (classification):

$$\hat{R}_D(\lambda_1, \ldots, \lambda_T) = \sum_{r=1}^{T} \sum_{i=1}^{m_r} \left[1 - y_i^r \left\{\lambda_r g(x_i^r) + (1 - \lambda_r) g_r(x_i^r)\right\}\right]_+^2 . \tag{3.27}$$

- Absolute loss (regression):

$$\hat{R}_D(\lambda_1, \ldots, \lambda_T) = \sum_{r=1}^{T} \sum_{i=1}^{m_r} \left| y_i^r - \left\{\lambda_r g(x_i^r) + (1 - \lambda_r) g_r(x_i^r)\right\}\right| . \tag{3.28}$$

- Squared loss (regression):

$$\hat{R}_D(\lambda_1, \ldots, \lambda_T) = \sum_{r=1}^{T} \sum_{i=1}^{m_r} \left( y_i^r - \left\{\lambda_r g(x_i^r) + (1 - \lambda_r) g_r(x_i^r)\right\}\right)^2 . \tag{3.29}$$

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 $\epsilon$-insensitive loss, which is the regression error of the L1-SVM, when $\epsilon = 0$. The L2-SVM uses 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.

### 3.3.1 Unified Formulation

Observe that in the classification errors with the hinge loss of (3.26) and with the squared hinge loss of (3.27), each term of the sum has the form

$$u\left(1 - \{\lambda_r g(x_i^r) + (1-\lambda_r) g_r(x_i^r)\}\, , y_i^r\right)$$

with the function $u$ 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 absolute loss of (3.28) and the squared loss of (3.29), 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 $u(\cdot) = |\cdot|$ or $u = (\cdot)^2$ for the absolute and squared loss, respectively. Again, these terms can be expressed as

$$u\left(\lambda_r \left\{g(x_i^r) - g_r(x_i^r)\right\} + g_r(x_i^r) - y_i^r\right).$$

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

$$c_i^r = y_i^r (g_r(x_i^r) - g(x_i^r)),\ d_i^r = 1 - y_i^r g_r(x_i^r), \tag{3.30}$$

and for the regression one

$$c_i^r = g(x_i^r) - g_r(x_i^r),\ d_i^r = g_r(x_i^r) - y_i^r. \tag{3.31}$$

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 $\lambda_r$ is only present in the training error terms concerning task $r$. That is, using the change of variables (3.30) and (3.31), we can consider for each task $r = 1, \ldots, T$ the problem

$$\arg\min_{\lambda_r \in [0,1]} \mathcal{J}(\lambda_r) = \sum_{i=1}^{m_r} u(\lambda_r c_i^r + d_i^r),$$

where, for a simpler formulation, we remove the task index, that is

$$\arg\min_{\lambda\in[0,1]} \mathcal{J}(\lambda) = \sum_{i=1}^{m} u(\lambda c_i + d_i), \tag{3.32}$$

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 subdifferentials 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 $\lambda_r^*$ we will consider the subdifferential of $\mathcal{J}(\lambda)$ in the elbows, that is, the set

$$\partial\mathcal{J}(\lambda) = \{c \in \mathbb{R}, \mathcal{J}(z) - \mathcal{J}(\lambda) \leq c(z - \lambda),\ \forall z\}.$$

Recall that all our functions $\mathcal{J}(\lambda)$ are convex, so to get the minimum we can apply Fermat's Theorem (Bauschke and Combettes, 2011), namely

$$\lambda^* = \arg\min_{0\leq\lambda\leq 1} \mathcal{J}(\lambda) \iff (0 \in \partial\mathcal{J}(\lambda^*) \text{ and } \lambda^* \in (0,1)) \text{ or } \lambda^* = 0 \text{ or } \lambda^* = 1.$$

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.

### 3.3.2 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 3.3. Using the formulation of (3.32), the training error corresponding to the squared loss is

$$\arg\min_{\lambda\in[0,1]} \mathcal{J}(\lambda) = \sum_{i=1}^{m} (\lambda c_i + d_i)^2 .$$

In this case, $\mathcal{J}(\lambda)$ is a differentiable function, and its derivative is

$$\mathcal{J}'(\lambda) = \sum_{i=1}^{m} 2c_i(\lambda c_i + d_i).$$

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 3.4 the error function using 10 random pairs $(c_i, d_i)$ and its corresponding differential is shown.

### 3.3.3 Absolute Loss

The absolute function

$$f(x) = \begin{cases} -x, & x \leq 0, \\ x, & x > 0 \end{cases}$$

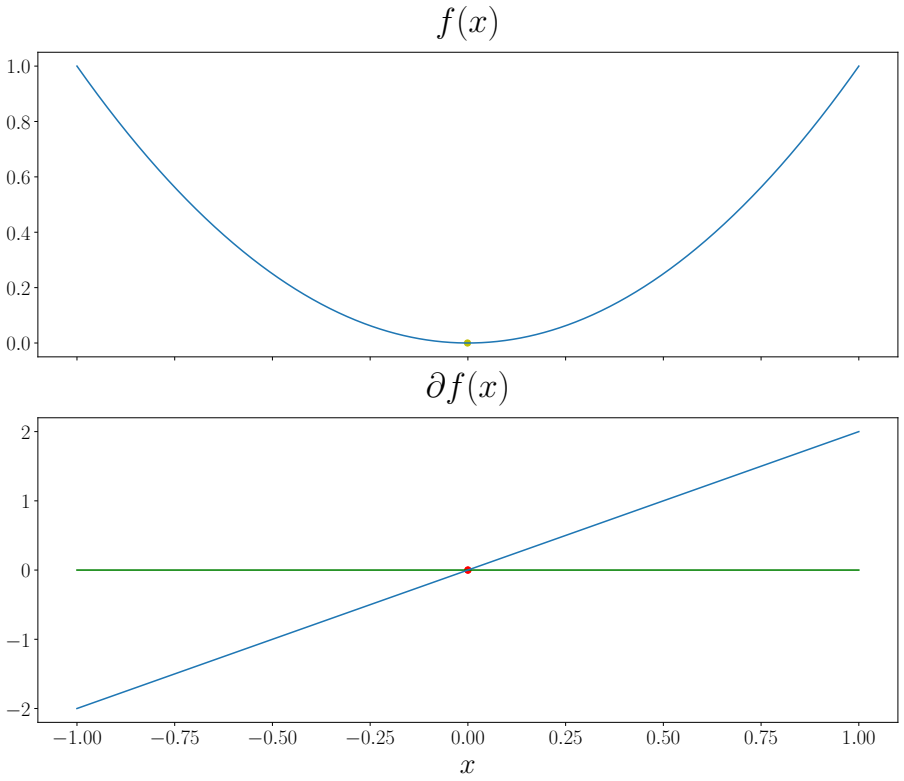



FIGURE 3.3: Squared value function (top) and its corresponding subgradient (bottom). The green line represents the 0 constant function. The yellow dot indicates the point minimizing the function.

is not differentiable at 0, but the subgradient can be expressed at any point as

$$\partial f(x) = \begin{cases} -1, & x < 0, \\ [-1, 1], & x = 0, \\ 1, & x > 0, \end{cases}$$

which is illustrated in Figure 3.5. Using the formulation of (3.32), the training error corresponding to the absolute value loss is

$$\underset{\lambda \in [0,1]}{\arg\min}\, \mathcal{J}(\lambda) = \sum_{i=1}^{m} |\lambda c_i + d_i| \,. \tag{3.33}$$

Observe that in each term of the sum the subdifferential is

$$\partial \, |\lambda c_i + d_i| = \begin{cases} -|c_i| \,, & \lambda c_i + d_i < 0, \\ [-|c_i| \,, |c_i|], & \lambda c_i + d_i = 0, \\ |c_i| \,, & \lambda c_i + d_i > 0. \end{cases} \tag{3.34}$$

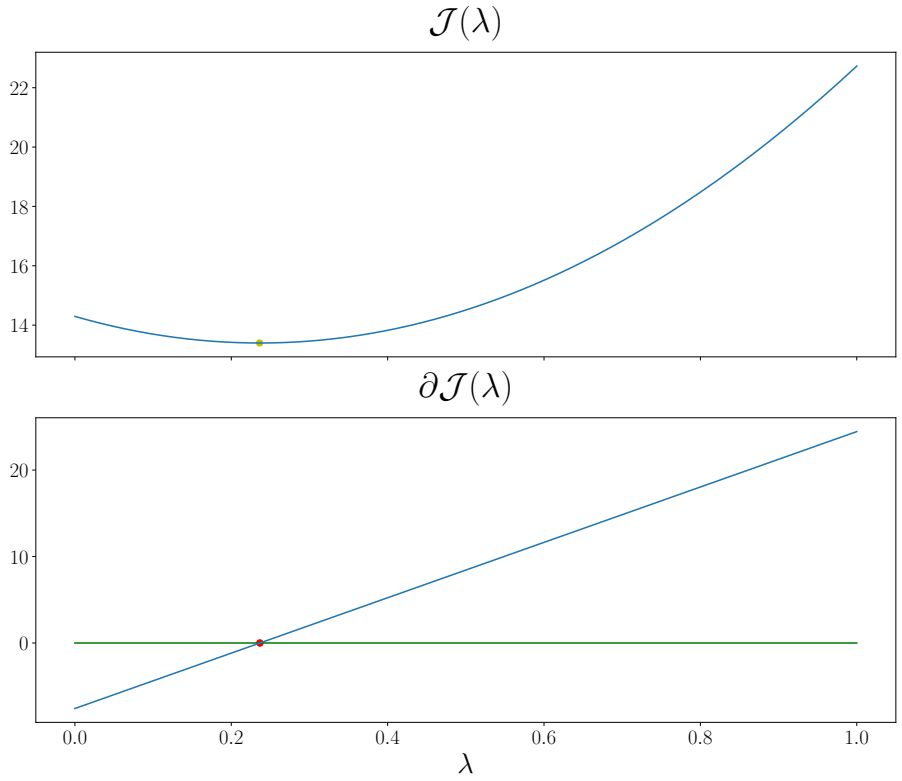



FIGURE 3.4: Error using the squared loss function (top) and its corresponding derivative (bottom). The green line represents the 0 constant function. The yellow dot is the point minimizing the error, and whose corresponding derivative contains the value 0.

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 $\lambda^*$.

**Proposition 3.2** (Optimal $\lambda^*$ with Absolute Value Loss)**.** *In problem* (3.33) $\lambda^* = 0$ *is optimal* iff

$$-\sum_{i:\, 0>\lambda_{(i)}} |c_{(i)}| + \sum_{i:\, 0<\lambda_{(i)}} |c_{(i)}| \leq 0.$$

*If this condition does not hold,* $\lambda^* \in (0,1)$ *is optimal* iff $\lambda^*$ *is a feasible elbow, that is,* $0 < \lambda^* = \lambda_{(k)} < 1$ *for some* $k = 1, \ldots, m$*, and*

$$-\sum_{i:\, \lambda_{(k)}>\lambda_{(i)}} |c_{(i)}| + \sum_{i:\, \lambda_{(k)}<\lambda_{(i)}} |c_{(i)}| \in \left[-|c_{(k)}|, |c_{(k)}|\right].$$

*If none of the previous conditions hold, then* $\lambda^* = 1$ *is optimal.*

*Proof.* Observe in (3.34) that, for each term $\partial\,|\lambda c_i + d_i|$, the change occurs at the elbow $\lambda_i = -d_i/c_i$. We can consider two cases:

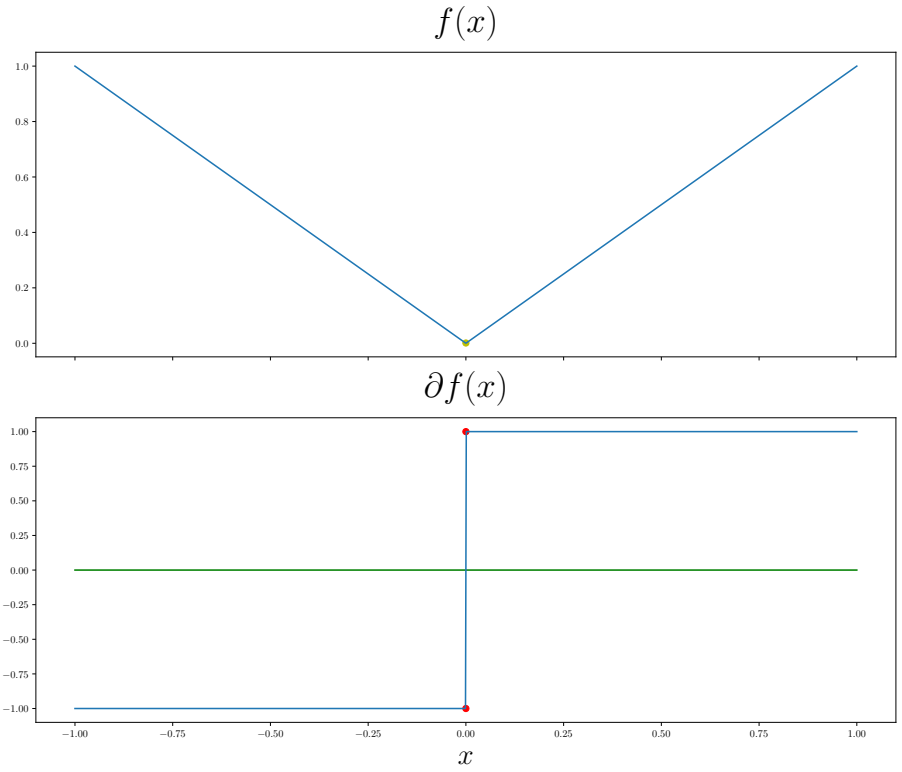



FIGURE 3.5: Absolute value function (top) and its corresponding subgradient (bottom). The green line represents the 0 constant function. The yellow dot indicates the point minimizing the function.

- $0 > c_i$; then

$$\partial \left|\lambda c_i + d_i\right| = \begin{cases} -c_i, & \lambda > \lambda_i, \\ [-\left|c_i\right|, \left|c_i\right|], & \lambda = \lambda_i, \\ c_i, & \lambda < \lambda_i. \end{cases}$$

- $0 < c_i$; then

$$\partial \left|\lambda c_i + d_i\right| = \begin{cases} -c_i, & \lambda < \lambda_i, \\ [-\left|c_i\right|, \left|c_i\right|], & \lambda = \lambda_i, \\ c_i, & \lambda > \lambda_i. \end{cases}$$

Consider the sorted elbow values $\lambda_{(1)}, \ldots, \lambda_{(m)}$. Recall that, since all functions share the same domain, the gradient of the sum is the sum of the gradients; then, for $\lambda \neq \lambda_{(i)}$ for

all $i = 1, \dots, m$,

$$\begin{aligned}\partial \mathcal{J}(\lambda) &= \sum_{\substack{i:\, 0>c_{(i)},\\ \lambda<\lambda_{(i)}}} c_{(i)} - \sum_{\substack{i:\, 0>c_{(i)},\\ \lambda>\lambda_{(i)}}} c_{(i)} - \sum_{\substack{i:\, 0<c_{(i)},\\ \lambda<\lambda_{(i)}}} c_{(i)} + \sum_{\substack{i:\, 0<c_{(i)},\\ \lambda>\lambda_{(i)}}} c_{(i)} \\ &= -\sum_{i:\, \lambda<\lambda_{(i)}} \operatorname{sign}(c_{(i)}) c_{(i)} + \sum_{i:\, \lambda>\lambda_{(i)}} \operatorname{sign}(c_{(i)}) c_{(i)} \\ &= -\sum_{i:\, \lambda<\lambda_{(i)}} \left|c_{(i)}\right| + \sum_{i:\, \lambda>\lambda_{(i)}} \left|c_{(i)}\right|.\end{aligned}$$

We can highlight two facts here: the value of $\partial \mathcal{J}(\lambda)$ is constant between elbows, and $\partial \mathcal{J}(\lambda)$ is a non-decreasing function. Next, when $\lambda = \lambda_{(k)}$ for some $k = 1, \dots, m$, we have that

$$\partial \mathcal{J}(\lambda_{(k)}) = \left[s_{(k)} - \left|c_{(k)}\right|, s_{(k)} + \left|c_{(k)}\right|\right],$$

where

$$s_{(k)} = -\sum_{i:\, \lambda_{(k)}<\lambda_{(i)}} c_{(i)} + \sum_{i:\, \lambda_{(k)}>\lambda_{(i)}} c_{(i)}.$$

Moreover, in $(\lambda_{(k-1)}, \lambda_{(k+1)})$ for $k = 2, \dots, m-1$, the value of the subdifferential is

$$\partial \mathcal{J}(\lambda) = \begin{cases} s_{(k)} - \left|c_{(k)}\right|, & \lambda < \lambda_{(k)}, \\ \left[s_{(k)} - \left|c_{(k)}\right|, s_{(k)} + \left|c_{(k)}\right|\right], & \lambda = \lambda_{(k)}, \\ s_{(k)} + \left|c_{(k)}\right|, & \lambda > \lambda_{(k)}. \end{cases}$$

That is, in these intervals containing one elbow $\partial \mathcal{J}(\lambda)$ is continuous, non-decreasing, and the values of the derivative before and after such elbow are contained in the subdifferential at the elbow. Thus, it is sufficient to look for the optimal value at the elbows. Then, using Fermat's Theorem, we want to find $\lambda^*$ such that $0 \in \partial \mathcal{J}(\lambda^*)$. We have three possible scenarios:

- $\partial \mathcal{J}(0) \geq 0$.
  Since $\partial \mathcal{J}(\lambda)$ is non-decreasing, the optimal value in $[0, 1]$ is $\lambda^* = 0$.
- $0 \in \left[s_{(k)} - \left|c_{(k)}\right|, s_{(k)} + \left|c_{(k)}\right|\right] = \partial \mathcal{J}(\lambda_{(k)})$, $0 < \lambda_{(k)} < 1$, for some $k = 1, \dots, m$.
  Then, the optimal value is $\lambda^* = \lambda_{(k)}$.
- $\partial \mathcal{J}(1) \leq 0$.
  None of the previous conditions hold, and since $\partial \mathcal{J}(\lambda)$ is non-decreasing, the optimal value in $[0, 1]$ is $\lambda^* = 1$.

□

The meaning of this proposition is depicted in Figure 3.6, 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 takes a step at the elbows. Then, the parameter $\lambda^*$ is optimal when $0 \in \partial J(\lambda^*)$.

Observe that, once the elbows are sorted, the cost of computing the optimal value $\lambda^*$ is linear, since we just have to compute the quantities $s_{(k)}$ for $k = 1, \dots, m$, and $s_{(k+1)} = s_{(k)} + 2\left|c_{(k)}\right|$ for each $k$. At each step, we check the optimality condition $0 \in [s_{(k)} - \left|c_{(k)}\right|, s_{(k)} + \left|c_{(k)}\right|]$. Thus, the bottleneck of the procedure is sorting the elbows, which supposes an average cost of $O(m \log(m))$.

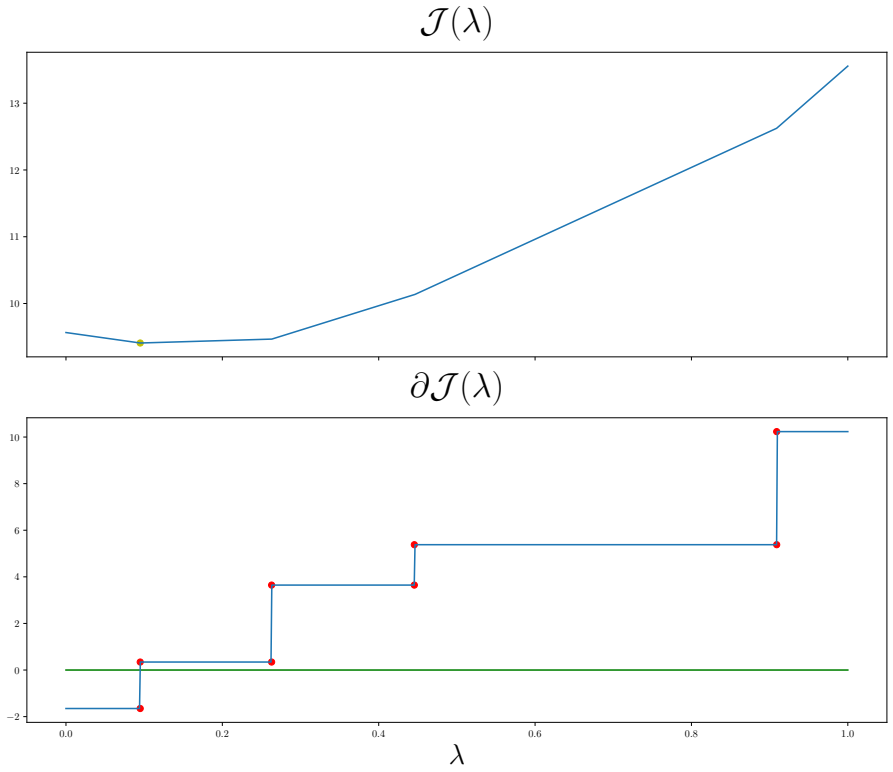



FIGURE 3.6: Error using the absolute loss function (top) and its corresponding subgradient (bottom). The green line represents the 0 constant function. Red dots mark the extremes of the subdifferential intervals of non-differentiable points. The yellow dot is the point minimizing the error, and whose corresponding subgradient contains the value 0.

### 3.3.4 Hinge Loss

The positive part function

$$f(x) = \begin{cases} 0, & x \leq 0, \\ x, & x > 0 \end{cases}$$

is not differentiable at 0, but, again, the subgradient can be expressed at any point as

$$\partial f(x) = \begin{cases} 0, & x < 0, \\ [0,1], & x = 0, \\ 1, & x > 0, \end{cases}$$

which is illustrated in Figure 3.7. Using the formulation of (3.32), the training error corresponding to the hinge loss is

$$\arg\min_{\lambda \in [0,1]} \mathcal{J}(\lambda) = \sum_{i=1}^{m} [\lambda c_i + d_i]_+ . \tag{3.35}$$

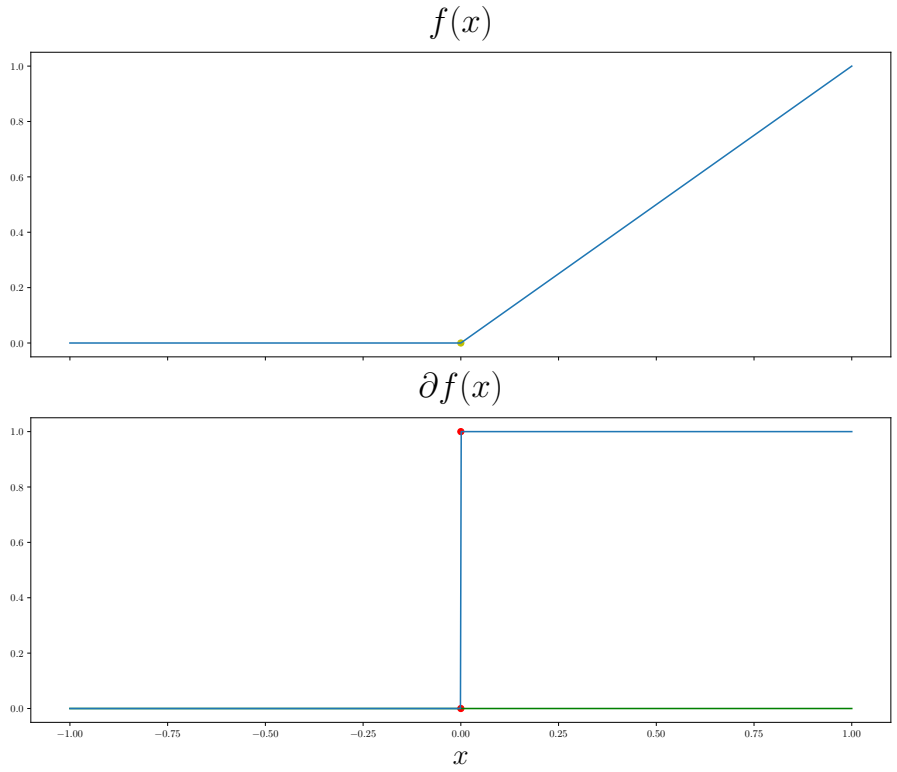



FIGURE 3.7: Positive part function (top) and its corresponding subgradient (bottom). The green line represents the 0 constant function. The yellow dot indicates the point minimizing the function.

Observe that in each term of the sum the subdifferential is

$$\partial \left[\lambda c_i + d_i\right]_+ = \begin{cases} 0, & \lambda c_i + d_i < 0, \\ [\min(0, c_i), \max(0, c_i)], & \lambda c_i + d_i = 0, \\ c_i, & \lambda c_i + d_i > 0. \end{cases} \tag{3.36}$$

That is, the elbows are related to the values $\frac{-d_i}{c_i}$, that can be sorted as $\lambda_{(1)} < \ldots < \lambda_{(m)}$. In Ruiz et al. (2021b, Proposition 2) we present the following result to get the optimal $\lambda^*$.

**Proposition 3.3** (Optimal $\lambda^*$ with Hinge Loss)**.** *In* (3.35)*,* $\lambda^* = 0$ *is optimal iff*

$$-\sum_{i:\, 0 > \lambda_{(i)}} \max\left(0, c_{(i)}\right) - \sum_{0 < \lambda_{(i)}} \min\left(0, c_{(i)}\right) \leq 0.$$

*If this condition does not hold, a value* $\lambda^* \in (0, 1)$ *is optimal for problem* (3.35) iff $\lambda^*$ *is a feasible elbow, that is,* $0 < \lambda^* = \lambda_{(k)} < 1$ *for some* $k = 1, \ldots, m$*, and*

$$-\sum_{i:\, \lambda_{(k)} > \lambda_{(i)}} \max\left(0, c_{(i)}\right) - \sum_{i:\, \lambda_{(k)} < \lambda_{(i)}} \min\left(0, c_{(i)}\right) \in \left[\min\left(0, c_{(k)}\right), \max\left(0, c_{(k)}\right)\right].$$

*If none of the previous conditions hold, then $\lambda^* = 1$ is optimal.*

*Proof.* Observe in (3.36) that, for each term $\partial\,|\lambda c_i + d_i|$, the change occurs at the elbow $\lambda_i = -d_i/c_i$. We can consider two cases:

- $0 > c_i$; then

$$\partial\,[\lambda c_i + d_i]_+ = \begin{cases} 0, & \lambda > \lambda_i, \\ [\min(0, c_i), \max(0, c_i)], & \lambda = \lambda_i, \\ c_i, & \lambda < \lambda_i. \end{cases}$$

- $0 < c_i$; then

$$\partial\,[\lambda c_i + d_i]_+ = \begin{cases} 0, & \lambda < \lambda_i, \\ [\min(0, c_i), \max(0, c_i)], & \lambda = \lambda_i, \\ c_i, & \lambda > \lambda_i. \end{cases}$$

Consider the sorted elbow values $\lambda_{(1)}, \ldots, \lambda_{(m)}$. Again, since all functions share the same domain, the gradient of the sum is the sum of the gradients, then, for $\lambda \neq \lambda_{(i)}$ for all $i = 1, \ldots, m$,

$$\partial\mathcal{J}(\lambda) = \sum_{\substack{i:\, 0>c_{(i)}, \\ \lambda<\lambda_{(i)}}} c_{(i)} + \sum_{\substack{i:\, 0<c_{(i)}, \\ \lambda>\lambda_{(i)}}} c_{(i)} = \sum_{i:\, \lambda<\lambda_{(i)}} \min\left(0, c_{(i)}\right) + \sum_{i:\, \lambda>\lambda_{(i)}} \max\left(0, c_{(i)}\right).$$

As before, we remark two facts: the value of $\partial\mathcal{J}(\lambda)$ is constant between elbows, and $\partial\mathcal{J}(\lambda)$ is a non-decreasing function. Moreover, when $\lambda = \lambda_{(k)}$ for some $k = 1, \ldots, m$, we have that

$$\partial\mathcal{J}(\lambda_{(k)}) = \left[s_{(k)} + \min\left(0, c_{(k)}\right), s_{(k)} + \max\left(0, c_{(k)}\right)\right],$$

where

$$s_{(k)} = \sum_{i:\, \lambda_{(k)}<\lambda_{(i)}} \min\left(0, c_{(i)}\right) + \sum_{i:\, \lambda_{(k)}>\lambda_{(i)}} \max\left(0, c_{(i)}\right).$$

Moreover, in $(\lambda_{(k-1)}, \lambda_{(k+1)})$ for $k = 2, \ldots, m-1$, the value of the subdifferential is

$$\partial\mathcal{J}(\lambda) = \begin{cases} s_{(k)} + \min\left(0, c_{(k)}\right), & \lambda < \lambda_{(k)}, \\ \left[s_{(k)} + \min\left(0, c_{(k)}\right), s_{(k)} + \max\left(0, c_{(k)}\right)\right], & \lambda = \lambda_{(k)}, \\ s_{(k)} + \max\left(0, c_{(k)}\right), & \lambda > \lambda_{(k)}. \end{cases}$$

That is, in these intervals $\partial\mathcal{J}(\lambda)$ is continuous, non-decreasing, and the values of the derivative in the open intervals with one elbow are contained in the that elbow subdifferential; thus, it is sufficient to look for the optimal value at the elbows. Then, using Fermat's Theorem, we want to find $\lambda^*$ such that $0 \in \partial\mathcal{J}(\lambda^*)$. We have three possible scenarios:

- $\partial\mathcal{J}(0) \geq 0$.
  Since $\partial\mathcal{J}(\lambda)$ is non-decreasing, the optimal value in $[0, 1]$ is $\lambda^* = 0$.
- $0 \in \left[s_{(k)} + \min\left(0, c_{(k)}\right), s_{(k)} + \max\left(0, c_{(k)}\right)\right] = \partial\mathcal{J}(\lambda_{(k)})$, $0 < \lambda_{(k)} < 1$ for some $k = 1, \ldots, m$.
  Then, the optimal value is $\lambda^* = \lambda_{(k)}$.

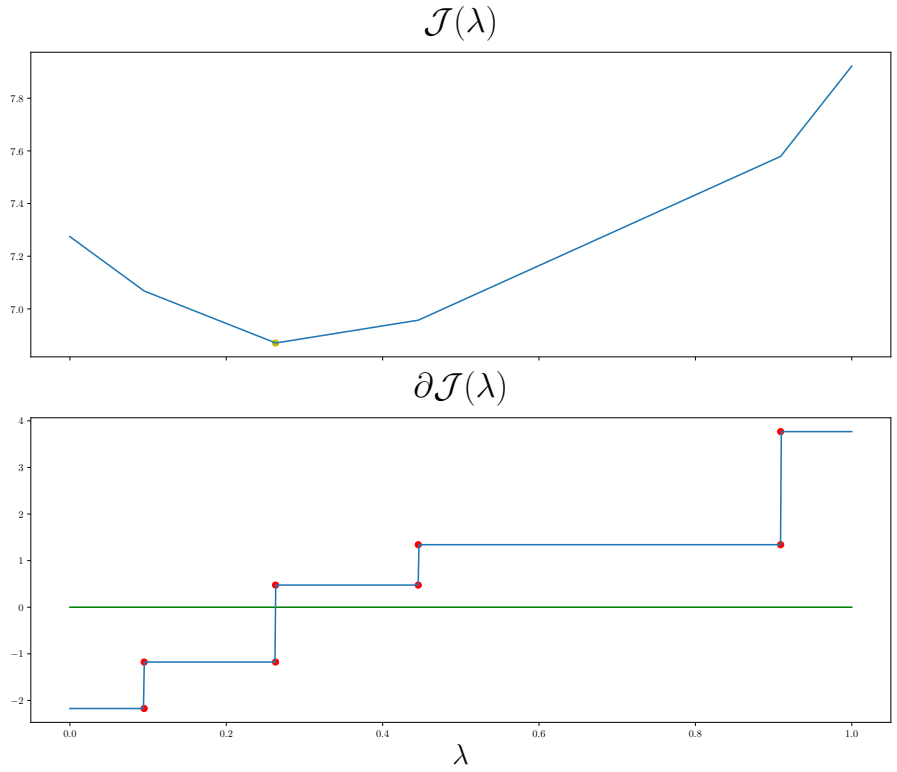



FIGURE 3.8: Error using the hinge loss function (top) and its corresponding subgradient (bottom). The green line represents the 0 constant function. Red dots mark the extremes of the subdifferential intervals of non-differentiable points. The yellow dot is the point minimizing the error, and whose corresponding subgradient contains the value 0.

- $\partial\mathcal{J}(1) \leq 0$.
  None of the previous conditions hold, and since $\partial\mathcal{J}(\lambda)$ is non-decreasing, the optimal value in $[0, 1]$ is $\lambda^* = 1$.

□

Again, the results of the proposition are illustrated in Figure 3.8, 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 an interval.

As with the absolute loss, once the elbows are sorted, the cost of computing the optimal value $\lambda^*$ is linear, since we just have to compute the quantities $s_{(k)}$ for $k = 1, \ldots, m$, and now $s_{(k+1)} = s_{(k)} + \left|c_{(k)}\right|$ for each $k$. At each step, we check the optimality condition $0 \in [s_{(k)} - \min\left(0, c_{(k)}\right), s_{(k)} + \max\left(0, c_{(k)}\right)]$. As before, the largest cost of the procedure is sorting the elbows, and it has an average cost of $O(m \log(m))$.

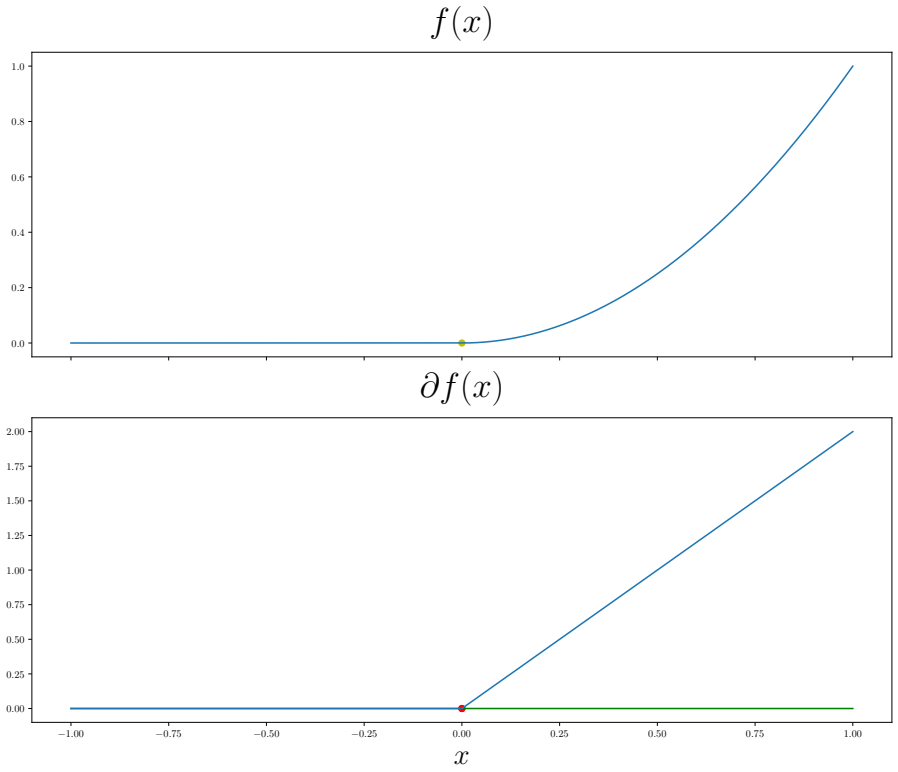



FIGURE 3.9: Positive part function (top) and its corresponding subgradient (bottom). The green line represents the 0 constant function. The yellow dot indicates one point minimizing the function.

### 3.3.5 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

$$\partial f(x) = \begin{cases} 0, & x \leq 0, \\ 2x, & x > 0, \end{cases}$$

which is illustrated in Figure 3.7. Although the gradient can be defined at any point, the definition by parts makes it difficult to get the optimal value $\lambda^*$ that minimizes the corresponding training error. Using the formulation of (3.32), the training error corresponding to the squared hinge loss is

$$\arg\min_{\lambda \in [0,1]} \mathcal{J}(\lambda) = \sum_{i=1}^{m} [\lambda c_i + d_i]_+^2 . \tag{3.37}$$

In each term of the sum the subdifferential is

$$\partial \left[\lambda c_i + d_i\right]_+^2 = \begin{cases} 0, & \lambda c_i + d_i \leq 0, \\ 2c_i(\lambda c_i + d_i), & \lambda c_i + d_i > 0. \end{cases} \tag{3.38}$$

The elbows are again related to the values $\frac{-d_i}{c_i}$, that can be sorted to write the elbows as $\lambda_{(1)} < \ldots < \lambda_{(m)}$. In Ruiz et al. (2021b, Proposition 2) we give the following result to get the optimal $\lambda^*$ when using the squared hinge loss.

**Proposition 3.4** (Optimal $\lambda^*$ with Squared Hinge Loss)**.** *In* (3.37)*,* $\lambda^* = 0$ *is optimal iff*

$$-\sum_{\substack{i:\, 0>c_{(i)},\\ 0<\lambda_{(i)}}} 2c_i d_i - \sum_{\substack{i:\, 0<c_{(i)},\\ 0>\lambda_{(i)}}} 2c_i d_i \leq 0.$$

*If this condition does not hold, consider the sorted list of elbows* $\lambda_{(1)}, \ldots, \lambda_{(m)}$*; then, for each interval between elbows* $(\lambda_{(k)}, \lambda_{(k+1)})$*, we define the value* $\widehat{\lambda}_k$ *for* $k = 1, \ldots, m-1$ *as*

$$\widehat{\lambda}_{(k)} = -\frac{\sum_{i:\, \lambda_{(k+1)} \geq \lambda_{(i)}} \max\left(0, c_{(i)}\right) d_{(i)} + \sum_{i:\, \lambda_{(k)} \leq \lambda_{(i)}} \min\left(0, c_{(i)}\right) d_{(i)}}{\sum_{i:\, \lambda_{(k+1)} \geq \lambda_{(i)}} \max\left(0, c_{(i)}\right)^2 + \sum_{i:\, \lambda_{(k)} \leq \lambda_{(i)}} \min\left(0, c_{(i)}\right)^2}. \tag{3.39}$$

*If* $\widehat{\lambda}_k$ *satisfies that* $\lambda_{(k)} \leq \widehat{\lambda}_k \leq \lambda_{(k+1)}$ *and* $0 \leq \widehat{\lambda}_{(k)} \leq 1$ *for some* $k = 1, \ldots, m$ *then* $\lambda^* = \widehat{\lambda}_k$ *is optimal. Finally, if none of the previous conditions holds,* (3.37) *has a minimum at* $\lambda^* = 1$*.*

*Proof.* Observe in (3.38) that, for each term $\partial \left|\lambda c_i + d_i\right|$, the change occurs at the elbow $\lambda_i = -d_i/c_i$. We can consider two cases:

- $0 > c_i$; then

$$\partial \left[\lambda c_i + d_i\right]_+^2 = \begin{cases} 0, & \lambda \geq \lambda_i, \\ 2c_i(\lambda c_i + d_i), & \lambda < \lambda_i. \end{cases}$$

- $0 < c_i$; then

$$\partial \left[\lambda c_i + d_i\right]_+^2 = \begin{cases} 0, & \lambda \leq \lambda_i, \\ 2c_i(\lambda c_i + d_i), & \lambda > \lambda_i. \end{cases}$$

Consider the sorted elbow values $\lambda_{(1)}, \ldots, \lambda_{(m)}$. Recall that, since all functions share the same domain, the gradient of the sum is the sum of the gradients, then, for all $\lambda \in \mathbb{R}$,

$$\partial \mathcal{J}(\lambda) = \sum_{\substack{i:\, 0>c_{(i)},\\ \lambda<\lambda_{(i)}}} 2c_{(i)}(\lambda c_{(i)} + d_{(i)}) + \sum_{\substack{i:\, 0<c_{(i)},\\ \lambda>\lambda_{(i)}}} 2c_{(i)}(\lambda c_{(i)} + d_{(i)}).$$

Observe that in both terms of the sum, $(\lambda c_{(i)} + d_{(i)}) > 0$, so the first term is negative and the second one positive; then, $\partial \mathcal{J}(\lambda)$ is a non-decreasing function. Also, we can check that $\partial \mathcal{J}(\lambda)$ is continuous. It is trivially continuous between elbows, and, at any elbow $\lambda_{(k)}$, its value is

$$\partial \mathcal{J}(\lambda_{(k)}) = \sum_{\substack{i:\, 0>c_{(i)},\\ \lambda_{(k)}<\lambda_{(i)}}} 2c_{(i)}(\lambda_{(k)} c_{(i)} + d_{(i)}) + \sum_{\substack{i:\, 0<c_{(i)},\\ \lambda_{(k)}>\lambda_{(i)}}} 2c_{(i)}(\lambda_{(k)} c_{(i)} + d_{(i)}),$$

and also, the left limit is

$$\lim_{\lambda \to \lambda_{(k)}^{-}} \partial \mathcal{J}(\lambda)$$
$$= 2c_{(k)}(\lambda c_{(k)} + d_{(k)}) + \sum_{\substack{i:\, 0>c_{(i)},\\ \lambda_{(k)}<\lambda_{(i)}}} 2c_{(i)}(\lambda c_{(i)} + d_{(i)}) + \sum_{\substack{i:\, 0<c_{(i)},\\ \lambda_{(k)}>\lambda_{(i)}}} 2c_{(i)}(\lambda c_{(i)} + d_{(i)}),$$

and the first term goes to zero, so the limit is $\partial \mathcal{J}(\lambda_{(k)})$. Analogously, we can find that the right limit is also $\partial \mathcal{J}(\lambda_{(k)})$. That is, $\partial \mathcal{J}(\lambda)$ is piece-wise defined, but it is continuous and non-decreasing; thus, using Fermat's Theorem, we want to find $\lambda^*$ such that $\partial \mathcal{J}(\lambda^*) = 0$. We have three possible scenarios:

- $\partial \mathcal{J}(0) \geq 0$.
  Since $\partial \mathcal{J}(\lambda)$ is non-decreasing, the optimal value in $[0, 1]$ is $\lambda^* = 0$.

- $0 = \partial \mathcal{J}(\lambda),\ 0 < \lambda < 1$.
  Between each pair of elbows $(\lambda_{(k)}, \lambda_{(k+1)})$, for $k = 1, \ldots, m-1$,

  $$\partial \mathcal{J}(\lambda) = \sum_{\substack{i:\, 0>c_{(i)},\\ \lambda_{(k)}\leq\lambda_{(i)}}} 2c_i(\lambda c_i + d_i) + \sum_{\substack{i:\, 0<c_{(i)},\\ \lambda_{(k+1)}\geq\lambda_{(i)}}} 2c_i(\lambda c_i + d_i);$$

  then, $\partial \mathcal{J}(\lambda) = 0$ has the solution

  $$\widehat{\lambda}_{(k)} = -\frac{\sum_{i:\, \lambda_{(k+1)}\geq\lambda_{(i)}} \max\left(0, c_{(i)}\right) d_{(i)} + \sum_{i:\, \lambda_{(k)}\leq\lambda_{(i)}} \min\left(0, c_{(i)}\right) d_{(i)}}{\sum_{i:\, \lambda_{(k+1)}\geq\lambda_{(i)}} \max\left(0, c_{(i)}\right)^2 + \sum_{i:\, \lambda_{(k)}\leq\lambda_{(i)}} \min\left(0, c_{(i)}\right)^2}.$$

  If $\lambda_{(k)} \leq \widehat{\lambda}_{(k)} \leq \lambda_{(k+1)}$ and $0 \leq \widehat{\lambda}_{(k)} \leq 1$, the optimal value is $\lambda^* = \widehat{\lambda}_{(k)}$.

- $\partial \mathcal{J}(1) \leq 0$.
  None of the previous conditions hold, and since $\partial \mathcal{J}(\lambda)$ is non-decreasing, the optimal value in $[0, 1]$ is $\lambda^* = 1$.

□

As with the other losses, the error function and respective gradient for 10 random pairs $(c_i, d_i)$ are represented in Figure 3.10. Now, the derivative is not constant between elbows, but a linear function, and we can observe that it is continuous, so it is more difficult to see the change of slope between elbows.

As with the two previous losses, once the elbows are sorted, the numerator and denominator that appear in (3.39) can be computed with a linear cost. For example, consider the sequence of numerators $N_{(k)} = \sum_{i:\, \lambda_{(k+1)}\geq\lambda_{(i)}} \max\left(0, c_{(i)}\right) d_{(i)} + \sum_{i:\, \lambda_{(k)}\leq\lambda_{(i)}} \min\left(0, c_{(i)}\right) d_{(i)}$; then,

$$N_{(k+1)} = N_{(k)} - \max\left(0, c_{(k+1)}\right) d_{(k+1)} + \min\left(0, c_{(k+1)}\right) d_{(k+1)}.$$

Analogously, for the sequence of denominators $D_{(k)} = \sum_{i:\, \lambda_{(k+1)}\geq\lambda_{(i)}} \max\left(0, c_{(i)}\right) d_{(i)} + \sum_{i:\, \lambda_{(k)}\leq\lambda_{(i)}} \min\left(0, c_{(i)}\right) d_{(i)}$, we have that

$$D_{(k+1)} = D_{(k)} - \max\left(0, c_{(k+1)}\right)^2 + \min\left(0, c_{(k+1)}\right)^2.$$

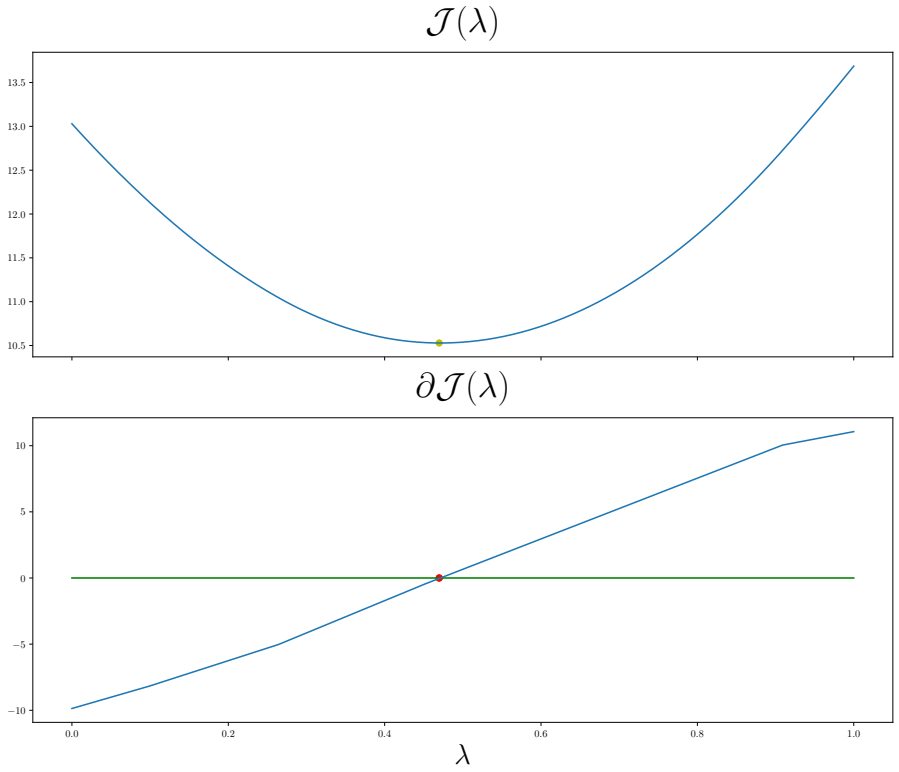



FIGURE 3.10: Error using the squared hinge loss function (top) and its corresponding subgradient (bottom). The green line represents the 0 constant function. The yellow dot is the point minimizing the error, whose corresponding derivative is 0.

Then, at each step $k = 1, \ldots, m-1$ we compute $\widehat{\lambda}_{(k)}$ and check if $\lambda_{(k)} \leq \widehat{\lambda}_{(k)} \leq \lambda_{(k+1)}$ and $0 \leq \widehat{\lambda}_{(k)} \leq 1$. Therefore, again, with sorted elbows we have a linear cost to check the optimality conditions, and the largest cost is the one corresponding to sorting the elbows, which is in average $O(m \log(m))$.

## 3.4 Conclusions

In this chapter, we have presented a convex formulation for MTL, which combines a common and task-specific parts. The idea of learning jointly two parts for MTL had already been proposed for SVMs, but in Section 3.1 we give a more general formulation, where the combination is convex and we have a specific hyperparameter $\lambda_r \in [0, 1]$ for each task. We present a proposition to show that a particular case of our formulation can be equivalent to the previous combination-based approach.

In Section 3.1 we have also described in depth the convex MTL formulation for kernel methods, concretely we present the convex MTL L1, L2, and LS-SVM. For these models, we give their corresponding primal problems and develop the dual ones.

Next, in Section 3.2 we apply the convex MTL strategy to NNs. Here, we also combine the outputs of a common and task-specific nets. The training procedure involves a joint optimization of all nets using backpropagation, which requires the gradient computation.

We describe how the gradients can be obtained with this MTL approach, but we give an implementation, using `PyTorch`, where we do not need to explicitly compute the gradients, we rely on the automatic differentiation instead.

One natural alternative for this MTL formulation is to consider convex combinations of models that have been previously trained. We present this idea in Section 3.3: we can take a common model fitted with the data from all tasks, and task-specific models trained only with the data corresponding to their task, and apply a convex combination of them to define an MTL model. It is necessary only to select the optimal convex combination parameters $\lambda_r \in [0, 1]$. We provide four propositions to find such parameters when using the absolute, squared, hinge and squared hinge losses.

In summary, we have proposed a novel convex formulation for MTL and we have described it thoroughly, presenting several models that apply this strategy: L1, L2, LS-SVM, NNs and also the convex combination of pre-trained models.

Chapter 4

# Adaptive Graph Laplacian for Multi-Task Learning

In Chapter 2 we have divided the Multi-Task Learning (MTL) strategies into feature-based, parameter-based and combination-based ones. 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 combination-based strategies, which combine a common part with task-specific parts in the models, a similar belief is held and, although this approach has many good properties, it relies on the assumption that all tasks can share the same common information. 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.

Some parameter-based approaches rely on enforcing low-rank matrices (Ando and Zhang, 2005; Chen et al., 2009; Pong et al., 2010), assuming, thus, that all tasks parameters belong to the same subspace. Others try to find the underlying task structure, either by task-relation learning or by clustering the tasks. In the task-relation learning we find strategies using Gaussian Processes and a Bayesian approach such as in Bonilla et al. (2007); Zhang and Yeung (2010). We also have the Graph Laplacian (GL) approaches, such as the works of Evgeniou et al. (2005) or Argyriou et al. (2013), where the tasks are assumed to be nodes of a graph, and the goal is to learn the weights on the edges, which determine the degree of relationship between tasks. In these works, iterated algorithms are used to learn both the model parameters and the graph of task relations. The clustering approaches are similar: they also use specific regularizers and alternating algorithms to find the clusters. However, these works using regularization as the mean to enforce the coupling between tasks are limited to linear models.

In general, in the GL strategies, the idea is to penalize 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 parameters $w_r$, 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

$$\sum_{r=1}^{T}\sum_{s=1}^{T}(A)_{rs}\left\|w_r - w_s\right\|^2, \tag{4.1}$$

where $(A)_{rs}$ are positive scalars that weight the pairwise distances. Although (4.1) is easy to compute for the linear case, it is not that direct in the case of kernel models where, as shown by the Representer Theorem, the optimal parameters $w_r^*$ are elements of an RKHS.

Besides, the choice 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 the models of two tasks 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 framework based on tensor products of RKHSs is presented in Section 4.1, and it is applied for the GL regularization with kernel methods in Section 4.1. Moreover, in Section the GL strategy is combined with the convex MTL one presented in Chapter 3; and we implement this convex GL approach for the L1, L2 and LS-SVM. Finally, a data-driven procedure to automatically select the weights of the adjacency matrix is given in Section 4.4.

## 4.1 Kernels for Multi-Task Learning

In MTL different functions have to be estimated, and we would like to capture the degree of relation between the tasks as well. Using kernels for these goals imposes some new challenges, that can be tackled from different perspectives. Here we propose a reformulation where a tensor product of scalar RKHSs 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 GL MTL problems and also give some examples of commonly used GL MT kernels.

### 4.1.1 Tensor Product of Reproducing Kernel Hilbert Spaces

Given two finite-dimensional vector spaces $V$ and $W$ over a field $F$, with dimensions $n$ and $m$, the tensor product of these spaces, $V \otimes W$, is associated with the bilinear map

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

such that

$$\begin{bmatrix} v_1 \\ \vdots \\ v_n \end{bmatrix} \otimes \begin{bmatrix} w_1 \\ \vdots \\ w_n \end{bmatrix} = \begin{bmatrix} v_1 w_1 \\ \vdots \\ v_1 w_n \\ \vdots \\ v_m w_1 \\ \vdots \\ v_m w_n \end{bmatrix};$$

this product is also called Kronecker product. The tensor product can be constructed from the basis of the vector spaces. Consider $B_V$ and $B_W$ the basis of $V$ and $W$, respectively; then the tensor product $V \otimes W$ is a vector space which has as basis the set $\{v \otimes w,\ v \in B_V,\ w \in B_W\}$. Since this definition seems recursive, we can also consider the interpretation of the tensor product as a quotient space. Let $R$ be the linear subspace of the Cartesian product $V \times W$ that is the span of the elements of one of the following forms:

$$\begin{aligned}(v_1 + v_2, w) - (v_1, w) - (v_2, w),\\ (v, w_1 + w_2) - (v, w_1) - (v, w_2),\\ (sv, w) - s(v, w),\\ (v, sw) - s(v, w),\end{aligned}$$

where $v, v_1, v_2 \in V$, $w, w_1, w_2 \in W$ and $s \in F$. Then, we can define the tensor product space as the quotient space $(V \times W)/R$. Observe that these are just the relations to ensure that the tensor product satisfies

$$\begin{aligned}(v_1 \otimes w) + (v_2 \otimes w) = ((v_1 + v_2) \otimes w),\\ (v \otimes w_1) + (v \otimes w_2) = (v \otimes (w_1 + w_2)),\\ s(v \otimes w) = (sv \otimes w),\\ s(v \otimes w) = (v \otimes sw).\end{aligned}$$

More concretely, we can call this definition as the algebraic tensor product of vector spaces and denote it as $V \otimes_{\text{alg}} W$, which is equivalent to $V \otimes W$ in the finite dimensional case. If $\mathcal{H}_1$ and $\mathcal{H}_2$ are finite-dimensional Hilbert spaces, then the tensor product space $\mathcal{H}_1 \otimes \mathcal{H}_2$ is a Hilbert space with inner product

$$\begin{array}{lllll}\langle , \rangle : (\mathcal{H}_1 \otimes \mathcal{H}_2) & \times & (\mathcal{H}_1 \otimes \mathcal{H}_2) \to & & \mathbb{R} \\ (f_1 \otimes f_2) & & (\hat{f}_1 \otimes \hat{f}_2) \to & \left\langle f_1, \hat{f}_1 \right\rangle \left\langle f_2, \hat{f}_2 \right\rangle & .\end{array} \tag{4.2}$$

Extending this definition to the infinite-dimensional case is not trivial, and we follow roughly the work of Kadison and Ringrose (1983). Consider the RKHSs $\mathcal{H}_1$ of functions $f : \mathcal{X}_1 \to \mathbb{R}$ and $\mathcal{H}_2$ of functions $g : \mathcal{X}_2 \to \mathbb{R}$; with the quotient space definition we can define the tensor product space $H_1 \otimes_{\text{alg}} H_2$ of functions

$$\begin{array}{lllll} f \otimes_{\text{alg}} g : \mathcal{X}_1 & \times & \mathcal{X}_2 \to & & \mathbb{R} \\ x_1 & & x_2 \to & f(x_1)g(x_2), & \end{array}$$

which is a pre-Hilbert space with the inner product defined in (4.2). We can complete this space, i.e. include the limits of the Cauchy series, to get the corresponding Hilbert space (Kadison and Ringrose, 1983), that we will denote as $\mathcal{H}_1 \otimes \mathcal{H}_2$.

To apply the Riesz Theorem in this Hilbert space it is necessary to check whether the evaluation functionals are continuous or, equivalently, bounded. Recall that given an RKHS $\mathcal{H}$ of functions $f : \mathcal{X} \to \mathbb{R}$, the evaluation functionals $E_x$ are defined as

$$\begin{array}{ll} E_x : \mathcal{H} & \to \mathbb{R} \\ f & \to f(x). \end{array}$$

**Proposition 4.1** (RKHS as tensor product of RKHSs)**.** *Let $\mathcal{H}_1$ and $\mathcal{H}_2$ be RKHSs ; then the space $\mathcal{H} = \mathcal{H}_1 \otimes \mathcal{H}_2$ , with evaluation functionals $E_{(x_1,x_2)}$ defined as*

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

*for $x_1 \otimes x_2 \in \mathcal{X}_1 \otimes \mathcal{X}_2$, is an RKHS with the inner product defined in (4.2).*

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

$$\left\| E_{(x_1,x_2)}(f_1 \otimes f_2) \right\| = \langle f_1(x_1), f_1(x_1) \rangle \langle f_2(x_2), f_2(x_2) \rangle \leq \|f_1(x_1)\|^2 \|f_2(x_2)\|^2 .$$

Since $\mathcal{H}_1$ and $\mathcal{H}_2$ are RKHSs, they have bounded evaluation functionals; then, the product $\|f_1(x_1)\| \|f_2(x_2)\|$ is also bounded. □

Recall also that, according to the Riesz theorem Whittaker (1991), if the evaluation functionals $E_x$ of a Hilbert space $\mathcal{H}$ are bounded, for every $x \in \mathcal{X}$ there exists a single $k^x \in \mathcal{H}$, such that

$$E_x(f) = \langle f, k^x \rangle , \forall f \in \mathcal{H}.$$

Therefore, if we define a function $k(x, \hat{x}) = \left\langle k^x, k^{\hat{x}} \right\rangle$, it is a kernel because it is positive definite, and with $k(x, \cdot) = k^x$ we can also check that it is a reproducing kernel. Now we define the kernel function for the tensor product of RKHSs.

**Proposition 4.2.** *Let $\mathcal{H}_1$ and $\mathcal{H}_2$ be RKHSs whose reproducing kernels are $K_1$ and $K_2$, respectively; then the function*

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

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

*Proof.* First, we define $(K_1 \otimes K_2)^{(x_1,x_2)} = K_1^{x_1} \otimes K_2^{x_2} \in \mathcal{H}_1 \otimes \mathcal{H}_2$, where for $x_1 \in \mathcal{X}_1$ $K_1^{x_1} \in \mathcal{H}_1$ is the element that satisfies

$$\langle K_1^{x_1}, f \rangle = f(x_1), \ \forall f \in \mathcal{H}_1,$$

and for $x_2 \in \mathcal{X}_2$ $K_2^{x_2} \in \mathcal{H}_2$ is the element that satisfies

$$\langle K_2^{x_2}, g \rangle = g(x_2), \ \forall g \in \mathcal{H}_2;$$

these elements $K_1^{x_1}$ and $K_2^{x_2}$ exist because $\mathcal{H}_1$ and $\mathcal{H}_2$ are RKHSs. Moreover, the reproducing kernels of such spaces are defined as

$$K_1(x_1, \hat{x_1}) = \left\langle K_1^{x_1}, K_1^{\hat{x_1}} \right\rangle$$

for $\mathcal{H}_1$, and as

$$K_2(x_2, \hat{x_2}) = \left\langle K_2^{x_2}, K_2^{\hat{x_2}} \right\rangle$$

for $\mathcal{H}_2$. Then, $(K_1 \otimes K_2)^{(x_1,x_2)}$ satisfies that for every $f \otimes g \in \mathcal{H}_1 \otimes \mathcal{H}_2$,

$$\left\langle (K_1 \otimes K_2)^{(x_1,x_2)}, f \otimes g \right\rangle = f(x_1) g(x_2).$$

The evaluation functionals are bounded as shown in the proof of Proposition 4.1, then we can apply the Riesz theorem: for every $(x_1, x_2) \in \mathcal{X}_1 \otimes \mathcal{X}_2$ there exists a single element $K^{(x_1,x_2)} \in \mathcal{H}_1 \otimes \mathcal{H}_2$ such that

$$\left\langle K^{(x_1,x_2)}, f \otimes g \right\rangle = f(x_1) \otimes g(x_2) = f(x_1)g(x_2) \; \forall f \otimes g \in \mathcal{H}_1 \otimes \mathcal{H}_2;$$

thus, this element must be $K^{(x_1,x_2)} = (K_1 \otimes K_2)^{(x_1,x_2)}$. Then, we can define

$$\begin{aligned}(K_1 \otimes K_2)((x_1, x_2), (\hat{x}_1, \hat{x}_2)) &= \left\langle (K_1 \otimes K_2)^{(x_1,x_2)}, (K_1 \otimes K_2)^{(\hat{x}_1,\hat{x}_2)} \right\rangle \\ &= \left\langle K_1^{x_1} \otimes K_2^{x_2}, K_1^{\hat{x}_1} \otimes K_2^{\hat{x}_2} \right\rangle \\ &= \left\langle K_1^{x_1}, K_1^{\hat{x_1}} \right\rangle \left\langle K_2^{x_2}, K_2^{\hat{x_2}} \right\rangle \\ &= K_1(x_1, \hat{x}_1) K_2(x_2, \hat{x}_2).\end{aligned}$$

We observe that $K_1 \otimes K_2$ is symmetric and positive-definite because both $K_1$ and $K_2$ are symmetric and positive definite, and, by construction, it has the reproducing property. □

Although these definitions are quite general and abstract, in our case of interest, MTL, our spaces are $\mathcal{X}_1 = \mathbb{R}^d$ and $\mathcal{X}_2 = \mathbb{R}^T$, that is, the feature space and the space of tasks. Consider the RKHS $\mathcal{H}_\mathcal{X}$ corresponding to the feature space $\mathcal{X} = \mathbb{R}^d$, with functions $f : \mathbb{R}^d \to \mathbb{R}$ and reproducing kernel $K_\mathcal{X}$. Also, consider the space of tasks is $\mathcal{T} = \mathbb{R}^T$, and let $M \in \mathbb{R}^{T \times T}$ be a symmetric positive-definite matrix; thus, we can express it as $M = B^\intercal B$. Using the feature map $\phi(u) = Bu$ for $u \in \mathbb{R}^T$, we define in this space the kernel function as

$$K_\mathcal{T}(u, \hat{u}) = \langle \phi(u), \phi(\hat{u}) \rangle = \langle Bu, B\hat{u} \rangle = \langle u, M\hat{u} \rangle = u^\intercal M \hat{u},$$

for every $u, \hat{u} \in \mathbb{R}^T$. Here, $\mathcal{H}_T$ is an RKHS, isomorphic to $\mathbb{R}^T$, with the reproducing kernel $K_\mathcal{T}$. Now, we can consider the tensor product space $\mathcal{H}_\mathcal{X} \otimes \mathbb{R}^T$. Given an instance $x$ from task $t$, we can encode with the vector

$$e_t = (0, \ldots, \overbrace{1}^{t}, \ldots, 0); \tag{4.3}$$

so $(\phi(x), e_t)$ is an element of $\mathcal{H}_\mathcal{X} \otimes \mathbb{R}^T$ and $\phi$ satisfies that $K_\mathcal{X}(x, \hat{x}) = \langle \phi(x), \phi(\hat{x}) \rangle$. Therefore, the kernel of the space $\mathcal{H}_\mathcal{X} \otimes \mathbb{R}^T$ is the following one:

$$\begin{aligned}(K_\mathcal{X} \otimes K_\mathcal{T})((x_i, e_r), (x_j, e_s)) &= K_\mathcal{X}(x_i, x_j) K_\mathcal{T}(e_r, e_s) \\ &= K_\mathcal{X}(x_i, x_j) e_r^\intercal M e_s \\ &= K_\mathcal{X}(x_i, x_j) M_{rs}.\end{aligned}$$

That is, although the theoretical framework of the tensor product of RKHSs is complex, the resulting kernel $(K_\mathcal{X} \otimes K_\mathcal{T})$ is a quite natural result: the kernel between instances of different tasks is a product of the kernel between features and the kernel between tasks. It is also important to observe that the matrix $M$ defines the similarity among tasks, since the encoding defined as in (4.3) does not provide information about such tasks. Unlike in the feature space, where the features trivially define the similarity between

instances, in the space of tasks it is necessary to define an adequate matrix $M$. We will call this kind of kernels MTL kernels or MT kernels.

**Definition 4.3** (MTL kernel)**.** Given a kernel function $K_{\mathcal{X}}$ defined on $\mathbb{R}^d \times \mathbb{R}^d$ and a positive definite matrix $M \in \mathbb{R}^{(T \times T)}$, a kernel function $K$ defined as

$$\begin{array}{rccl} K : (\mathbb{R}^d \times \mathbb{R}^T) & \times & (\mathbb{R}^d \times \mathbb{R}^T) \to & \mathbb{R} \\ (x_i, e_r) & & (x_j, e_s) \to & K_{\mathcal{X}}(x_i, x_j)(e_r^\intercal M e_s) \end{array}$$

is an MTL kernel.

This kind of kernels will be useful to express MTL problems with a CTL formulation. They also have a close connection with operator-valued kernels, concretely with the particular case of separable kernels, concepts that we define in the Appendix B.

### 4.1.2 Kernel extensions for Multi-Task Learning

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

$$\min_w \sum_{i=1}^{n} \ell(y_i, \langle w, \phi(x_i) \rangle) + \lambda \langle w, w \rangle . \tag{4.4}$$

Here, $\ell$ is the loss function and $\phi$ 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 $\ell$ and $\phi$. One crucial result for this kind of problems is the Representer Theorem, which states that any minimizer of problem (4.4) has the form

$$w = \sum_{i=1}^{n} c_j \phi(x_j). \tag{4.5}$$

Given $w$ represented as in (4.5), 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 $\phi$ only implicitly. In this subsection it is shown how a broad class of MTL problems can be expressed as regularized Single-Task Learning (STL) problems.

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 STL 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 $u_r \in \mathbb{R}^d : r = 1, \ldots, T$, so we define $\boldsymbol{u}^\intercal = (u_1^\intercal, \ldots, u_T^\intercal) \in \mathbb{R}^{Td}$. Then we want to minimize

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

where

$$J(\boldsymbol{u}) = \boldsymbol{u}^\intercal E \boldsymbol{u}$$

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 related.

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 $\langle B_r^\intercal \boldsymbol{w}, x\rangle$, where $B_r$ is a $p \times d$ matrix yet to be specified. One condition for $B_r$ is to be full rank $d$, so we can find such a $\boldsymbol{w}$. 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 kernel:

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

Using these feature maps we would like to write the MTL problem as a STL problem

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

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 4.4.** *Consider the MTL problem*

$$R(\boldsymbol{u}) = \sum_{r=1}^{T} \sum_{i=1}^{m} \ell(y_i^r, \langle u_r, x_i^r\rangle) + \mu (\boldsymbol{u}^\intercal E \boldsymbol{u}), \tag{4.6}$$

*and the CTL problem*

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

*if the feature matrix $B$ has full rank and we define the matrix $E$ in equation (4.6) as $E = (B^\intercal B)^{-1}$, then*

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

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

One important consequence of this result is that since we can solve the MTL problem (4.6) as the STL problem (4.4) with $\phi$ being the identity function, then we can apply the Representer Theorem. That is, the solution $\boldsymbol{w}^*$ of problem (4.4) has the form

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

Here, we are using the transformations $\phi(x_i^r) = B_r x_i^r$, and the prediction of a new instance $\hat{x}^s$ can be expressed as

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

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

**Lemma 4.5.** *Given a space $\mathcal{T}$ such that for every $r = 1, \ldots, T$ there are prescribed mappings $\psi_r : \mathcal{X} \to \mathcal{T}$, if $G$ is a kernel on $\mathcal{T} \times \mathcal{T}$, then*

$$K((x, r), (\hat{x}, s)) = G(\psi_r(x), \psi_s(t)),\ x, \hat{x} \in \mathcal{X},\ r, s = 1, \ldots, T,$$

*is a kernel that incorporates the multi-task information.*

The mappings described in Evgeniou et al. (2005) are

$$\psi_r(x) = B_r x,$$

where $B_r$ are the $p \times d$ matrices previously defined. Then, two examples of multi-task kernels using this lemma are given. The linear 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, applying Proposition 4.4, since $E^{-1} = B^\intercal B$, we can incorporate the task-regularizer information into the Gaussian kernel using that

$$\begin{aligned} \left\| B_r x - B_s y \right\|^2 &= x^\intercal B_r^\intercal B_r x + y^\intercal B_s^\intercal B_s y - 2 x_r^\intercal B_r^\intercal B_s y_s \\ &= x^\intercal (E^{-1})_{rr} x + y^\intercal (E^{-1})_{ss} y - 2 x_r^\intercal (E^{-1})_{rs} y_s, \end{aligned}$$

where $(E^{-1})_{rs}$ is the $s$-th column of the $r$-th row of matrix $E^{-1}$. That is, we use the task information in the original space and then apply the non-linear transformation implicitly using the kernel trick. This approach has some limitations, because the task information is applied before the non-linear transformation, and it is not clear how this task information is transformed in the kernel space. We will see in the next section how we can use an alternative kernel extension in which the task information, and the kernel that defines the similarity between features are decoupled.

## 4.2 Graph Laplacian Multi-Task Learning with Kernel Methods

Although the GL approach had already been proposed in Evgeniou et al. (2005), it is restricted to linear models, or to a kernel extension where the task information of the Laplacian is included before the kernel-related transformation. Here, we use the framework of tensor product of RKHSs to extend the GL formulation and consider to incorporate the Laplacian information in the Hilbert space, after the kernel-related transformation has been applied.

We will first develop the linear case of the SVM-based GL MTL formulation. After this, we will present a framework based on tensor products to extend the GL formulation to the kernel case.

### 4.2.1 Linear Case

We start from the simplest scenario, a linear approach, that is, given a $d$-dimensional input space $\mathcal{X}$, e.g. $\mathbb{R}^d$, we consider models of the form

$$h_r(\cdot) = \langle w_r, \cdot \rangle + b_r,\ w_r \in \mathbb{R}^d.$$

First, observe that we can express the Laplacian regularization as

$$\begin{aligned}\Omega(w_1, \ldots, w_T) &= \sum_{r=1}^{T}\sum_{s=1}^{T}(A)_{rs}\left\|w_r - w_s\right\|^2 \\ &= \sum_{r=1}^{T}\sum_{s=1}^{T}(A)_{rs}\left\{\|w_r\|^2 + \|w_s\|^2 - 2\langle w_r, w_s\rangle\right\}.\end{aligned}$$

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. $w_r = w$, the regularization for such model $w$ would be 0. This regularization can be combined with the individual model regularizers for each task. To illustrate this, we first consider a linear L1-SVM, whose primal problem, using the unified formulation for regression and classification, is

$$\begin{aligned}\underset{\boldsymbol{w},\boldsymbol{b},\boldsymbol{\xi}}{\arg\min}\quad & C\sum_{r=1}^{T}\sum_{i=1}^{m}\xi_i^r + \frac{\nu}{4}\sum_{r=1}^{T}\sum_{s=1}^{T}(A)_{rs}\left\|w_r - w_s\right\|^2 + \frac{1}{2}\sum_r\|w_r\|^2 \\ \text{s.t.}\quad & y_i^r(w_r \cdot x_i^r + b_r) \geq p_i^r - \xi_i^r,\ i = 1, \ldots, m_r;\ r = 1, \ldots, T, \\ & \xi_i^r \geq 0,\ i = 1, \ldots, m_r;\ r = 1, \ldots, T\ ,\end{aligned}$$

where we are using the vectors

$$\boldsymbol{w}^\intercal = (w_1^\intercal, \ldots, w_T^\intercal),\ \boldsymbol{b} = (b_1, \ldots, b_T),\ \boldsymbol{\xi} = (\xi_1^1, \ldots, \xi_{m_T}^T).$$

The corresponding Lagrangian is

$$\begin{aligned}\mathcal{L}(\boldsymbol{w}, \boldsymbol{b}, \boldsymbol{\xi}, \boldsymbol{\alpha}, \boldsymbol{\beta}) = C\sum_{r=1}^{T}\sum_{i=1}^{m_r}\xi_i^r + \frac{\nu}{2}\sum_{r=1}^{T}\sum_{s=1}^{T}(A)_{rs}\left\|w_r - w_s\right\|^2 + \frac{1}{2}\sum_r\|w_r\|^2 \\ - \sum_{r=1}^{T}\sum_{i=1}^{m_r}\alpha_i^r\left(y_i^r(w_r \cdot x_i^r + b_r) - p_i^r + \xi_i^r\right) - \sum_{r=1}^{T}\sum_{i=1}^{m_r}\beta_i^r\xi_i^r\ ,\end{aligned}$$

with $\boldsymbol{\alpha} = (\alpha_1^1, \ldots, \alpha_{m_T}^T)$ and $\boldsymbol{\beta} = (\beta_1^1, \ldots, \beta_{m_T}^T)$; then, taking the derivatives of the Lagrangian with respect to the primal variables and equating them to 0, we get

$$\frac{\partial\mathcal{L}}{\partial w_r} = 0 \implies w_r + \frac{\nu}{2}\sum_{s=1}^{T}((A)_{rs} + (A)_{sr})(w_r - w_s) = \sum_{i=1}^{m_r}\alpha_i^r y_i^r x_i^r\ , \tag{4.7}$$

$$\frac{\partial\mathcal{L}}{\partial b_r} = 0 \implies \sum_{i=1}^{m_r}\alpha_i^r y_i^r = 0\ , \tag{4.8}$$

$$\frac{\partial\mathcal{L}}{\partial \xi_i^r} = 0 \implies C_r - \alpha_i^r - \beta_i^r = 0\ . \tag{4.9}$$

Consider the following matrices:

$$\underset{T\times T}{L} = \begin{bmatrix} \sum_{s\neq 1}(A)_{1s} & -(A)_{12} & \dots & -(A)_{1T} \\ \vdots & \vdots & \ddots & \vdots \\ (A)_{T1} & (A)_{T2} & \dots & \sum_{s\neq T}(A)_{Ts} \end{bmatrix}, \quad \underset{T\times T}{E} = \{I_T + \nu L\}, \quad \underset{Td\times Td}{E_{\otimes}} = E \otimes I_d,$$

where $\otimes$ here is the Kronecker product of matrices, such that

$$\begin{bmatrix} E_{11} & E_{12} & \dots & E_{1T} \\ \vdots & \vdots & \ddots & \vdots \\ E_{T1} & E_{12} & \dots & E_{TT} \end{bmatrix} \otimes I_d = \begin{bmatrix} E_{11}I_d & E_{12}I_d & \dots & E_{1T}I_d \\ \vdots & \vdots & \ddots & \vdots \\ E_{T1}I_d & E_{12}I_d & \dots & E_{TT}I_d \end{bmatrix}.$$

We also define the matrix

$$\underset{(\sum_r m_r)\times Td}{\Phi} = \begin{bmatrix} Y_1X_1 & 0 & \dots & 0 \\ 0 & Y_2X_2 & \dots & 0 \\ \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & \dots & Y_TX_T \end{bmatrix},$$

where $X_r$ is the data matrix with the examples from task $r$ and

$$Y_r = \begin{bmatrix} y_1^r & 0 & \dots & 0 \\ 0 & y_2^r & \dots & 0 \\ \vdots & \vdots & \ddots & 0 \\ 0 & 0 & \dots & y_{m_r}^r \end{bmatrix}.$$

Then, we can write (4.7) in a matrix formulation as

$$E_{\otimes}\boldsymbol{w} = \Phi^{\intercal}\boldsymbol{\alpha} \implies \boldsymbol{w} = (E_{\otimes})^{-1}\Phi^{\intercal}\boldsymbol{\alpha}.$$

With these definitions and replacing (4.8) and (4.9) in the Lagrangian, the result is

$$\begin{aligned} \mathcal{L}(\boldsymbol{\alpha}) &= \frac{1}{2}\boldsymbol{w}^{\intercal}E_{\otimes}\boldsymbol{w} - \boldsymbol{\alpha}^{\intercal}\Phi\boldsymbol{w} + p^{\intercal}\boldsymbol{\alpha} \\ &= \frac{1}{2}((E_{\otimes})^{-1}\Phi^{\intercal}\boldsymbol{\alpha})^{\intercal}E_{\otimes}(E_{\otimes})^{-1}\Phi^{\intercal}\boldsymbol{\alpha} - \boldsymbol{\alpha}^{\intercal}\Phi(E_{\otimes})^{-1}\Phi^{\intercal}\boldsymbol{\alpha} + p^{\intercal}\boldsymbol{\alpha} \\ &= \frac{1}{2}\boldsymbol{\alpha}^{\intercal}\Phi(E_{\otimes}^{\intercal})^{-1}\Phi^{\intercal}\boldsymbol{\alpha} - \boldsymbol{\alpha}^{\intercal}\Phi(E_{\otimes})^{-1}\Phi^{\intercal}\boldsymbol{\alpha} + p^{\intercal}\boldsymbol{\alpha} \\ &= -\frac{1}{2}\boldsymbol{\alpha}^{\intercal}\Phi(E_{\otimes})^{-1}\Phi^{\intercal}\boldsymbol{\alpha} + p^{\intercal}\boldsymbol{\alpha}. \end{aligned}$$

Here, the block structure of $\Phi$ and $E_{\otimes}$ makes it possible to write $\Phi(E_{\otimes})^{-1}\Phi^{\intercal}$ as

$$\begin{bmatrix} (E^{-1})_{11}X_1Y_1Y_1X_1^{\intercal} & (E^{-1})_{12}X_1Y_1Y_2X_2^{\intercal} & \dots & (E^{-1})_{1T}X_1Y_1Y_TX_T^{\intercal} \\ (E^{-1})_{21}X_2Y_2Y_1X_1^{\intercal} & (E^{-1})_{22}X_2Y_2Y_2X_2^{\intercal} & \dots & (E^{-1})_{2T}X_2Y_2Y_TX_T^{\intercal} \\ \vdots & \vdots & \ddots & \vdots \\ (E^{-1})_{T1}X_TY_TY_1X_1^{\intercal} & (E^{-1})_{T2}X_TY_TY_2X_2^{\intercal} & \dots & (E^{-1})_{TT}X_TY_TY_TX_T^{\intercal} \end{bmatrix}. \tag{4.10}$$

Here, we note that $E = I_T + \nu L$ is invertible because it is a strictly diagonally dominant matrix. Observe that $L$ is diagonally dominant, because it is a Laplacian matrix, but not strictly so. However, with the addition of the identity matrix $I_T$, this property becomes

strict. The identity matrix has its origin in the individual regularization of the primal problem; thus, this additional regularization ultimately makes the solution of the problem more numerically stable.

Now we can define the kernel matrix $\widetilde{Q} = \Phi(E_{\otimes})^{-1}\Phi^{\intercal}$, such that its corresponding the kernel function is

$$\widetilde{k}(x_i^r, x_j^s) = \left((I_T + \nu L)^{-1}\right)_{rs} \left\langle x_i^r, x_j^s \right\rangle,$$

so the block $Q[rs]$, corresponding to the $r$-th and $s$-th task, is defined as

$$Q[rs]_{ij} = y_i^r y_j^s \widetilde{k}(x_i^r, x_j^s).$$

Using this kernel matrix, the corresponding dual problem can be expressed as

$$\begin{aligned} \arg\min_{\boldsymbol{\alpha}} \quad & \Theta(\boldsymbol{\alpha}) = \frac{1}{2}\boldsymbol{\alpha}^t \widetilde{Q}\boldsymbol{\alpha} - p\boldsymbol{\alpha} \\ \text{s.t.} \quad & 0 \leq \alpha_i^r \leq C,\ i = 1, \ldots, m_r; r = 1, \ldots, T, \\ & \sum_{i=1}^{n_r} \alpha_i^r y_i^r = 0,\ r = 1, \ldots, T. \end{aligned} \tag{4.11}$$

### 4.2.2 Kernel Extension

The kernel extension presented in Evgeniou et al. (2005) proposes using 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 allow to perform the, possibly infinite dimensional, mapping corresponding to a kernel and then incorporate the task information in the new space.

Here we propose another approach using tensor product of RKHSs and MTL kernels as given in Definition 4.3. Consider the RKHS $\mathcal{H}$ and the functional

$$R(u_1, \ldots, u_T) = \sum_{r=1}^{T} \sum_{i=1}^{m} \ell(y_i^r, \langle u_r, \phi(x_i^r) \rangle) + \mu \sum_r \sum_s (E)_{rs} \langle u_r, u_s \rangle, \tag{4.12}$$

where $\phi$ is a feature transformation, $u_1, \ldots, u_T \in \mathcal{H}$, and $E$ is a $T \times T$ symmetric, positive definite matrix. Recall that for the linear case of the GL MTL formulation, shown above, we have used the vector

$$\boldsymbol{w} = \begin{bmatrix} w_1 \\ \vdots \\ w_T \end{bmatrix} = \sum_{t=1}^{T} e_r \otimes w_r,$$

where $w_r \in \mathbb{R}^d$ and $\{e_1, \ldots, e_T\}$ is an orthonormal basis of $\mathbb{R}^T$. To use a similar approach in the infinite-dimensional scenario, we define

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

where we are using the tensor product as defined in the previous section, such that $\boldsymbol{u} \in \mathbb{R}^T \otimes \mathcal{H}$. Then, we can reformulate (4.12) with a tensor product formulation as the

problem

$$R(\boldsymbol{u}) = \sum_{r=1}^{T}\sum_{i=1}^{m} \ell(y_i^r, \langle \boldsymbol{u}, e_r \otimes \phi(x_i^r)\rangle) + \mu\left(\boldsymbol{u}^\intercal (E \otimes I)\boldsymbol{u}\right). \tag{4.13}$$

The following lemma illustrates how to minimize this functional as a single task problem.

**Lemma 4.6.** *The solutions $u_1^*, \ldots, u_T^*$, from the MTL problem (4.12), and equivalently the solution $\boldsymbol{u}$ from the problem (4.13), can be obtained solving the problem*

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

*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), 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 $\operatorname{rank} B = T$ such that $E^{-1} = B^\intercal B$, using for example the SVD. Then, with the properties of the tensor product of linear maps,

$$E^{-1} \otimes I = (B^\intercal B) \otimes I = (B^\intercal \otimes I)(B \otimes I).$$

Consider the change of variable $\boldsymbol{u} = (B^\intercal \otimes I)\boldsymbol{w}$, where $\boldsymbol{w} \in \mathbb{R}^p \otimes \mathcal{H}$.

Rewriting (4.13) using $\boldsymbol{w}$, we obtain

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

which is equivalent to problem (4.13).

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:

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

where $k(\cdot,\cdot)$ is the reproducing kernel of the space of functions $\phi(\cdot)$. However, in the minimization problem we only have values $z = Be_r = B_r$ for some $r = 1, \ldots, T$; then $\langle B_r, B_s\rangle = \left(E^{-1}\right)_{rs}$. 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

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

where we define vect $(a_1, \ldots, a_T)$ as

$$\operatorname{vect}(a_1, \ldots, a_T) = \begin{bmatrix} a_1 \\ \vdots \\ a_T \end{bmatrix}.$$

Then, we can recover the predictions corresponding to the solutions $u_r^*$ as

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

□

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

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

That is, we have expressed the MTL problem as a STL problem with the MTL kernel being

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

as defined in Definition 4.3. Note that the kernels obtained in this way, unlike those of Lemma 4.5, split the inter-task relations and the similarity between data points. That is, we can implicitly send our data into another, 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.

Now we can use Lemma 4.6 to solve the kernelized GL MTL problem

$$\begin{aligned}
\underset{\boldsymbol{w},\boldsymbol{b},\boldsymbol{\xi}}{\arg\min} \quad & C\sum_{r=1}^{T}\sum_{i=1}^{m} \xi_i^r + \frac{\nu}{4}\sum_{r=1}^{T}\sum_{s=1}^{T}(A)_{rs}\|w_r - w_s\|^2 + \frac{1}{2}\sum_r \|w_r\|^2 \\
\text{s.t.} \quad & y_i^r(\langle w_r, \phi(x_i^r)\rangle + b_r) \geq p_i^r - \xi_i^r,\ i = 1, \ldots, m_r;\ r = 1, \ldots, T, \\
& \xi_i^r \geq 0,\ i = 1, \ldots, m_r;\ r = 1, \ldots, T\,.
\end{aligned} \tag{4.14}$$

When we use an implicit kernel transformation, the result (4.10) is not direct. However, we can take the following approach. First, we define $\boldsymbol{v}$ as

$$\boldsymbol{v} = \sum_{t=1}^{T} e_r \otimes v_r,$$

where $\{e_1, \ldots, e_T\}$ is the canonical basis of $\mathbb{R}^T$. Then, we can observe that

$$\begin{aligned}
\boldsymbol{v}^\intercal (I_T \otimes I_d)\boldsymbol{v} &= \sum_{r=1}^{T} \|v_r\|^2, \\
\boldsymbol{v}^\intercal (L \otimes I_d)\boldsymbol{v} &= \frac{1}{2}\sum_{r=1}^{T}\sum_{s=1}^{T} (A)_{rs} \|v_r - v_s\|^2.
\end{aligned}$$

To check the second equation, see

$$\begin{aligned}
\boldsymbol{v}^\intercal (L \otimes I_d)\boldsymbol{v} &= \boldsymbol{v}^\intercal (D \otimes I_d)\boldsymbol{v} - \boldsymbol{v}^\intercal (A \otimes I_d)\boldsymbol{v} \\
&= \sum_{r=1}^{T}\sum_{s=1}^{T} D_{rs} v_r^\intercal v_s - \sum_{r=1}^{T}\sum_{s=1}^{T} (A)_{rs} v_r^\intercal v_s \\
&= \sum_{r=1}^{T} D_{rr} v_r^\intercal v_r - \sum_{r=1}^{T}\sum_{s=1}^{T} (A)_{rs} v_r^\intercal v_s \\
&= \sum_{r=1}^{T}\sum_{s=1}^{T} (A)_{rs} v_r^\intercal v_r - \sum_{r=1}^{T}\sum_{s=1}^{T} (A)_{rs} v_r^\intercal v_s \\
&= \sum_{r=1}^{T}\sum_{s=1}^{T} (A)_{rs} (v_r^\intercal v_r - v_r^\intercal v_s),
\end{aligned}$$

which can be expressed as

$$\begin{aligned}
&\sum_{r=1}^{T}\sum_{s=r}^{T} \{(A)_{rs}(v_r^\intercal v_r - v_r^\intercal v_s) + (A)_{sr}(v_s^\intercal v_s - v_s^\intercal v_r)\} \\
&= \sum_{r=1}^{T}\sum_{s=r}^{T} \{((A)_{rs} + (A)_{sr})(v_r^\intercal v_r + v_s^\intercal v_s - 2 v_r^\intercal v_s)\} \\
&= \sum_{r=1}^{T}\sum_{s=r}^{T} \{((A)_{rs} + (A)_{sr}) \|v_r - v_s\|^2\} \\
&= \sum_{r=1}^{T}\sum_{s=1}^{T} \{(A)_{rs} \|v_r - v_s\|^2\}.
\end{aligned}$$

Therefore, we can express the kernelized problem (4.14) as

$$R(\boldsymbol{v}) = C \sum_{r=1}^{T}\sum_{i=1}^{m} \ell(y_i^r, \langle \boldsymbol{v}, e_r \otimes \phi(x_i^r)\rangle) + (\boldsymbol{v}^\intercal ((\nu L + I_T) \otimes I)\boldsymbol{v}),$$

where $\ell$ is the hinge loss. Note now that this is the optimization problem (4.13) with the matrix $E = (\nu L + I_T)$; then, as shown in Lemma 4.6, this is equivalent to solving a dual

problem where the kernel function is

$$\widetilde{k}(x_i^r, x_j^s) = \left((\nu L + I_T)^{-1}\right)_{rs} k(x_i^r, x_j^s), \tag{4.15}$$

where $k(\cdot,\cdot)$ is the reproducing kernel induced by the implicit transformation $\phi(\cdot)$.

Thus, the dual problem corresponding to problem (4.14) is the problem presented in (4.11), but where the kernel matrix $\widetilde{Q}$ is now defined using the kernel function (4.15).

## 4.3 Convex Graph Laplacian Multi-Task Learning

In Ruiz et al. (2020a) we proposed a convex formulation for 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

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

where $\phi(\cdot)$ and $\psi(\cdot)$ are the implicit transformations for the common and task-specific parts, and can be possibly distinct to try 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 (3.1), where each task-specific part can use a different Hilbert space, here all tasks must use the same two transformations: $\phi$ and $\psi$. 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 RKHS. Again, as with the convex MTL approach, the hyperparameters $\lambda_r \in [0,1]$ define how relevant is 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. Therefore, the $\lambda_r$ values are hyperparameters that must be selected for each specific problem.

### 4.3.1 General Result for Kernel Methods

In general, we can apply Lemma 4.6 to find the solution of kernel methods problems that use an GL regularization. Here, for the convex formulation, we follow an analogous approach as the one used above; with $e_1, \ldots, e_T$ being the canonical basis of $\mathbb{R}^T$, we define

$$\boldsymbol{v} = \sum_{t=1}^{T} e_r \otimes v_r \in \mathbb{R}^T \otimes \mathcal{H}_\psi, \tag{4.16}$$

such that $\langle \boldsymbol{v}, e_r \otimes \psi(x_i^r)\rangle = \langle v_r, \psi(x_i^r)\rangle$. Consider then the product space of the RKHS $\mathcal{H}_\phi$ and the tensor product of spaces $(\mathbb{R}^T \otimes \mathcal{H}_\psi)$: $\mathcal{H}_\phi \times (\mathbb{R}^T \otimes \mathcal{H}_\psi)$. In this space, the norm of $(w, \boldsymbol{v}) \in \mathcal{H}_\phi \times (\mathbb{R}^T \otimes \mathcal{H}_\psi)$ 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 GL MTL can be expressed as

$$\begin{aligned}\underset{(w,\boldsymbol{v})\in\mathcal{H}_\phi\times(\mathbb{R}^T\otimes\mathcal{H}_\psi)}{\arg\min} R((w,\boldsymbol{v})) = &\sum_{r=1}^{T}\sum_{i=1}^{m_r}\ell(y_i^r,(\langle(w,\boldsymbol{v}),(\lambda_r\phi(x_i^r),(1-\lambda_r)(e_r\otimes\psi(x_i^r)))\rangle))\\ &+\left\langle(w,\boldsymbol{v}),(I_{\mathcal{H}_\phi}\times(E\otimes I_{\mathcal{H}_\psi}))(w,\boldsymbol{v})\right\rangle,\end{aligned} \tag{4.17}$$

where $\ell$ is a loss function, $I_{\mathcal{H}_\phi}$ in our case is the identity operator in $\mathcal{H}_\phi$, and $E$ is a symmetric, positive definite operator in $\mathbb{R}^T$, which for our GL formulation is

$$E=(\nu L+I),$$

with $L$ being a GL matrix. Using a modification of Lemma 4.6, we can state the following result.

**Lemma 4.7.** *The solution $(w^*,\boldsymbol{v}^*)$ from the Multi-Task optimization problem (4.17) can be obtained solving the minimization problem*

$$\begin{aligned}\underset{(w,\boldsymbol{u})\in\mathcal{H}_\phi\times(\mathbb{R}^T\otimes\mathcal{H}_\psi)}{\arg\min} S((w,\boldsymbol{u})) = &\sum_{r=1}^{T}\sum_{i=1}^{m_r}\ell(y_i^r,\langle(w,\boldsymbol{u}),(\lambda_r\phi(x_i^r),(1-\lambda_r)(B_r\otimes\psi(x_i^r)))\rangle)\\ &+\mu\left(\langle w,w\rangle+\langle\boldsymbol{u},\boldsymbol{u}\rangle\right),\end{aligned} \tag{4.18}$$

*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 $E^{-1}=B^\intercal B$.*

*Proof.* 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 rank $B=T$ such that $E^{-1}=B^\intercal B$, using for example the SVD; then, we can express the inverse of the operator $E\otimes I_{\mathcal{H}_\psi}$ from (4.17) as

$$\left(E\otimes I_{\mathcal{H}_\psi}\right)^{-1}=E^{-1}\otimes I_{\mathcal{H}_\psi}=(B^\intercal B)\otimes I_{\mathcal{H}_\psi}=(B^\intercal\otimes I_{\mathcal{H}_\psi})(B\otimes I_{\mathcal{H}_\psi}).$$

Consider the change of variable $\boldsymbol{v}=(B^\intercal\otimes I_{\mathcal{H}_\psi})\boldsymbol{u}$, where $\boldsymbol{u}\in\mathbb{R}^p\otimes\mathcal{H}$. Then we can write

$$\begin{aligned}R(w,\boldsymbol{u}) = &\sum_{r=1}^{T}\sum_{i=1}^{m_r}\ell(y_i^r,\left\langle(w,(B^\intercal\otimes I_{\mathcal{H}_\psi})\boldsymbol{u}),(\lambda_r\phi(x_i^r),(1-\lambda_r)(e_r\otimes\psi(x_i^r)))\right\rangle)\\ &+\mu\left\langle(w,(B^\intercal\otimes I_{\mathcal{H}_\psi})\boldsymbol{u}),(I_{\mathcal{H}_\phi}\times(E\otimes I_{\mathcal{H}_\psi}))(w,(B^\intercal\otimes I_{\mathcal{H}_\psi})\boldsymbol{u})\right\rangle.\end{aligned}$$

This is equivalent to problem (4.18), where the regularizer is increasing in $\|(w,\boldsymbol{u})\|^2$, so we can apply the representer theorem (Schölkopf et al., 2001), which states that the minimizer $\omega^*$ of any empirical regularized risk

$$\sum_{i=1}^{n}\ell(\langle\omega,\phi(x_i)\rangle,y_i)+\Omega(\|f\|),$$

where $\Omega$ is a strictly increasing function, admits a representation of the form $\omega^*=\sum_{i=1}^n\alpha_i\phi(x_i)$. Thus, the minimizer of $S(w,\boldsymbol{u})$ in (4.18) has the form

$$(w^*,\boldsymbol{u}^*)=\sum_{r=1}^{T}\sum_{i=1}^{m_r}\alpha_i^r(\lambda_r\phi(x_i^r),(1-\lambda_r)(B_r\otimes\psi(x_i^r))).$$

Applying the correspondence between $\boldsymbol{u}^*$ and $\boldsymbol{v}^*$, namely, $\boldsymbol{v}^* = (B^\intercal \otimes I_{\mathcal{H}_\psi})\boldsymbol{u}^*$, we get

$$\begin{aligned}(w^*, \boldsymbol{v}^*) &= (w^*, (B^\intercal \otimes I_{\mathcal{H}_\psi})\boldsymbol{u}^*) \\ &= \sum_{r=1}^{T}\sum_{i=1}^{m_r} \alpha_i^r(\lambda_r\phi(x_i^r), (1-\lambda_r)\text{vect}\left(\langle B_1, B_r\rangle, \ldots, \langle B_T, B_r\rangle\right) \otimes \psi(x_i^r)) \\ &= \sum_{r=1}^{T}\sum_{i=1}^{m_r} \alpha_i^r(\lambda_r\phi(x_i^r), (1-\lambda_r)\text{vect}\left((E^{-1})_{1r}, \ldots, (E^{-1})_{Tr}\right) \otimes \psi(x_i^r)),\end{aligned} \tag{4.19}$$

where $\text{vect}\,(a_1, \ldots, a_l)$ is the vector whose elements are the scalars $a_1, \ldots, a_l$. Now, we can recover the predictions corresponding to the solutions $(w^*, \boldsymbol{v}^*)$ as

$$\begin{aligned}&\left\langle (w^*, \boldsymbol{v}^*), \left(\lambda_t\phi(\hat{x}^t), (1-\lambda_t)e_t \otimes \psi(\hat{x}^t)\right)\right\rangle \\ &= \Bigg\langle \sum_{r=1}^{T}\sum_{i=1}^{m_r} \alpha_i^r(\lambda_r\phi(x_i^r), (1-\lambda_r)\text{vect}\left((E^{-1})_{1r}, \ldots, (E^{-1})_{Tr}\right) \otimes \phi(x_i^r)), \\ &\qquad \left(\lambda_t\phi(\hat{x}^t), (1-\lambda_t)e_t \otimes \psi(\hat{x}^t)\right)\Bigg\rangle \\ &= \sum_{r=1}^{T}\sum_{i=1}^{m_r} \alpha_i^r(\lambda_r\lambda_t \left\langle\phi(x_i^r), \phi(x_i^r)\right\rangle + (1-\lambda_r)(1-\lambda_t)(E^{-1})_{rs}\left\langle\psi(x_i^r), \psi(\hat{x}^t)\right\rangle).\end{aligned}$$

□

As with Lemma 4.6, this is a result that gives an easy way of finding the solutions because it shows that the MTL problem (4.17) can be expressed as a CTL problem with the MTL kernel function

$$\bar{k}(x_i^r, x_j^s) = \lambda_r\lambda_s k_\phi(x_i^r, x_j^s) + (1-\lambda_r)(1-\lambda_s)\left((\nu L + I_T)^{-1}\right)_{rs} k_\psi(x_i^r, x_j^s), \tag{4.20}$$

where $k_\phi(\cdot,\cdot)$ and $k_\psi(\cdot,\cdot)$ are the reproducing kernels corresponding to the transformations $\phi$ and $\psi$, respectively. Thus, we can use standard CTL kernel methods with a modified kernel to solve GL MTL problems.

We derive next the Convex GL MTL formulations for the L1, L2 and LS-SVMs by applying this result.

### 4.3.2 Convex Graph Laplacian L1-SVM

The primal problem for the linear L1-SVM with the convex GL approach, that we have presented in (Ruiz et al., 2020a), is the following one

$$\begin{aligned}\underset{\substack{v_1,\ldots,v_T;\\ b_1,\ldots,b_T;\\ \boldsymbol{\xi},w;}}{\arg\min}\quad & C\sum_{r=1}^{T}\sum_{i=1}^{m_r}\xi_i^r + \frac{\nu}{2}\sum_{r=1}^{T}\sum_{s=1}^{T}(A)_{rs}\|v_r - v_s\|^2 + \frac{1}{2}\sum_r\|v_r\|^2 + \frac{1}{2}\|w\|^2 \\ \text{s.t.}\quad & y_i^r(\lambda_r(w\cdot x_i^r) + (1-\lambda_r)(v_r\cdot x_i^r) + b_r) \geq p_i^r - \xi_i^r, \\ & \xi_i^r \geq 0,\ i = 1,\ldots,m_r,\ r = 1,\ldots,T.\end{aligned} \tag{4.21}$$

Note that with $\nu = 0$, this problem is equivalent to our proposed convex MTL formulation shown in (3.8). Recall here that we are using the unifying formulation, such that, for every task $r = 1, \ldots, T$, when $p_i^r = 1$ for $i = 1, \ldots, m_r$, it is equivalent to the SVM for

classification; but if we double the number of instances and select $y_i^r = 1$ for $i = 1, \ldots, m_r$, $y_i = -1$ for $i = m_r + 1, \ldots, 2m_r$, and $p_i^r$ as the target values, we get the SVM problem for regression. The extension to the kernel case requires using a different formulation of (4.21), that is

$$\begin{aligned} \underset{\substack{\boldsymbol{v};\\ b_1,\ldots,b_T;\\ \boldsymbol{\xi},w;}}{\arg\min} \quad & C\sum_{r=1}^{T}\sum_{i=1}^{m_r}\xi_i^r + \frac{\nu}{2}\boldsymbol{v}^\intercal(L\otimes I_{\mathcal{H}})\boldsymbol{v} + \frac{1}{2}\boldsymbol{v}^\intercal(I_T\otimes I_{\mathcal{H}})\boldsymbol{v} + \frac{1}{2}\|w\|^2 \\ \text{s.t.} \quad & y_i^r(\lambda_r\langle w, \phi(x_i^r)\rangle + (1-\lambda_r)(\langle \boldsymbol{v}, e_r\otimes\psi(x_i^r)\rangle) + b_r) \geq p_i^r - \xi_i^r, \\ & \xi_i^r \geq 0,\ i = 1,\ldots,m_r,\ r = 1,\ldots,T. \end{aligned} \tag{4.22}$$

Although the result from Lemma 4.7 can be applied for this problem, for illustration purposes we will develop the entire procedure for this L1-SVM case. The Lagrangian corresponding to (4.22) is

$$\begin{aligned} &\mathcal{L}(w, \boldsymbol{v}, \boldsymbol{b}, \boldsymbol{\xi}, \boldsymbol{\alpha}, \boldsymbol{\beta}) \\ &= C\sum_{r=1}^{T}\sum_{i=1}^{m_r}\xi_i^r + \frac{\nu}{2}\boldsymbol{v}^\intercal(L\otimes I_{\mathcal{H}})\boldsymbol{v} + \frac{1}{2}\boldsymbol{v}^\intercal(I_T\otimes I_{\mathcal{H}})\boldsymbol{v} + \frac{1}{2}\|w\|^2 \\ &\quad - \sum_{r=1}^{T}\sum_{i=1}^{m_r}\alpha_i^r[y_i^r(\lambda_r\langle w, \phi(x_i^r)\rangle + (1-\lambda_r)(\langle \boldsymbol{v}, e_r\otimes\psi(x_i^r)\rangle) + b_r) - p_i^r + \xi_i^r] \\ &\quad - \sum_{r=1}^{T}\sum_{i=1}^{m_r}\beta_i^r\xi_i^r, \end{aligned}$$

where $\alpha_i^r, \beta_i^r \geq 0$. Taking derivatives and making them 0, we get

$$\begin{aligned} &\nabla_w\mathcal{L} = 0 \implies w^* = \sum_{r=1}^{T}\lambda_r\sum_{i=1}^{m_r}\alpha_i^r\left\{y_i^r\phi(x_i^r)\right\}, \\ &\nabla_{\boldsymbol{v}}\mathcal{L} = 0 \implies ((I_T + \nu L)\otimes I_{\mathcal{H}})\,\boldsymbol{v} = \sum_{r=1}^{T}(1-\lambda_r)\sum_{i=1}^{m_r}\alpha_i^r y_i^r(e_r\otimes\psi(x_i^r)), \\ &\nabla_{b_r}\mathcal{L} = 0 \implies \sum_{i=1}^{m_r}\alpha_i^r y_i^r = 0, \\ &\nabla_{\xi_i^r}\mathcal{L} = 0 \implies C - \alpha_i^r - \beta_i^r = 0. \end{aligned} \tag{4.23}$$

Here we have

$$E = (I_T + \nu L),\ E_{\otimes} = (E\otimes I_{\mathcal{H}})\,. \tag{4.24}$$

Substituting these results in the Lagrangian, we get

$$\begin{aligned} &\mathcal{L}(w, \boldsymbol{v}, \boldsymbol{b}, \boldsymbol{\xi}, \boldsymbol{\alpha}, \boldsymbol{\beta}) \\ &= -\frac{1}{2}\sum_{r,s=1}^{T}(1-\lambda_r)(1-\lambda_s)\sum_{i,j=1}^{m_r,m_s}\alpha_j^s\alpha_i^r y_j^s y_i^r\left\langle(e_s\otimes\psi(x_j^s)), (E_{\otimes})^{-1}(e_r\otimes\psi(x_i^r))\right\rangle \\ &\quad - \frac{1}{2}\sum_{r,s=1}^{T}\lambda_r\lambda_s\sum_{i,j=1}^{m_r,m_s}\alpha_j^s\alpha_i^r y_j^s y_i^r\left\langle\phi(x_j^s), \phi(x_i^r)\right\rangle - \sum_{r=1}^{T}\sum_{i=1}^{m_r}\alpha_i^r p_i^r. \end{aligned}$$

Due to (4.23) 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

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

Here, we use the matrix

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

$I_n$ 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 GL information. We can define now the kernel matrix

$$\bar{Q} = \Lambda Q \Lambda + (I_n - \Lambda)\,\widetilde{Q}\,(I_n - \Lambda)\,, \tag{4.27}$$

as the matrix that is generated using the kernel function (4.20). That is, in each block $\bar{Q}[rs]$, corresponding to the $r$-th and $s$-th tasks, we have that

$$\bar{Q}[rs]_{ij} = y_i^r y_j^s \bar{k}(x_i^r, x_j^s),$$

so the entire matrix $\bar{Q}$ is defined as

$$\bar{Q} = \begin{bmatrix} \bar{Q}[11] & \bar{Q}[12] & \ldots & \bar{Q}[1T] \\ \vdots & \vdots & \ddots & \vdots \\ \bar{Q}[T1] & \bar{Q}[T2] & \ldots & \bar{Q}[TT] \end{bmatrix}.$$

### 4.3.3 Convex Graph Laplacian L2-SVM

The primal problem for convex GL MTL based on the linear L2-SVM, with the unifying formulation for regression and classification, is

$$\begin{aligned} \underset{\substack{v_1,\ldots,v_T;\\ b_1,\ldots,b_T;\\ \boldsymbol{\xi},w;}}{\arg\min} \quad & C\sum_{r=1}^{T}\sum_{i=1}^{m_r}(\xi_i^r)^2 + \frac{\nu}{2}\sum_{r=1}^{T}\sum_{s=1}^{T}(A)_{rs}\|v_r - v_s\|^2 + \frac{1}{2}\sum_r \|v_r\|^2 + \frac{1}{2}\|w\|^2 \\ \text{s.t.} \quad & y_i^r(\lambda_r(w \cdot x_i^r) + (1-\lambda_r)(v_r \cdot x_i^r) + b_r) \geq p_i^r - \xi_i^r; \end{aligned} \tag{4.28}$$

and with the tensor product formulation we can express the kernelized version of problem (4.28) as

$$\begin{aligned} \underset{\substack{\boldsymbol{v};\\ b_1,\ldots,b_T;\\ \boldsymbol{\xi},w;}}{\arg\min} \quad & C\sum_{r=1}^{T}\sum_{i=1}^{m_r}(\xi_i^r)^2 + \frac{\nu}{2}\boldsymbol{v}^{\intercal}(L \otimes I_{\mathcal{H}})\boldsymbol{v} + \frac{1}{2}\boldsymbol{v}^{\intercal}(I_T \otimes I_{\mathcal{H}})\boldsymbol{v} + \frac{1}{2}\|w\|^2 \\ \text{s.t.} \quad & y_i^r(\lambda_r \langle w, \phi(x_i^r)\rangle + (1-\lambda_r)(\langle \boldsymbol{v}, e_r \otimes \psi(x_i^r)\rangle) + b_r) \geq p_i^r - \xi_i^r. \end{aligned} \tag{4.29}$$

The corresponding Lagrangian it then the following one:

$$\begin{aligned}
&\mathcal{L}(w, \boldsymbol{v}, \boldsymbol{b}, \boldsymbol{\xi}, \boldsymbol{\alpha}) \\
&= C \sum_{r=1}^{T} \sum_{i=1}^{m_r} (\xi_i^r)^2 + \frac{\nu}{2} \boldsymbol{v}^\intercal (L \otimes I_{\mathcal{H}}) \boldsymbol{v} + \frac{1}{2} \boldsymbol{v}^\intercal (I_T \otimes I_{\mathcal{H}}) \boldsymbol{v} + \frac{1}{2} \|w\|^2 \\
&\quad - \sum_{r=1}^{T} \sum_{i=1}^{m_r} \alpha_i^r [y_i^r (\lambda_r \langle w, \phi(x_i^r) \rangle + (1 - \lambda_r)(\langle \boldsymbol{v}, e_r \otimes \psi(x_i^r) \rangle) + b_r) - p_i^r + \xi_i^r],
\end{aligned}$$

where $\alpha_i^r \geq 0$ are the Lagrange multipliers. Computing the gradients with respect to the primal variables and making them 0, we obtain

$$\nabla_w \mathcal{L} = 0 \implies w = \sum_{r=1}^{T} \lambda_r \sum_{i=1}^{m_r} \alpha_i^r \left\{ y_i^r \phi(x_i^r) \right\}, \tag{4.30}$$

$$\nabla_{\boldsymbol{v}} \mathcal{L} = 0 \implies \left( (I_T + \nu L) \otimes I_{\mathcal{H}} \right) \boldsymbol{v} = \sum_{r=1}^{T} (1 - \lambda_r) \sum_{i=1}^{m_r} \alpha_i^r y_i^r (e_r \otimes \psi(x_i^r)), \tag{4.31}$$

$$\nabla_{b_r} \mathcal{L} = 0 \implies \sum_{i=1}^{m_r} \alpha_i^r y_i^r = 0, \tag{4.32}$$

$$\nabla_{\xi_i^r} \mathcal{L} = 0 \implies C \xi_i^r - \alpha_i^r = 0. \tag{4.33}$$

With $E_{\otimes}$ as defined in (4.24), applying (4.30), (4.31) to substitute $w$ and $\boldsymbol{v}$ and (4.33) to replace $\xi_i^r$ in the Lagrangian, we get

$$\begin{aligned}
&\mathcal{L}(w, \boldsymbol{v}, \boldsymbol{b}, \boldsymbol{\xi}, \boldsymbol{\alpha}, \boldsymbol{\beta}) \\
&= \frac{1}{2C} \sum_{r=1}^{T} \sum_{i=1}^{m_r} (\alpha_i^r)^2 - \frac{1}{C} \sum_{r=1}^{T} \sum_{i=1}^{m_r} (\alpha_i^r)^2 + \frac{1}{2} \langle \boldsymbol{v}, E_{\otimes} \boldsymbol{v} \rangle + \frac{1}{2} \langle w, w \rangle \\
&\quad - \sum_{r=1}^{T} \sum_{i=1}^{m_r} \alpha_i^r [y_i^r (\lambda_r \langle w, \phi(x_i^r) \rangle + (1 - \lambda_r)(\langle \boldsymbol{v}, e_r \otimes \psi(x_i^r) \rangle) + b_r) - p_i^r],
\end{aligned}$$

which is equal to

$$\begin{aligned}
&\frac{1}{2C} \sum_{r=1}^{T} \sum_{i=1}^{m_r} (\alpha_i^r)^2 - \frac{1}{C} \sum_{r=1}^{T} \sum_{i=1}^{m_r} (\alpha_i^r)^2 \\
&\quad - \frac{1}{2} \sum_{r,s=1}^{T} (1 - \lambda_r)(1 - \lambda_s) \sum_{i,j=1}^{m_r, m_s} \alpha_j^s \alpha_i^r y_j^s y_i^r \left\langle (e_s \otimes \psi(x_j^s)), (E_{\otimes})^{-1} (e_r \otimes \psi(x_i^r)) \right\rangle \\
&\quad - \frac{1}{2} \sum_{r,s=1}^{T} \lambda_r \lambda_s \sum_{i,j=1}^{m_r, m_s} \alpha_j^s \alpha_i^r y_j^s y_i^r \left\langle \phi(x_j^s), \phi(x_i^r) \right\rangle - \sum_{r=1}^{T} \sum_{i=1}^{m_r} \alpha_i^r p_i^r.
\end{aligned}$$

Thus, alongside (4.32), the dual problem for the L2-SVM based formulation is

$$\begin{aligned} \underset{\boldsymbol{\alpha}}{\arg\min} \quad & \Theta(\boldsymbol{\alpha}) = \frac{1}{2}\boldsymbol{\alpha}^t \left\{ \left( \Lambda Q \Lambda + (I_n - \Lambda)\, \widetilde{Q}\, (I_n - \Lambda) \right) + \frac{1}{C} I_n \right\} \boldsymbol{\alpha} - \boldsymbol{p}\boldsymbol{\alpha} \\ \text{s.t.} \quad & 0 \leq \alpha_i^r,\ i = 1, \ldots, m_r; r = 1, \ldots, T, \\ & \sum_{i=1}^{n_r} \alpha_i^r y_i^r = 0,\ r = 1, \ldots, T. \end{aligned} \tag{4.34}$$

Here, again we use the matrix $\Lambda$ defined in (4.26). Now, we have the same differences that we find in the standard L1 and L2-SVM; there is no upper bound for $\alpha_i^r$, but an additional diagonal term, the identity matrix $\frac{1}{C} I_n$, appears, which can be interpreted as a soft constraint for the dual coefficients $\alpha_i^r$. Again, we end up with the kernel matrix $\bar{Q}$ as defined in (4.27), which combines the common matrix $Q$ and the matrix $\widetilde{Q}$ with the GL information.

### 4.3.4 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 GL approach, based on the linear LS-SVM is the following one:

$$\begin{aligned} \underset{\substack{v_1,\ldots,v_T;\\ b_1,\ldots,b_T;\\ \boldsymbol{\xi},w;}}{\arg\min} \quad & C \sum_{r=1}^{T} \sum_{i=1}^{m_r} (\xi_i^r)^2 + \frac{\nu}{2} \sum_{r=1}^{T} \sum_{s=1}^{T} (A)_{rs} \|v_r - v_s\|^2 + \frac{1}{2} \sum_r \|v_r\|^2 + \frac{1}{2} \|w\|^2 \\ \text{s.t.} \quad & y_i^r (\lambda_r (w \cdot x_i^r) + (1 - \lambda_r)(v_r \cdot x_i^r) + b_r) \geq p_i^r - \xi_i^r. \end{aligned} \tag{4.35}$$

Here, we are using the unifying formulation for the LS-SVM. Problem (4.35) is equivalent to the classification problem when $p_i^r = 1$ and $y_i^r \in \{-1, 1\}$ are the class labels ; and it is equivalent to the regression problem when $y_i^r = 1$ and $p_i^r$ are the target values, for all $i = 1, \ldots, m_r$ and $r = 1, \ldots, T$. We can extend this problem to the non-linear case using the tensor product formulation, which is then expressed as:

$$\begin{aligned} \underset{\substack{\boldsymbol{v};\\ b_1,\ldots,b_T;\\ \boldsymbol{\xi},w;}}{\arg\min} \quad & C \sum_{r=1}^{T} \sum_{i=1}^{m_r} (\xi_i^r)^2 + \frac{\nu}{2} \boldsymbol{v}^\intercal (L \otimes I_{\mathcal{H}}) \boldsymbol{v} + \frac{1}{2} \boldsymbol{v}^\intercal (I_T \otimes I_{\mathcal{H}}) \boldsymbol{v} + \frac{1}{2} \|w\|^2 \\ \text{s.t.} \quad & y_i^r (\lambda_r \langle w, \phi(x_i^r) \rangle + (1 - \lambda_r)(\langle \boldsymbol{v}, e_r \otimes \psi(x_i^r) \rangle) + b_r) = p_i^r - \xi_i^r. \end{aligned} \tag{4.36}$$

The Lagrangian corresponding to this optimization problem is

$$\begin{aligned} & \mathcal{L}(w, \boldsymbol{v}, \boldsymbol{b}, \boldsymbol{\xi}, \boldsymbol{\alpha}) \\ & = C \sum_{r=1}^{T} \sum_{i=1}^{m_r} (\xi_i^r)^2 + \frac{\nu}{2} \boldsymbol{v}^\intercal (L \otimes I_{\mathcal{H}}) \boldsymbol{v} + \frac{1}{2} \boldsymbol{v}^\intercal (I_T \otimes I_{\mathcal{H}}) \boldsymbol{v} + \frac{1}{2} \|w\|^2 \\ & \quad - \sum_{r=1}^{T} \sum_{i=1}^{m_r} \alpha_i^r [y_i^r (\lambda_r \langle w, \phi(x_i^r) \rangle + (1 - \lambda_r)(\langle \boldsymbol{v}, e_r \otimes \psi(x_i^r) \rangle) + b_r) - p_i^r + \xi_i^r], \end{aligned}$$

where now, unlike in the L1 and L2-SVM cases, there are no restrictions for the Lagrange multipliers $\alpha_i^r$. The KKT conditions for this problem are then

$$\nabla_w \mathcal{L} = 0 \implies w^* = \sum_{r=1}^{T} \lambda_r \sum_{i=1}^{m_r} \alpha_i^r \left\{ y_i^r \phi(x_i^r) \right\}, \tag{4.37}$$

$$\nabla_{\boldsymbol{v}} \mathcal{L} = 0 \implies \left( (I_T + \nu L) \otimes I_{\mathcal{H}} \right) \boldsymbol{v} = \sum_{r=1}^{T} (1 - \lambda_r) \sum_{i=1}^{m_r} \alpha_i^r y_i^r (e_r \otimes \psi(x_i^r)), \tag{4.38}$$

$$\nabla_{b_r} \mathcal{L} = 0 \implies \sum_{i=1}^{m_r} \alpha_i^r y_i^r = 0, \tag{4.39}$$

$$\nabla_{\xi_i^r} \mathcal{L} = 0 \implies C\xi_i^r - \alpha_i^r = 0, \tag{4.40}$$

$$\nabla_{\alpha_i^r} \mathcal{L} = 0 \implies y_i^r (\lambda_r \langle w, \phi(x_i^r) \rangle + (1 - \lambda_r)(\langle \boldsymbol{v}, e_r \otimes \psi(x_i^r) \rangle) + b_r) + \xi_i^r = p_i^r. \tag{4.41}$$

Applying (4.37), (4.38) and (4.40) to replace $w, \boldsymbol{v}$ and $\xi_i^r$ in (4.41), with $E_\otimes$ as defined in (4.24), we get

$$\begin{aligned}
& y_i^r (\lambda_r \langle w, \phi(x_i^r) \rangle + (1 - \lambda_r)(\langle \boldsymbol{v}, e_r \otimes \psi(x_i^r) \rangle) + b_r) + \xi_i^r = p_i^r \\
& \implies \lambda_r \left\langle \sum_{s=1}^{T} \lambda_s \sum_{j=1}^{m_s} \alpha_i^s \left\{ y_i^s \phi(x_j^s) \right\}, y_i^r \phi(x_i^r) \right\rangle \\
& \quad + (1 - \lambda_r) \left\langle (E_\otimes)^{-1} \sum_{s=1}^{T} (1 - \lambda_s) \sum_{j=1}^{m_s} \alpha_i^s y_i^s (e_s \otimes \psi(x_i^s)), y_i^r (e_r \otimes \psi(x_i^r)) \right\rangle + b_r + \frac{\alpha_i^r}{C} = p_i^r.
\end{aligned}$$

For each coefficient $\alpha_i^r$ we get an equality like this one, which, all together and alongside (4.39) form a system of equations that can be expressed as

$$\left[\begin{array}{c|c} 0_{T \times T} & A^\intercal Y \\ \hline Y A & \left( \Lambda Q \Lambda + (I_n - \Lambda) \widetilde{Q} (I_n - \Lambda) \right) + \frac{1}{C} I_n \end{array}\right] \begin{bmatrix} b_1 \\ \vdots \\ b_T \\ \boldsymbol{\alpha} \end{bmatrix} = \begin{bmatrix} \mathbf{0}_T \\ \boldsymbol{p} \end{bmatrix}. \tag{4.42}$$

Again, $\Lambda$ is the matrix defined in (4.26), and we have kernel matrix $\bar{Q}$, defined as

$$\bar{Q} = \Lambda Q \Lambda + (I_n - \Lambda) \widetilde{Q} (I_n - \Lambda),$$

appears again in the dual problem. This matrix is computed with the kernel function (4.20) and combines the common information, in matrix $Q$, and that from tasks-specific parts coupled through the GL regularization, in matrix $\widetilde{Q}$.

## 4.4 Adaptive Graph Laplacian Algorithm

Until now, in this chapter we have seen GL 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 (4.1) to enforce a coupling between tasks according to the graph 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, which we presented in (Ruiz et al., 2021a), we discuss the procedure and its computational cost, and also show how the distances can be computed in kernel spaces.

### 4.4.1 Motivation and 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 $r, s = 1, \dots, T$.
- The rows of $A$ add up to 1.

The first one is a necessary condition to express the regularizer of (4.1) using the Laplacian matrix $L$. The second requirement is a natural one, and the third is meant to offer a better interpretability of the matrix $A$, since we can view each row as a probability distribution. Then, we can interpret the entropy of the rows as a measure of how sparse or concentrated are its connection with other tasks. 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 $a_r^s \in [0, 1]$ due the conditions stated before, and reaches its maximum when the distribution is uniform. If the weight $a_r^r = (A)_{rr}$ is 1 and the rest, $a_r^s$, $s \neq r$, are 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 $A = \frac{1}{T}\mathbf{1}_T\mathbf{1}_T^\intercal$ 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 situations we look at the following simplified formulation for the convex GL MTL problem,

$$\begin{aligned}\underset{w,\boldsymbol{v},\boldsymbol{b}}{\arg\min} \quad & C\sum_{r=1}^{T}\sum_{i=1}^{m_r} \ell(\lambda_r \langle w, \phi(x_i^r)\rangle + (1-\lambda_r)\langle v_r, \psi(x_i^r)\rangle + b_r, y_i^r) \\ & + \nu \sum_{r=1}^{T}\sum_{s=1}^{T} (A)_{rs} \|v_r - v_s\|^2 + \sum_{r=1}^{T} \|v_r\|^2 + \|w\|^2 .\end{aligned} \tag{4.43}$$

When we use the minimal entropy matrix $A = I_T$, it is equivalent to the problem

$$\underset{w,\boldsymbol{v},\boldsymbol{b}}{\arg\min} \quad C\sum_{r=1}^{T}\sum_{i=1}^{m_r} \ell(\lambda_r \langle w, \phi(x_i^r)\rangle + (1-\lambda_r)\langle v_r, \psi(x_i^r)\rangle + b_r, y_i^r) + \sum_{r=1}^{T} \|v_r\|^2 + \|w\|^2,$$

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 $A = \frac{1}{T}\mathbf{1}_T\mathbf{1}_T^\intercal$, if $\nu$ is large enough, (4.43), it tends to

$$\underset{w,\boldsymbol{v},\boldsymbol{b}}{\arg\min} \quad C\sum_{r=1}^{T}\sum_{i=1}^{m_r}\ell(\lambda_r\left\langle w,\phi(x_i^r)\right\rangle + (1-\lambda_r)\left\langle v,\psi(x_i^r)\right\rangle + b_r, y_i^r) + T\left\|v\right\|^2 + \left\|w\right\|^2,$$

where a convex combination of two common models is used. In the linear case, or when $\phi = \psi$, it is almost equivalent to a CTL approach, since we would use a single common model for all tasks but with task-specific biases.

Between these two extremes of minimal and maximal entropy, there exists an infinite range of matrices whose rows have intermediate entropies. Our goal is then to find an adjacency matrix $A$ that reflects the underlying tasks relations and, therefore, helps to improve the learning process. To find such an adjacency matrix, we rely on the distances computed between the task parameters; if two tasks parameters are close, those tasks should be strongly related. Consider the Laplacian term

$$\sum_{r=1}^{T}\sum_{s=1}^{T}(A)_{rs}\left\|v_r - v_s\right\|^2;$$

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; however, this is a trivial choice of matrix $A$, because it does not give any information about the underlying tasks relations. To avoid falling in such a trivial solution, we add to the objective function the negative entropies of the rows of $A$. Thus, the optimization problem to be solved is

$$\begin{aligned}\underset{\substack{w,\boldsymbol{v},\boldsymbol{b};\\ A\in(\mathbb{R}_{\geq 0})^T\times(\mathbb{R}_{\geq 0})^T,\\ A\mathbf{1}_T=\mathbf{1}_T}}{\arg\min} \quad & C\sum_{r=1}^{T}\sum_{i=1}^{m_r}\ell(\lambda_r\left\langle w,\phi(x_i^r)\right\rangle + (1-\lambda_r)\left\langle v_r,\psi(x_i^r)\right\rangle + b_r, y_i^r) \\ & + \frac{\nu}{2}\sum_{r=1}^{T}\sum_{s=1}^{T}(A)_{rs}\left\|v_r - v_s\right\|^2 + \frac{1}{2}\sum_{r=1}^{T}\left\|v_r\right\|^2 + \frac{1}{2}\left\|w\right\|^2 \\ & - \mu\sum_{r=1}^{T}H(\boldsymbol{a}^r),\end{aligned} \tag{4.44}$$

where $\boldsymbol{v}^\intercal = (v_1^\intercal, \ldots, v_T^\intercal)$ and $\boldsymbol{b} = (b_1, \ldots, b_T)$. Also, $\mathbb{R}_{\geq 0}$ 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 $\mu$ regulates how much the entropy is promoted; the bigger $\mu$ is, the closer the solution for $A$ should be to the constant matrix, which has rows of maximal entropy. Using the

tensor product formulation of (4.17), this problem can be expressed as

$$\begin{aligned}\underset{\substack{(w,\boldsymbol{v})\in\mathcal{H}_\phi\times(\mathbb{R}^T\otimes\mathcal{H}_\psi),\boldsymbol{b};\\ A\in(\mathbb{R}_{\geq 0})^T\times(\mathbb{R}_{\geq 0})^T,\\ A\mathbf{1}_T=\mathbf{1}_T}}{\arg\min} & \sum_{r=1}^{T}\sum_{i=1}^{m_r}\ell(y_i^r,\langle(w,\boldsymbol{v}),(\lambda_r\phi(x_i^r),(1-\lambda_r)(e_r\otimes\psi(x_i^r)))\rangle+b_r)\\ &+\left\langle(w,\boldsymbol{v}),(I_{\mathcal{H}_\phi}\times(E\otimes I_{\mathcal{H}_\psi}))(w,\boldsymbol{v})\right\rangle-\mu\sum_{r=1}^{T}H(\boldsymbol{a}^r),\end{aligned} \tag{4.45}$$

where $E=(\nu L+I_T)$, $L=\operatorname{diag}(A\mathbf{1}_T)-A$, and $\boldsymbol{v}$ as defined in (4.16).

### 4.4.2 Algorithm and Analysis

To find the solution of the optimization problem (4.44), or equivalently (4.45), we will use a alternate descent minimization algorithm: we first fix $A$ and find optimal values for $w,\boldsymbol{v},\boldsymbol{b}$; then we fix $w,\boldsymbol{v},\boldsymbol{b}$ and find the optimal matrix $A$. Given a convex loss function $\ell$, the optimization problem of (4.44) is convex in the parameters $(w,\boldsymbol{v},\boldsymbol{b})$ and in $A$, but not jointly convex in $(w,\boldsymbol{v},b,A)$. Then, an iterated procedure where a alternate optimization is done, as our proposal, ensures that a local minimum is found, albeit, not a global one.

More concretely, in the first step, we fix the matrix $A$ and find the optimal task parameters $w,\boldsymbol{v},\boldsymbol{b}$ that are the solutions to the problem

$$\begin{aligned}\underset{w,\boldsymbol{v},\boldsymbol{b}}{\arg\min}\quad & C\sum_{r=1}^{T}\sum_{i=1}^{m_r}\ell(\lambda_r\langle w,\phi(x_i^r)\rangle+(1-\lambda_r)\langle v_r,\psi(x_i^r)\rangle+b_r,y_i^r)\\ &+\frac{\nu}{2}\sum_{r=1}^{T}\sum_{s=1}^{T}(A)_{rs}\|v_r-v_s\|^2+\frac{1}{2}\sum_{r=1}^{T}\|v_r\|^2+\frac{1}{2}\|w\|^2.\end{aligned} \tag{4.46}$$

To solve this problem, its corresponding dual problem is actually minimized, and optimal dual coefficients $\boldsymbol{\alpha}^*$ are obtained. In the case of the L1-SVM-based model, the problem (4.25) is used, while for the L2, and LS-SVM variants, we have (4.34) and (4.42), respectively.

In the second step, we fix $w,\boldsymbol{v},\boldsymbol{b}$ and find the optimal matrix $A$ that is the solution to the problem

$$\underset{\substack{A\in(\mathbb{R}_{\geq 0})^T\times(\mathbb{R}_{\geq 0})^T,\\ A\mathbf{1}_T=\mathbf{1}_T}}{\arg\min}\ \frac{\nu}{2}\sum_{r=1}^{T}\sum_{s=1}^{T}(A)_{rs}\|v_r-v_s\|^2-\mu\sum_{r=1}^{T}H(\boldsymbol{a}^r). \tag{4.47}$$

Here we see the role of the entropy term since without it the trivial solution would be $(A)_{rs}=\delta_{rs}$, the identity matrix, which corresponds to the minimum-entropy solution $A=I_T$; however, using the entropy term, different, more informative 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 (4.47), we have

$$\frac{\partial}{\partial(A)_{rs}}J(A)=\frac{1}{2}\left(\nu\|v_r-v_s\|^2+\mu\log(A)_{rs}+\mu\right),$$

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 solution is

$$(A)_{rs} = \frac{\exp -\frac{\nu}{\mu} \|v_r - v_s\|^2}{\sum_t \exp -\frac{\nu}{\mu} \|v_r - v_t\|^2}. \tag{4.48}$$

In the second step of our proposed algorithm we need the distances $\|v_r - v_s\|^2$ to define the problem (4.47) and find the optimal adjacency matrix; however, computing such distances when $\boldsymbol{v}_r$ are elements of the RKHS $\mathcal{H}_\psi$ is not trivial. Next, we show how we can use the dual solution to get them. Recall that the first step requires solving the dual problem corresponding to problems (4.22), (4.29) or (4.36) for L1, L2 or LS-SVM, respectively. Observe that all these convex GL primal problems are particular cases of the problem (4.17) when $E = (I_T + \nu L)$. Therefore, we can apply Lemma 4.7 and express the solution $\boldsymbol{v}$ as in (4.19), that is:

$$\begin{aligned}\boldsymbol{v} &= \sum_{r=1}^{T}(1-\lambda_r)\sum_{i=1}^{m_r}\alpha_i^r y_i^r \left(\operatorname{vect}\left((E^{-1})_{1r},\ldots,(E^{-1})_{Tr}\right)\otimes\psi(x_i^r)\right)\\ &= \left((I_T+\nu L)\otimes I_{\mathcal{H}}\right)^{-1}\sum_{r=1}^{T}(1-\lambda_r)\sum_{i=1}^{m_r}\alpha_i^r y_i^r (e_r\otimes\psi(x_i^r)),\end{aligned}$$

where $\alpha_i^r$ are the dual solutions, and recall that we have different dual problems for the L1, L2 and LS-SVM. 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 $r, s = 1,\ldots,T$. These inner products, with our formulation for $\boldsymbol{v}$ as given in (4.16), are

$$\langle v_r, v_s\rangle = \langle (e_r^\intercal\otimes I_{\mathcal{H}})\boldsymbol{v}, (e_s^\intercal\otimes I_{\mathcal{H}})\boldsymbol{v}\rangle,$$

where

$$\begin{aligned}(e_r^\intercal\otimes I_{\mathcal{H}})\,\boldsymbol{v} &= (e_r^\intercal\otimes I_{\mathcal{H}})\left(\left((I_T+\nu L)\otimes I_{\mathcal{H}}\right)^{-1}\sum_{t=1}^{T}(1-\lambda_t)\sum_{i=1}^{m_t}\alpha_i^t y_i^t(e_t\otimes\psi(x_i^t))\right)\\ &= \sum_{t=1}^{T}(1-\lambda_t)\sum_{i=1}^{m_t}\alpha_i^t y_i^t\left((e_r^\intercal(I_T+\nu L)^{-1}e_t)\otimes\psi(x_i^t)\right)\\ &= \sum_{t=1}^{T}(1-\lambda_t)\sum_{i=1}^{m_t}\alpha_i^t y_i^t\left(((I_T+\nu L)^{-1}_{rt})\otimes\psi(x_i^t)\right).\end{aligned}$$

Then, using $E = I_T + \nu L$, as defined in (4.24), the inner product $\langle v_r, v_s\rangle$ is computed as

$$\begin{aligned}&\langle (e_r^\intercal\otimes I_{\mathcal{H}})\boldsymbol{v}, (e_s^\intercal\otimes I_{\mathcal{H}})\boldsymbol{v}\rangle\\ &= \left\langle\sum_{t=1}^{T}(1-\lambda_t)\sum_{i=1}^{m_t}\alpha_i^t y_i^t\left((E^{-1})_{rt}\otimes\psi(x_i^t)\right), \sum_{\tau=1}^{T}(1-\lambda_\tau)\sum_{i=1}^{m_t}\alpha_i^\tau y_i^\tau\left((E^{-1})_{st}\otimes\psi(x_i^\tau)\right)\right\rangle\\ &= \sum_{t=1}^{T}\sum_{\tau=1}^{T}(1-\lambda_t)(1-\lambda_\tau)\sum_{i=1}^{m_t}\sum_{i=1}^{m_t}\alpha_i^t y_i^t\alpha_i^\tau y_i^\tau\left\langle\left((E^{-1})_{rt}\otimes\psi(x_i^t)\right), \left((E^{-1})_{st}\otimes\psi(x_i^\tau)\right)\right\rangle,\end{aligned}$$

**Algorithm 2:** Adaptive GL algorithm.

**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
$o^{\text{old}} = \infty$
$A = A_0$ // Constant matrix
**while** *True* **do**
  $L_{\text{inv}} \leftarrow \text{getInvLaplacian}(A)$ // Step 0
  $\alpha_{\text{opt}} \leftarrow \text{solveDualProblem}((X, y), L_{\text{inv}}, \text{params})$ // Step 1
  $o \leftarrow \text{computeObjectiveValue}((X, y), L_{\text{inv}}, \alpha_{\text{opt}})$ // Objective function value
  **if** $o^{old} - o \leq \delta_{tol}$ **then**
    break // Exit condition
  **end**
  $o^{old} \leftarrow o$
  $D \leftarrow \text{computeDistances}((X, y), L_{\text{inv}}, \alpha_{\text{opt}})$ // Step 2
  $A \leftarrow \text{updateAdjMatrix}(D, \text{params})$ // Step 3
**end**
**return** $\alpha_{opt}, A$

which, using a matrix formulation, can be expressed as

$$\langle v_r, v_s \rangle = \boldsymbol{\alpha}^\intercal \left(I_n - \Lambda\right) \widetilde{Q^{rs}} \left(I_n - \Lambda\right) \boldsymbol{\alpha}, \tag{4.49}$$

where $\widetilde{Q^{rs}}$ is the kernel matrix computed using the kernel function

$$\widetilde{k^{rs}}(x_i^t, x_j^\tau) = (I_T + \nu L)^{-1}_{rt} (I_T + \nu L)^{-1}_{s\tau} k_\psi(x_i^t, x_j^\tau).$$

Observe here that for each pairwise distance we need Therefore, the distances are computed as

$$\|v_r - v_s\|^2 = \boldsymbol{\alpha}^\intercal \left(I_n - \Lambda\right) \left(\widetilde{Q^{rr}} + \widetilde{Q^{ss}} - 2\widetilde{Q^{rs}}\right) \left(I_n - \Lambda\right) \boldsymbol{\alpha}. \tag{4.50}$$

Observe here that for each pairwise distance $\|v_r - v_s\|^2$ we need to compute three $n \times n$ matrices, where $n = \sum_{r=1}^{T} m_r$, and perform the matrices product defined in (4.50).

For a better understanding, we now describe the entire procedure. We start from an agnostic point of view, with a fixed maximal entropy matrix $A^0 = \frac{1}{T}\mathbf{1}_T\mathbf{1}_T^\intercal$, where all tasks are equally related among them. Note that the inverse of $(I_T + \nu L)$ is needed for the definition of the dual problem, and also for the distance computations. Thus, at the beginning of each iteration, we compute the inverse of the matrix $(I_T + \nu L^0)$, where $L^\tau$ is the Laplacian matrix corresponding to the adjacency matrix $A^\tau$ in the $\tau$-th iteration. After this, we find the solutions of the problem (4.46); we actually solve the corresponding dual problem and obtain the dual solution $\boldsymbol{\alpha}^*$, which has a correspondence with the optimal primal variables $w, \boldsymbol{v}, \boldsymbol{b}$. Finally, we can compute the distances between each pair of parameters $v_r, v_s$ as shown in (4.50) for $r, s = 1, \ldots, T$, and use these distances to obtain an adjacency matrix $A^1$ using (4.48). This in an iterated algorithm and all these steps are repeated with the new adjacency matrix, until convergence of the value of the objective function defined in (4.44) or a maximum number of iterations is reached.

To sum it up, this is Algorithm 2, which consists on an iterated procedure where the following steps are repeated until convergence:

- Step 0: invert the matrix $(I_T + \nu L)$.
- Step 1: minimize the dual problem to obtain $\alpha^*$.
- Step 2: compute the distances $\|v_r - v_s\|^2$ between task parameters.
- Step 3: update the adjacency matrix $A$ using (4.48).

To study the computational cost of our algorithm we consider the cost of each step, where, for a simpler notation, we will use $n = \sum_{r=1}^{T} m_r$.

In step 0, the inverse of the $T \times T$ Laplacian matrix, which is used in the first and second steps, has a cost of $C_0 = O(T^3)$. Step 1, where the dual problem is solved, has the standard cost corresponding to these SVM variants, which all have a complexity $C_1 = O(n^{2+\epsilon})$ with $O(n^2) \leq C_1$. The step 2, the distance computations, involves the inner products $\langle v_r, v_s \rangle$ shown in (4.49) for $r, s = 1, \ldots, T$, each with a cost $n_{\text{sv}}^2$ with $n_{\text{sv}}$ being the number of support vectors; then, the complexity of the second step is $C_2 = O(T^2 n_{\text{sv}}^2)$. Finally, in step 3, involving the update of the adjacency matrix $A$, we perform the computation of the $T \times T$ elements of $A$ using equation (4.48), which has then a total cost of $C_3 = O(T^2)$. Therefore, the total computational cost of each iteration is

$$C_0 + C_1 + C_2 + C_3 = O(T^3 + n^{2+\epsilon} + T^2 n_{\text{sv}}^2 + T^2).$$

Assuming a standard situation, where $n$ is much larger than $T$, steps 1 and 2 have clearly the greater cost; however, it is difficult to determine what steps are more computationally challenging, since it depends on the specific problem being solved. Anyway, step 2, unlike step 1, can be easily parallelized, computing each distance at the same time, which would result in a cost of $O(n_{\text{sv}}^2)$ for that step.

## 4.5 Conclusions

In this chapter, we have focused on the GL approach for MTL, which penalizes the distances between task models; we consider that the tasks can be interpreted as nodes in a graph, and the corresponding adjacency matrix values are used to weight the sum of pairwise distances. Since this method relies on distance computation, it is trivial to apply it in the linear case, but it is more challenging when non-linear transformations, related to reproducing kernels, are applied.

To overcome this issue, in Section 4.1 we present an MTL framework based on tensor product of RKHSs. We describe some relevant concepts, and we develop the necessary elements to define the kernel of the tensor product space as the product of the kernels in each space. Then, we show how they can be applied in the context of MTL: we define a kernel between tasks using a fixed tasks relation matrix and, given a kernel defined over the feature space, the MTL kernel can be defined as the product of these two kernels.

Next, in Section 4.2 we describe the GL MTL formulation for kernel methods, and specifically for the L1-SVM. We first present the linear case, and then, with the tensor product of RKHSs definitions, we present the extension to the kernelized case.

We combine this GL approach with the convex combination of Chapter 3 in Section 4.3. Again, for the kernel case we need to use the concepts related to tensor product of RKHSs. We describe the convex GL MTL for the L1, L2 and LS-SVM.

Finally, in Section 4.4 we propose an algorithm to automatically learn the adjacency weights between tasks from data. We motivate this method using the entropy interpretation of the rows of the adjacency matrix, we describe the algorithm and discuss its computational complexity.

Chapter 5

# Experiments

In the previous chapter, we have given the theoretical framework of our proposals for Multi-Task Learning (MTL). In Chapter 3 we have presented the convex MTL formulation that uses a convex combination of common and task-specific parts, which is convenient for interpretability and hyperparameter selection. We developed the convex formulation for kernel methods, such as L1, L2 and LS-Support Vector Machine (SVM), and for Neural Networks (NNs). We also show how we can optimally select the convex combination of pre-trained models. In this chapter we will present several experiments to better understand this convex formulation, and also to test its performance on MTL problems. Moreover, in Chapter 4 we presented the Graph Laplacian (GL) formulation for MTL, which we have combined also with the convex approach to define the convex GL formulation. We developed it for linear models and then for kernel methods, such as the L1, L2 and LS-SVMs. This approach, however, involves the selection of an adjacency matrix that reflects the relations between the tasks. We proposed also the Adaptive GL approach, which uses a data-driven procedure to automatically learn this adjacency matrix. We will present in this chapter multiple experiments to test a fixed convex GL model, as well as its adaptive version.

More specifically, in Section 5.1 we present experiments that test the convex formulation with kernel methods. In Section 5.2 we present the application of the convex formulation to the problem of predicting the production of solar and wind energy. In Section 5.3 we give experiments that consider Multi-Task (MT) image datasets and, thus, NNs are the models better suited for this kind of problems, in particular we test our proposed convex MTL NN. Then, in Section 5.4 we give experiments with models that implement GL-based MTL. And, finally, in Section 5.5 we illustrate our adaptive algorithm for GL-based MTL in several experiments with real and synthetic data.

## 5.1 Convex Multi-Task Learning with Kernel Methods

In this section we present the experiments used to test the convex MTL formulation with kernel models, which we have proposed in (Ruiz et al., 2019) and (Ruiz et al., 2021b) and have been explained in Section 3.1. First, following our work in (Ruiz et al., 2019), experiments comparing convex and additive MTL formulations in the L1-SVM are described. Next, we present experiments where multiple convex MTL models, using L1, L2 and LS-SVMs, as well as optimal convex combinations 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 (3.8), uses task-specific hyperparameters $\lambda_r$, 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. In general, to select the hyperparameters $p_1, \dots, p_L$ of a learning algorithm using CV, the following problem is solved:

$$p_1^*, \dots, p_L^* = \underset{p_i \in \mathcal{S}_{p_i}, i=1,\dots,L}{\arg\min} \sum_{j=1}^{F} \rho(\mathcal{A}(p_1, \dots, p_L; (X^j_{\text{train}}, y^j_{\text{train}})); (X^j_{\text{val}}, y^j_{\text{val}})), \tag{5.1}$$

where $\rho$ is some score measure; $\mathcal{A}$ is our choice of learning algorithm, $p_1, \dots, p_L$ are the parameters on which $\mathcal{A}$ depends and $\mathcal{S}_{p_i}$ a feasible space for the parameter $p_i$, and $(X^j_{\text{train}}, y^j_{\text{train}}), (X^j_{\text{val}}, y^j_{\text{val}})$ are the training and validation sets, respectively. We are using $F$ folds, that is, for $j = 1, \dots, F$, we select different training and validation sets and we train our algorithm $\mathcal{A}$ on the training set $(X^j_{\text{train}}, y^j_{\text{train}})$; then we use $\rho$ to measure the performance of our model on $(X^j_{\text{val}}, y^j_{\text{val}})$. 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 $\gamma$. For regression problems we also add a third parameter: $\epsilon$. That is, to select the hyperparameters of a standard SVM for a single task we have to solve the problem

$$C^*, \gamma^*(, \epsilon^*) = \underset{\substack{C \in \mathcal{S}_C; \\ \gamma \in \mathcal{S}_\gamma; \\ (\epsilon \in \mathcal{S}_\epsilon);}}{\arg\min} \sum_{j=1}^{F} \rho(\mathcal{A}(C, \gamma(, \epsilon); (X^j_{\text{train}}, y^j_{\text{train}})); (X^j_{\text{val}}, y^j_{\text{val}})),$$

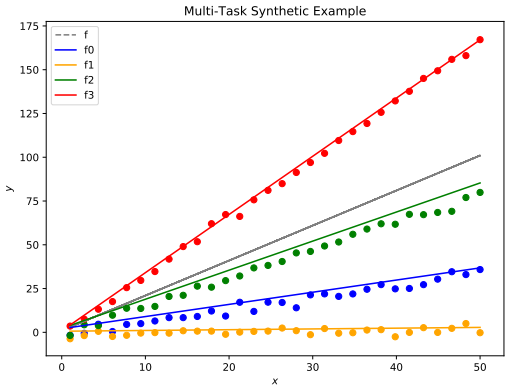
which is typically feasible by using a grid search method in the space $\mathcal{S}_C \times \mathcal{S}_\gamma (\times \mathcal{S}_\epsilon)$.

However, using the convex MTL formulation, we have the following hyperparameters:

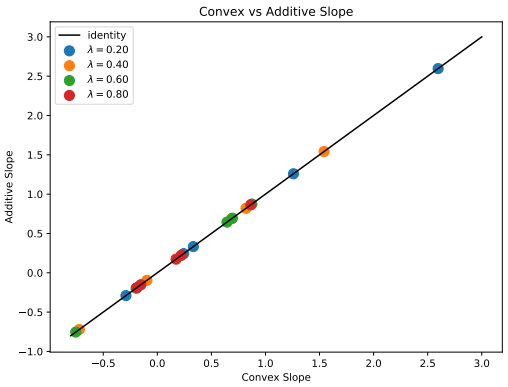
- the regularization parameter $C$;
- the common kernel width $\gamma$ and task-specific ones $\gamma_1, \dots, \gamma_T$;
- the convex combination parameters $\lambda_1, \dots, \lambda_T$; and
- possibly the $\epsilon$ parameter.

That is, there are at least $2T + 2$ hyperparameters that have to be selected. Even with $T = 2$, a search space of dimension 6 can be computationally unfeasible to cover with a grid search. Although there exist other methods to select thy hyperparameters, such as random search or Bayesian optimization, we will only consider CV in this work.

To face this issue and select the optimal parameters for the convex MTL formulation we use the following strategy strategy. For the kernel widths, 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 $\epsilon$ values, but we discard them. Observe that in the CTL approach, where a single task is solved, and in the ITL approach, where we solve the tasks independently, we have either 2 or 3 hyperparameters. We solve the corresponding problems, which are represented in Equation (5.1), and using grid search, we obtain the optimal common $C^*, \gamma^*$ and the task-specific ones $C_r^*, \gamma_r^*$ for $r = 1, \dots, T$, and also the respective $\epsilon^*$ values in the




(A) Synthetic problem, 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 5.1: Synthetic problem and comparison of convex and additive formulations weights.

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 $\lambda$ for all tasks, that is, $\lambda_1 = \ldots = \lambda_T = \lambda$. With these considerations, the problem to select the remaining hyperparameters is

$$C^*, \lambda^*(, \epsilon^*) = \underset{\substack{C \in \mathcal{S}_C; \\ \lambda \in \mathcal{S}_\lambda; \\ (\epsilon \in \mathcal{S}_\epsilon);}}{\arg\min} \sum_{j=1}^{F} \rho(\mathcal{A}(C, \lambda, \gamma^*, \gamma_1^*, \ldots, \gamma_T^*(, \epsilon); (X^j_{\text{train}}, y^j_{\text{train}})); (X^j_{\text{val}}, y^j_{\text{val}})),$$

where we have a maximum of 3 hyperparameters.

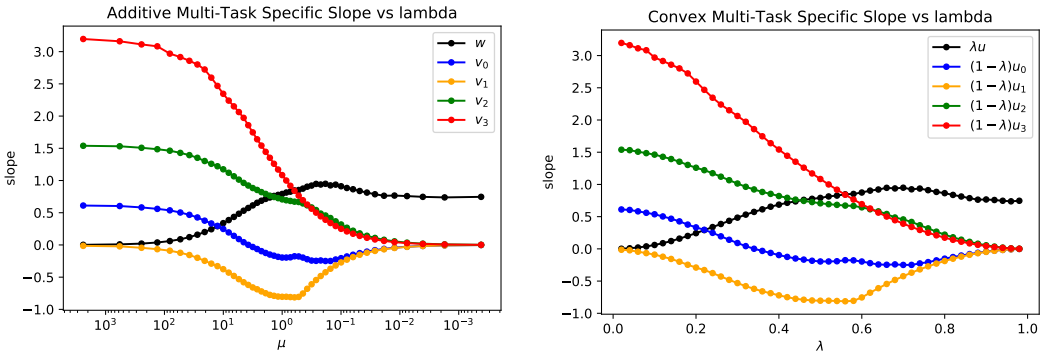
Therefore, these are the two adjustments that we make to carry out the experiments shown in this subsection. The first adjustment concerning kernel widths does not change the model definition, but it assumes that kernel widths that are good for CTL or ITL models are good for the MTL one. Consider the Gaussian kernel

$$k(x, y) = \exp\left(-\gamma \frac{\|x - y\|^2}{2}\right),$$

where $\gamma$ is the kernel width and 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 seems to be a sensible decision. The combined data from all tasks, which is used in the CTL approach, is well "characterized" by the width $\gamma^*$ 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, because we use a single $\lambda$ that determines the specifity of all models. However, we expect that the possible difference in specifity among tasks can be alleviated in the training process.

### 5.1.1 Comparison of Convex and Additive Formulations

In Ruiz et al. (2019), we illustrate the results of the equivalence between the additive and convex formulations in an empirical way. To do this we generate a synthetic problem,




(A) Additive MTL slope estimates weights as a function of $\mu$. We represent the common part of the models as $w$, and the task-specific parts as $v_r$.

(B) Convex MTL slope estimates weights as a function of $\lambda$. We represent the common part of the models as $\lambda u$, and the task-specific parts as $(1-\lambda)u_r$.

FIGURE 5.2: Additive and Convex MTLSVR slope estimates.

shown in Figure 5.1a, as a four tasks linear regression 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 level $\sigma_r$ for each task uniformly in $(0, 5)$. Then, the $r$-th task consists in estimating $m_r$ and $n_r$ from the 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 $\epsilon_r^i \sim N(0, \sigma_r)$ and $f_r(x) = m_r x + n_r$.

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 3.1.

Our first goal is to compare the influence of $\mu$ with that of $\lambda$ in the final models that are obtained. To do that, we need $C$ to be small enough so that the regularization and, therefore, $\mu$ 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 5.1b, we show the estimated slopes for each of the $\lambda$ 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 3.1.

To further compare the two formulations, we visualize how the change on the hyper-parameters imposes a change on the final models. Using the already described synthetic problem of Figure 5.1a, we select values of $\lambda$ 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 Subfigures 5.2a and 5.2b we show the coefficients obtained for the common and task-specific parts with the additive and convex formulations, respectively. We can observe how both graphics show a similar behaviour, with the common model starting in

0 when $\mu$ is large or $\lambda = 0$, and, as $\mu$ decreases and $\lambda$ 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 the 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.

### 5.1.2 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 and ITL approaches with our convex MTL formulation. To do this, we use fourteen different problems, six regression ones and eight classification ones.

Considering the models, we use LX such that it can stand for L1, L2 or LS, we use the following models to test our proposal:

- Common Task Learning LX-SVM (CTL-LX): A single LX-SVM fitted using data from all the tasks, which 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 Section 3.3.
- Convex Multi-Task Learning LX-SVM (MTL-LX): The Convex MTL formulations shown in Section 3.1.

We next describe the problems regression and classification problems. We use a total of six regression problems:

- majorca and tenerife: The goal is to predict the photovoltaic energy production in Mallorca and Tenerife, respectively. The tasks are defined as the energy prediction at each of the 14 hours with sunlight (we remove the night hours).
- boston[1]: This is the Boston housing problem, where the goal is to predict house prices, and the tasks are defined as the predictions in different areas of the city. In this problem we define two tasks: the houses that are next to the river and those that are not.
- california[2]: This is the housing problem in California, where the goal is also to predict house prices. Here the tasks correspond to four different areas of California, according to their distance to the sea.
- abalone[3]: The goal is to predict the number of rings of a species of marine molluscs. The tasks are male, female or infant specimens.

[1] https://www.kaggle.com/datasets/schirmerchad/bostonhoustingmlnd
[2] https://www.kaggle.com/datasets/camnugent/california-housing-prices
[3] https://archive.ics.uci.edu/ml/datasets/abalone

TABLE 5.1: Sample sizes, dimensions and number of tasks of the datasets used.

| Dataset | Size | No. feat. | No. tasks | Avg. task size | Min. task size | Max. task size |
|---|---|---|---|---|---|---|
| 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 |

- crime[4]: 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 nine different states.

For the classification setting, we consider eight problems, six of which are generated by applying different task definitions to two different problems.

- landmine[5]: This is a binary classification problem in which the goal is to decide whether a landmine is present in a given image. Detection of different types of landmines define different tasks.
- binding[6]: 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 and the patterns are the characteristics of the peptides.
- adult[7]: 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[8]: 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 5.1 a description of the characteristics of the datasets.

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

[4] https://archive.ics.uci.edu/ml/datasets/communities+and+crime
[5] https://andreric.github.io/files/datasets/LandmineData_19.mat
[6] https://github.com/pcpLiu/DeepSeqPan/tree/master/dataset
[7] https://archive.ics.uci.edu/ml/datasets/adult
[8] https://www.kaggle.com/datasets/danofer/compass

TABLE 5.2: Hyperparameters, grids used to select them (when appropriate) and hyperparameter selection method for each model.

| | Grid | CTL-L1,2 | ITL-L1,2 | MTL-L1,2 | CTL-LS | ITL-L,S | MTL-LS |
|---|---|---|---|---|---|---|---|
| $C$ | $\{4^k : -2 \leq k \leq 6\}$ | CV | CV | CV | CV | CV | CV |
| $\epsilon$ | $\left\{\frac{\sigma}{4^k} : 1 \leq k \leq 6\right\}$ | CV | CV | CV | - | - | - |
| $\gamma_c$ | $\left\{\frac{4^k}{d} : -2 \leq k \leq 3\right\}$ | CV | - | CTL-L1,2 | CV | - | CTL-LS |
| $\gamma_s^r$ | $\left\{\frac{4^k}{d} : -2 \leq k \leq 3\right\}$ | - | CV | ITL-L1,2 | - | CV | ITL-LS |
| $\lambda$ | $\{0.1k : 0 \leq k \leq 10\}$ | - | - | CV | - | - | CV |

this, we follow the adjustments and experimental procedure described at the beginning of this subsection, reusing the kernel widths from the CTL and ITL approaches and limiting the convex MTL formulation to a single convex combination hyperparameter $\lambda$. In all models considered we use Gaussian kernels, so all features have been normalized to have zero mean and a standard deviation of 1. 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 $\gamma^*$, and the task-specific widths $\gamma_r^*$ in the ITL approach, we use these values to fix them in the convex MTL formulation. Therefore, 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 specific hyperparameters, since $\lambda$ 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 present the test results of models that have been selected with two different validation metrics. In Table 5.3 we present the test results of models selected using the MAE as validation metric, while in Table 5.4 the models have been selected using the MSE. 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. Therefore, we consider both scores as validation metrics to compare the results more fairly. In the classification problems (see Table 5.5) 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 5.2 we show for each hyperparameter of the considered approaches whether they are selected using a CV procedure or reusing them from another 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. In every problem, except for majorca and tenerife, we will consider three external folds, each with the internal three folds for CV. That is, the entire 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

TABLE 5.3: Test MAE (top) and R2 score (bottom) and Wilcoxon-based ranking. Here, the optimal hyperparameters have been selected using the MAE. 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) |
| MTL-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) |
| MTL-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) |
| MTL-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) |
| MTL-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) |
| MTL-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) |
| MTL-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)** |

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 sensible 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 5.3 and 5.4, we show both the MAE and R2 score, closely related to MSE, obtained in the test set. Recall that the results in Table 5.3 have been obtained with models selected using the MAE as the CV metric, while those of Table 5.4 have been selected with the MSE as the metric. For classification, in Table 5.5, we show the F1 and the accuracy scores.

Regarding the numerical results, the tables presenting them 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. Instead of showing every Wilcoxon test result between all pairs of models, which would result in a $12 \times 12$ matrix difficult to interpret, we take the following approach. We first sort the models, according to some criterion, and test the significance between one model and the next one in the sorting order. Then we create a new significance ranking; the ranking increases only if this

TABLE 5.4: Test MAE (top) and R2 score (bottom) and Wilcoxon-based ranking. Here, the optimal hyperparameters have been selected using the MSE. 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) |
| MTL-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) |
| MTL-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) |
| MTL-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) |
| MTL-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 (4) | 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) |
| MTL-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) |
| MTL-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) |

difference is significant. For example, if there are no significant differences between the first and second model, according to the sorting order, they both obtain a ranking of 1.

To apply the Wilcoxon test in the regression problems we combine all blocks. We sort the models (using all blocks) according to their mean score. Then, we test whether 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, and we take as better the model with smaller error.

The results for regression problems are shown in Table 5.3, where the best parameters are selected according to the MAE validation score, and in Table 5.4, 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 the 18 R2 blocks of Table 5.3; in Table 5.4 it obtains 12 out of 18 MAE blocks and 15 out of 18 R2 blocks. Also, a convex MTL approach yields 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 5.5 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 significance ranking shown in the last column. In this case, the Wilcoxon tests are done with samples of size eight, 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

TABLE 5.5: Test F1 (top) and accuracy (bottom) scores, global ranking 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 |
| MTL-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 |
| MTL-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 |
| MTL-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 |
| MTL-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 |
| MTL-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 |
| MTL-LS | **0.757** | **0.764** | **0.767** | **0.851** | **0.850** | **0.851** | **0.944** | **0.858** | **0.830** | 2 | 1 |

size and is given only for illustration purposes. The convex MTL approaches get the best 18 F1 scores out of 24 and the 22 best accuracy scores out of 24. The best overall model is the MTL-L2, while the other convex MTL models are tied with it in the significance ranking.

## 5.2 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 Ruiz et al. (2020b) we presented an application of our convex MTL techniques to renewable energies forecast, in particular we focus on the hourly, day-ahead prediction, that is, the prediction of tomorrow, of each hour, is given today.

Machine Learning (ML), like in other forecasting scenarios, has an increasing presence in energy prediction approaches. The use of ML models requires choosing the predictive features, 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 those from Numerical Weather Prediction (NWP), that can be provided by entities such as the European Centre for Medium-Range Weather Forecasts (ECMWF), which is the one used in this work. For the hourly day-ahead predictions of interest here, we use the ECMWF forecasts of the ECMWF run at 00 UTC in a given day to predict the hourly energy productions the day after.

TABLE 5.6: 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 $\sigma$ is the standard deviation of the target.

| Par. | Grid | CTL | ITL | MTL |
|---|---|---|---|---|
| $C$ | $\{10^k : -1 \leq k \leq 6\}$ | CV | CV | CV |
| $\epsilon$ | $\left\{\frac{\sigma}{2^k} : 1 \leq k \leq 6\right\}$ | CV | CV | CV |
| $\gamma$ | $\left\{\frac{4^k}{d} : -2 \leq k \leq 3\right\}$ | CV | - | CTL |
| $\gamma_r$ | $\left\{\frac{4^k}{d} : -2 \leq k \leq 3\right\}$ | - | CV | ITL |
| $\lambda$ | $\{10^{-1}k : 0 \leq k \leq 10\}$ | - | - | CV |

That is, using the ECMWF generated at 00 UTC, the energy predictions are given for each hour from 24h 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 Photovoltaic (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 assymetric 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 an MTL model, where the models built are specialized in each scenario but all scenarios are used in the 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 next. In the first subsection the experimental methodology is described, showing how the models are chosen and the hyperparameters are selected. Then, the next two subsections present the approach and the detailed task definitions used for solar and wind energy, respectively, as well as the results to measure the performance.

### 5.2.1 Experimental Methodology

Here we describe the methodology that we have followed to conduct the experiments of renewable energy prediction. For both solar and wind energy, we use the same procedure. In each problem, we have train, validation and test sets, each corresponding to one year of data. In the solar energy problems we have data from photovoltaic parks in Mallorca and Tenerife, we consider 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 have the years 2016, 2017 and 2018. For both solar and wind problems

we apply different definition of tasks. For task definition we use only the training samples 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 instances. For example, if we consider the wind velocity to define three tasks, we study the velocities of the training samples 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:

- CTL: 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, $\gamma$ is the kernel width, and $\epsilon$ the width of the error insensitive area.
- ITL: an ITL model, that is, an independent SVR for each task, each with its hyperparameters: $\{C_r, \gamma_r, \epsilon_r\}$ for $r = 1, \ldots, T$.
- MTL: a convex MTL model, as shown in Section 3.1, with its corresponding set of hyperparameters is in principle $\{C, \epsilon, \gamma, \gamma_1, \ldots, \gamma_T, \lambda_1, \ldots, \lambda_T\}$, where $\gamma$ is the kernel width of the common part and $\gamma_r$ the one of the $r$-th specific part; also, $\lambda_r$ 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 instance, prediction at different hours can define different tasks, where one possible value is (hour=14) or (hour=12). The models using this task definition will be named (hour)_ITL and (hour)_MTL. 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 take 14 different hours, hence, 14 tasks, and consider the season definition, which defines the prediction in each season as a different task, hence, four tasks. The combined definition (hour, season) generates $14 \times 4$ possible tasks, whose values can be, for example, (hour = 12, season = summer). The models using this combination will be named (hour, season)_ITL or (hour, season)_MTL.

As explained in Section 5.1, the cost of selecting the optimal hyperparameters scales exponentially with their number, so it is not feasible to use a CV grid search, for example, if we have more than 3 hyperparameters. In the CTL and ITL it is not a problem, since we have 3 hyperparameters per model: the single common SVR, and each one of the task-independent SVRs, respectively. We apply then a CV grid search, with the train and validation sets described above. However, to find the hyperparameters of MTL we have to make some adjustments, as discussed in Section 5.1. First, we apply a convex MTL formulation with a single $\lambda$ 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 $\gamma^*$ and task-specific $\gamma_1^*, \ldots, \gamma_T^*$, and fix them in the MTL model, not including them in the grid search procedure. Then, we apply a CV grid search to find the optimal values of the remaining hyperparameters, that is, $\{C, \epsilon, \lambda\}$. In Table 5.6 we show the method to obtain each hyperparameter, as well as the grids considered in the CV grid search procedures. We use the MAE as the validation metric, because it is the most natural for the $\epsilon$-insensitive loss that is used in the SVRs, and the most common in energy forecasting.

The complete procedure to get the final scores is:

1. **Scale the target and normalize 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, 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 with 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 $\hat{y}_i^r = f(x_i^r)(y_{\max} - y_{\min}) + y_{\min}$.

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 the class `StandardScaler` to normalize the data, and the `TransformedTargetRegressor` class to scale the targets. All these classes are part of the *scikit-learn* library (Pedregosa et al., 2011).

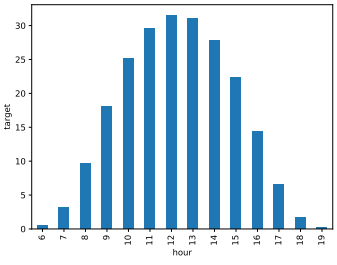
To put our results in perspective, we also show the errors of simple persistence models and of multilayer perceptrons. For the perceptron we use the `MLPRegressor` class of *scikit-learn*. The architecture for both problems consists on fully connected networks with two hidden layers, with 100 and 50 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 $\left\{4^k : -2 \leq j \leq 3\right\}$.

### 5.2.2 Solar Energy

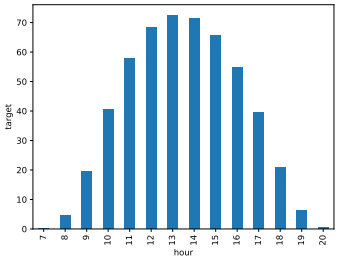
The goal is to predict the hourly energy production in the islands of Mallorca and Tenerife, and the corresponding problems are named majorca and tenerife, respectively. In this subsection the experiments with these solar problems are presented, using the experimental procedure already described. First a description of the problems and the data used is given; then the results are presented and analyzed.

For both problems, majorca and tenerife, the same variables, extracted from the ECMWF, are used:

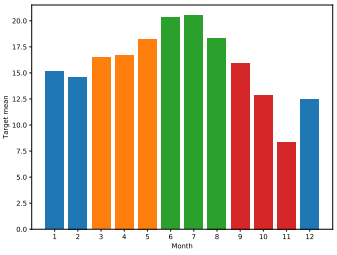
- Surface net solar radiation (SSR).
- Surface solar radiation downwards (SSRD).
- Total Cloud Cover (TCC).
- Temperature at 2 meters above surface (T2M).
- Module of the speed of wind at 10 meters above surface (v10).



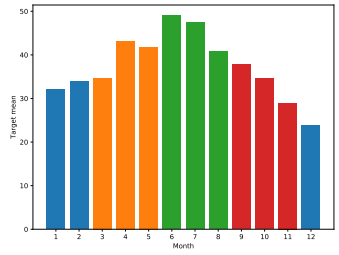

(A) Hourly photovoltaic energy mean in majorca.




(B) Hourly photovoltaic energy mean in tenerife.




(C) Monthly photovoltaic energy mean in majorca.




(D) Monthly photovoltaic energy mean in tenerife.

FIGURE 5.3: Hourly photovoltaic energy mean in `majorca` and `tenerife` measured in MWh. Photovoltaic energy monthly averages for `majorca` and `tenerife`, colored with the tasks defined using the season and measured in MWh. All the histograms have been computed using data from year 2016.

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 km. This correlation will affect those models based on matrix-vector computations, such as linear models, but, also, to some extent, to neural networks. Although ridge (or Tikhonov) regularization can alleviate this issue for these models, kernel-methods, such as the kernel SVMs, seems better suited for this kind of problems. When we apply kernels, like the Gaussian kernel

$$\exp(-\gamma \|x - y\|^2),$$

TABLE 5.7: Test MAEs (left), test MSEs (center), with the corresponding rankings and associated Wilcoxon-based rankings, and optimal mixing $\lambda^*$ (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 | Wil. | MWh | ‰ | rank | Wil. | |
| CTL | 5.265 | 7.265 | (6) | (4) | 59.322 | 112.985 | (6) | (4) | - |
| (season)_ITL | 5.305 | 7.384 | (7) | (5) | 59.591 | 113.498 | (7) | (4) | - |
| (season)_MTL | **4.884** | **6.740** | **(1)** | **(1)** | 53.222 | 101.366 | (2) | (3) | 0.4 |
| (hour)_ITL | 5.083 | 7.015 | (4) | (2) | 54.540 | 103.877 | (3) | (3) | - |
| (hour)_MTL | 4.957 | 6.840 | (2) | (1) | **52.614** | **100.208** | **(1)** | **(1)** | 0.3 |
| (hour, season)_ITL | 5.250 | 7.251 | (5) | (3) | 57.927 | 110.328 | (5) | (4) | - |
| (hour, season)_MTL | 5.038 | 6.952 | (3) | (2) | 54.601 | 103.992 | (4) | (3) | 0.3 |

TABLE 5.8: Test MAEs (left), test MSEs (center), with the corresponding rankings and associated Wilcoxon-based rankings, and optimal mixing $\lambda^*$ (right) of the solar energy models considered in `tenerife`. Base units are either MWh or percentages (%). The best model errors are shown in bold.

| | **MAE** | | | | **MSE** | | | | $\boldsymbol{\lambda^*}$ |
|---|---|---|---|---|---|---|---|---|---|
| | MWh | % | rank | Wil. | MWh | ‰ | rank | Wil. | |
| CTL | 5.786 | 5.373 | (5) | (5) | 88.323 | 76.174 | (5) | (5) | - |
| (season)_ITL | 5.930 | 5.545 | (6) | (5) | 97.454 | 84.611 | (6) | (6) | - |
| (season)_MTL | 5.579 | 5.181 | (4) | (2) | 86.227 | 74.366 | (3) | (3) | 0.8 |
| (hour)_ITL | 5.403 | 5.018 | (2) | (2) | 86.686 | 74.762 | (4) | (4) | - |
| (hour)_MTL | **5.376** | **4.993** | **(1)** | (1) | **84.207** | **72.624** | **(1)** | (1) | 0.7 |
| (hour, season)_ITL | 6.025 | 5.554 | (7) | (6) | 104.536 | 90.297 | (7) | (7) | - |
| (hour, season)_MTL | 5.494 | 5.102 | (3) | (3) | 85.440 | 73.687 | (2) | (2) | 0.7 |

the algorithm learns using the distances among patterns $\|x - y\|$, instead of its features. These distances scale linearly with the dimension of data; consider the features scaled to $[0, 1]$, then a rough estimate of the distance between two patterns would scale with $d$. 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 $\gamma$, 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 a 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 5.3a and 5.3b the hourly averages of the PV energy in MWh are shown for majorca and tenerife. Also, in Subfigures 5.3c and 5.3d the monthly averages, colored by season, where we take the season in first half of each month to select the color.

We show the numerical results for majorca and tenerife in Tables 5.7 and 5.8, 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 $\lambda^*$ selected by the CV for the MTL models.

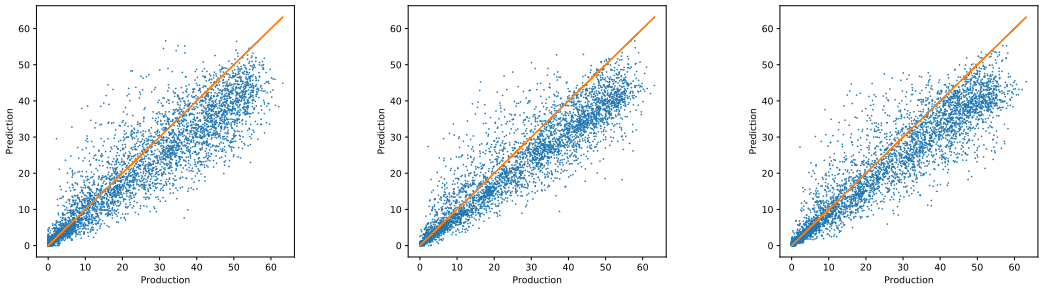
The Wilcoxon test is used to assess statistical significance, but instead of testing every pair of models, the rankings of Tables 5.7 and 5.8 are given. As explained before, we first sort the models and then the Wilcoxon test is applied between each model and the next in the ranking, at the 0.05 level, to determine whether the difference is significant. The test is applied over the list of errors commited by each model in the patterns of the test set; we consider absolute errors for the MAE ranking and squared errors for the MSE one. When the null hypothesis of the Wilcoxon test is rejected, we can assume that the distribution of the difference between the errors does not have its median at 0. With this procedure, a new significance ranking 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. In Tables 5.7 and 5.8 we present these significance rankings for both the MAE and MSE corresponding rankings. For example, in Table 5.7, in terms of MAE, the first model, (season)_MTL, is tested against the second one, (hour)_MTL. However, if we apply the Wilcoxon test to their corresponding list of errors, the null hypothesis is not rejected, so 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 CTL 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)_ITL in majorca which is second; but they can also have bad performances, as the (season)_ITL in tenerife.

The $\lambda^*$ values can help to understand this behaviour. In both problems, the selected values lie far from the extremes 0 (ITL-equivalent) or 1 (CTL-equivalent). Therefore, the validation scores of the CTL or ITL-equivalent models are worse than those of a pure MTL approach. This is reflected also in the test set, as shown in the tables. Moreover, 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)_MTL gets a second best result, which is not significantly worse than the best one in majorca, and is the single best result in tenerife.

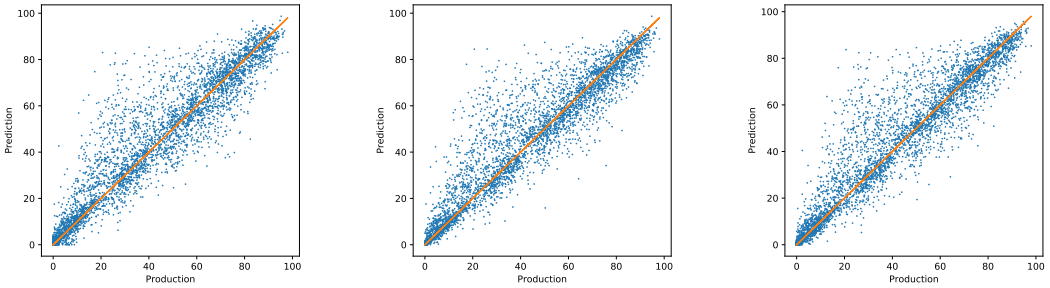
For completeness, the scores of persistence models and a neural network are given here. The persistence forecasts are obtained by predicting at each hour the production value 24 hours prior. By doing this, the MAE scores for majorca and tenerife are 5.776 MWh and 7.766 MWh, which scaled to $[0, 100]$ 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 CTL, (hour)_ITL and (season)_MTL in Figure 5.4. From these scatter plots it seems that the CTL approach has a larger deviation in its predictions, while the




(A) CTL prediction. (B) ITL prediction. (C) MTL prediction.

FIGURE 5.4: Real energy production against the prediction made by the best models in `majorca`; the perfect prediction line is shown in orange. The units of the axis are MWh.




(A) CTL prediction. (B) ITL prediction. (C) MTL prediction.

FIGURE 5.5: Real energy production against prediction made by the best models in `tenerife`; the perfect prediction line is shown in orange. The units of the axis are MWh.

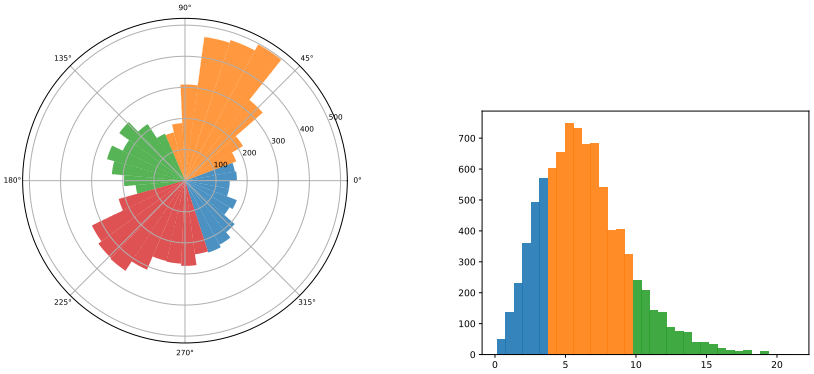
ITL one has a bias, that is, it systematically underestimates the prediction corresponding to larger values of energy production. The MTL approach seems to correct, to a certain degree, this bias of the ITL model, while preserving a smaller variance than the CTL one. For tenerife we plot the predictions of CTL, (hour)_ITL and (hour)_MTL, which are shown in Figure 5.5. 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.

### 5.2.3 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 that of solar radiation. Detecting scenarios where the wind energy production behaves differently but is stable inside 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.

The variables from the ECMWF that we consider as predictors are:

- Eastward component of the wind at 10m (U10).




(A) Histogram of wind angles colored by task definition angle.

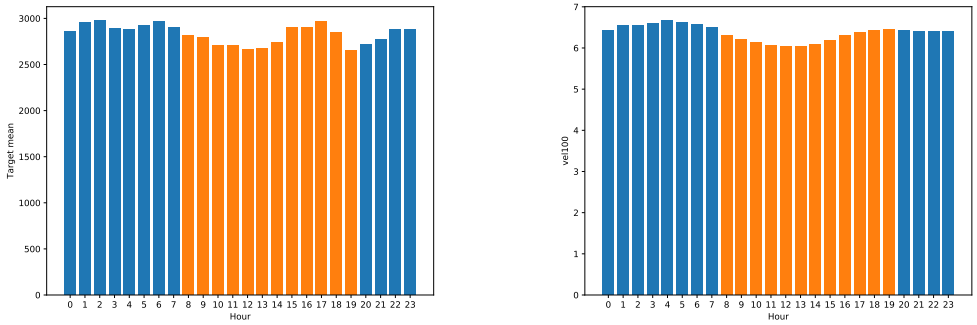
(B) Histogram of wind velocity (m/s) in Sotavento colored by task definition velocity.

FIGURE 5.6: Histograms of wind derived from ECMWF data for the year 2016 in Sotavento.

- Northward component of the wind at 10m (V10).
- Module of velocity of the wind at 10m (vel10).
- Eastward component of the wind at 100m (U100).
- Northward component of the wind at 100m (V100).
- Module of velocity of the wind at 100m (vel100).
- 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, 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 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 different 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 5.6a.
- 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




(A) Hourly mean measured of generated energy (kWh) colored by task definition timeOfDay.

(B) Hourly mean of velocity (m/s) at 100 m colored by task definition timeOfDay.

FIGURE 5.7: Histograms of generated energy and wind velocities derived from ECMWF data for the year 2016 in Sotavento.

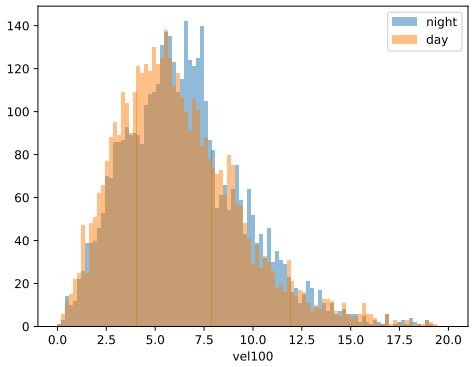



FIGURE 5.8: Histograms of the velocity of wind at 100m derived from ECMWF data for the year 2016 and measured in m/s during the day and night in Sotavento.

the starting point of wind energy generation and its maximum power plateau, before cut-off speed. In Figure 5.6b 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 5.7a and 5.7b the hourly average energy production and wind speed are shown, with the hours colored according to the two tasks defined. In Figure 5.8 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-cut 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 remarkable ones. For the timeOfDay definition, the two histograms of Figure 5.8 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.

Regarding the experimental results for wind energy prediction, in Table 5.9 the MAE and MSE scores are shown, and also the ranking is given in parentheses. Unlike the solar

TABLE 5.9: Test MAEs (left), MSEs (center), with the corresponding rankings and associated Wilcoxon-based rankings, and optimal mixing $\lambda^*$ (right) of the Sotavento wind energy models considered. The best model errors are shown in bold.

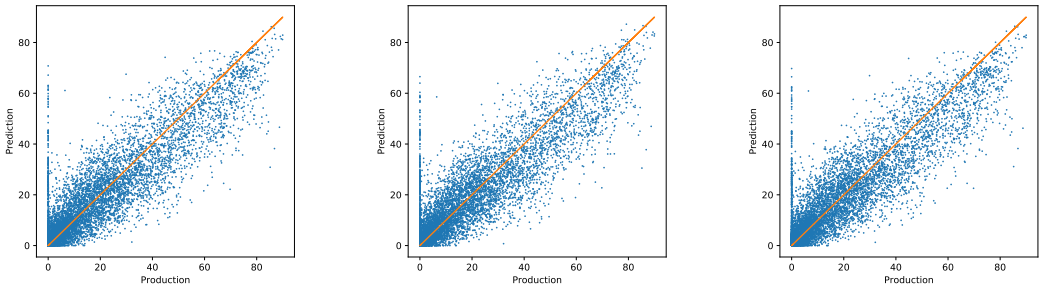
| | **MAE** | | | **MSE** | | | $\boldsymbol{\lambda^*}$ |
|---|---|---|---|---|---|---|---|
| | MWh | rank | Wil. | MWh | rank | Wil. | |
| CTL | **6.132** | **(1)** | **(1)** | 90.228 | (2) | (2) | - |
| (velocity)_ITL | 6.211 | (7) | (3) | 93.363 | (7) | (3) | - |
| (velocity)_MTL | 6.208 | (6) | (3) | 93.199 | (6) | (3) | 0 |
| (timeOfDay)_ITL | 6.283 | (9) | (4) | 93.594 | (9) | (4) | - |
| (timeOfDay)_MTL | **6.132** | **(1)** | **(1)** | 90.228 | (2) | (2) | 1 |
| (timeOfDay, velocity)_ITL | 6.341 | (11) | (4) | 97.250 | (11) | (5) | - |
| (timeOfDay, velocity)_MTL | 6.312 | (10) | (4) | 94.774 | (10) | (4) | 0.4 |
| (timeOfDay, angle)_ITL | 6.266 | (8) | (4) | 93.517 | (8) | (4) | - |
| (timeOfDay, angle)_MTL | **6.132** | **(1)** | **(1)** | 90.228 | (2) | (2) | 1 |
| (timeOfDay, angle, velocity)_ITL | 6.410 | (12) | (4) | 102.031 | (12) | (6) | - |
| (timeOfDay, angle, velocity)_MTL | **6.132** | **(1)** | **(1)** | 90.228 | (2) | (2) | 1 |
| (angle)_ITL | 6.170 | (4) | (3) | 91.586 | (4) | (3) | - |
| (angle)_MTL | 6.135 | (2) | (2) | **90.026** | **(1)** | **(1)** | 0.9 |
| (angle, velocity)_ITL | 6.173 | (5) | (3) | 92.529 | (5) | (3) | - |
| (angle, velocity)_MTL | 6.168 | (3) | (3) | 90.990 | (3) | (3) | 0.7 |

energy case, here the CTL approach seems better suited than using an ITL one. This is also reflected in the selection of $\lambda^*$ values in the models that get best results, which are close to 1, the CTL equivalent case. That is MTL models such as (timeOfDay)_MTL, (timeOfDay, angle)_MTL and (timeOfDay, angle, velocity)_MTL, where $\lambda^* = 1$, obtain the best results and are equivalent to CTL. Also note the cases of (angle)_MTL and (angle, velocity)_MTL, that are second and third best, and where the selected values are $\lambda^* = 0.9$ and $\lambda^* = 0.7$. Those models that put the emphasis on the independent parts, like (timeOfDay, velocity)_MTL, 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 5.9, the significance of the difference between one model and its immediate succesor is tested. 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)_MTL, (timeOfDay, angle)_MTL and (timeOfDay, angle, velocity)_MTL, while the (angle)_MTL gets the second best MAE score. When the MSE scores are analyzed, the roles are reversed, the (angle)_MTL gets the best result, and the CTL 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 behave differently 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 other models considered.

Again, we plot the predictions against the target values of the CTL model and best ITL, (angle)_ITL, and pure MTL, (angle)_MTL, 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 noticeable the frequency of small production values, below 20 %, which is due to the




(A) Best CTL prediction. (B) Best ITL prediction. (C) Best MTL prediction.

FIGURE 5.9: Real energy production against prediction made by the best models for sotavento; the perfect prediction line is shown in orange. The units of the axis are percentage points of the total PV energy installed.

approximate Weibull distribution of wind speeds, where small speed values have higher frequencies.

## 5.3 Convex Multi-Task Learning with Neural Networks

In the experiments presented in the previous sections, we have focused on the convex MTL formulation for kernel methods; however, as described in Section 3.2, we can also use this formulation with NNs. The idea is then to use as the model for each task a convex combination of a common and task-specific NNs.

Neural networks have experimented a great success in many areas such as a Computer Vision and Natural Language Processing; however, these results have been achieved using large networks, with possibly millions of parameters, that need a great amount of data in the training process. For relatively small datasets, say below 25 000 patterns, SVMs usually achieve a better performance, because of the better generalization properties that characterize them. Anyway, nowadays, in the era of big data, many problems are composed of tens of thousands, even millions of patterns. Kernel methods, such as SVMs, due to their computational cost, cannot deal with such problems. With these methods, training time grows faster than quadratically with the number of training patterns, which make them unfeasible on large datasets.

Moreover, data is not always tabular, where the features have no apparent connection among them, but there might exist a spatial structure, as in images, or temporal dependencies, as in time series or natural language. Incoporating the knowledge of such structures with kernel methods is not an easy task, since the feature space where the problem is actually solved is possibly infinite-dimensional, and very difficult to interpret. Neural networks, on the other hand, are very flexible to incorporate such knowledge, and they are easily adaptable to a wide range of architectures without significant modifications of their training procedure. In this section, we want to show the effectiveness of the convex MTL formulation when applied to NNs. To do this, we consider four different image datasets, so convolutional neural networks will be the base of our MTL models. Each of these problems has 70 000 images, so using SVM-based models with the entire dataset is quite time consuming. Instead, we will use two different approaches to MTL with NNs, a feature-based one, the hard sharing of hidden weights, and our proposal, the convex MTL approach.

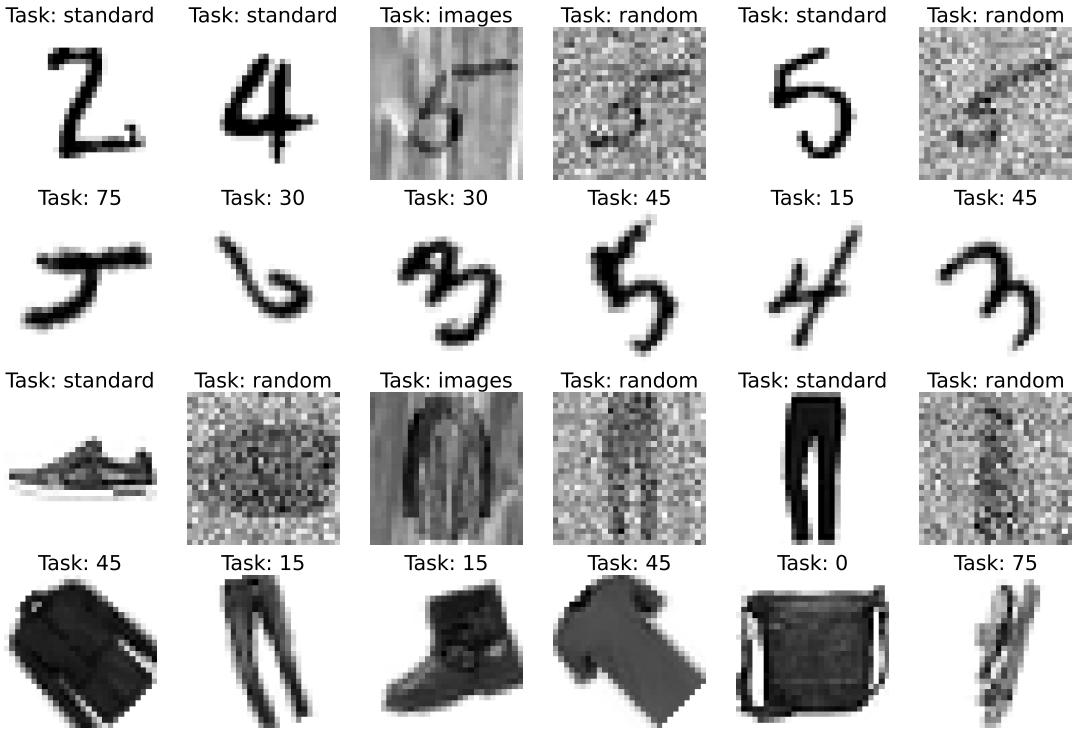



FIGURE 5.10: Images of the four classification problems used. Each image has a title indicating the corresponding task. The rows correspond to var-MNIST, rot-MNIST, var-FMNIST and rot-FMNIST (from top to bottom).

To test the convex MTL neural networks we consider four MT image datasets: var-MNIST and rot-MNIST, based on MNIST (LeCun et al., 1998), and var-FMNIST and rot-FMNIST, based on fashion-MNIST (Xiao et al., 2017). The MNIST and fashion-MNIST datasets are both composed of 70 000 examples of $28 \times 28$ 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, that is, 10 different digits or 10 different types of clothes. These classes are balanced in both datasets. To generate our Multi-Task image datasets, we take the images from MNIST or fashion-MNIST and apply either the *variations* procedure or the *rotation* one. For the var-MNIST and var-FMNIST we use the *variations* procedure where, 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 apply the *rotations* procedure. With 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 consider a random split to divide the original datasets in six groups and each group is applied one of the transformations defined above, so we get six tasks: four with 11 667 examples and two with 11 666. In Figure 5.10 we show examples of the four MTL image problems that we generate, with the corresponding task annotation for each one.

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: an MTL-based neural network using the hard sharing strategy. That is, a single neural network is defined for all tasks, where the first layers are shared among tasks, but a task-specific output layer is used for each task.
- mtlNN: an 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[9]. The architecture of convNet consists, in this order, of 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 apply 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 considered for each task. For the hsNN, a single network with the convNet architecture is considered, but we have a group of 10 output neurons for each task in the problem. For example, if we are using 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 mtlNN we define 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 10 $r$-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 $\alpha$ 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 apply a $2 \times 2$ max pooling layer with a stride of 2. The mtlNN model also has $\lambda$ as a hyperparameter, and it is selected, alongside $\alpha$, using a CV grid search, where the grid for $\lambda$ 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 use

[9] www.pytorch.org/tutorials/intermediate/spatial_transformer_tutorial.html

TABLE 5.10: 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 |
| mtlNN | **0.974** ($\lambda^* = 0.6$) | **0.984** ($\lambda^* = 0.8$) | **0.812** ($\lambda^* = 0.6$) | **0.880** ($\lambda^* = 0.6$) |

TABLE 5.11: Test categorical cross entropy of the averaged logits.

| | var-MNIST | rot-MNIST | var-FMNIST | rot-FMNIST |
|---|---|---|---|---|
| ctlNN | $1.274 \pm 0.143$ | $1.145 \pm 0.039$ | $2.369 \pm 0.183$ | $1.757 \pm 0.075$ |
| itlNN | $1.072 \pm 0.029$ | $0.873 \pm 0.058$ | $2.356 \pm 0.130$ | $1.598 \pm 0.042$ |
| hsNN | $1.087 \pm 0.253$ | $0.898 \pm 0.073$ | $3.067 \pm 0.888$ | $1.888 \pm 0.075$ |
| mtlNN | $\mathbf{0.924 \pm 0.024}$ ($\lambda^* = 0.6$) | $\mathbf{0.831 \pm 0.029}$ ($\lambda^* = 0.8$) | $\mathbf{2.147 \pm 0.090}$ ($\lambda^* = 0.6$) | $\mathbf{1.482 \pm 0.063}$ ($\lambda^* = 0.6$) |

a 5-fold CV scheme. These folds are task-stratified, that is, all of them 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.

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 entire 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 using the logits, in the output neurons, of each instance and averaging them, so we have 10 values, one for each class. In Table 5.10 we show the test accuracy, using the logits average already described, for each of the approaches considered. Other way to visualize these results is to compute the categorical cross entropy directly on the averaged logits, which we show in Table 5.11. In both tables we also show the optimal values for $\lambda$ selected in the CV.

Our proposal, the mtlNN 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 $\lambda$, 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 5.10 and 5.11, where the MTLNN 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

TABLE 5.12: Sample sizes, dimensions and number of tasks of the datasets used.

| Dataset | Size | No. features | No. tasks | Avg. task size | Min. task size | Max. task size |
|---|---|---|---|---|---|---|
| majorca | 15 330 | 765 | 14 | 1095 | 1095 | 1095 |
| tenerife | 15 330 | 765 | 14 | 1095 | 1095 | 1095 |
| binding | 32 302 | 184 | 47 | 687 | 59 | 3089 |
| landmine | 14 820 | 10 | 28 | 511 | 445 | 690 |
| 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 |

effectively capture the differences among different tasks, so it is frequently surpassed by the ITL network.

## 5.4 Convex Graph Laplacian for Multi-Task Learning

The convex MTL that we have seen in the previous experiments assumes that the information that can be shared is common for all tasks. However, there might be problems where there is not an information that is common to all tasks, but there are some elements of knowledge that are shared by each pair of tasks. To take into account these kind of dependences, we have proposed the convex GL MTL formulation in Ruiz et al. (2020a), which we have described in Section 4.3 for the L1, L2 and LS-SVM. In this section we present the experiments that we gave in Ruiz et al. (2020a), which test our proposal and compare it to the baseline models using 6 regression problems and 2 classification ones.

We test our proposal over eight different problems: majorca, tenerife, california, boston, abalone and crime for regression and landmine and binding for classification. Although we have already defined these datasets in Section 5.1, for convenience we summarize here the description of each one again. We point out that the adult and compas datasets, which are present in Section 5.1, were not considered for these experiments. In majorca and tenerife each task goal is to predict the photovoltaic production at different hours. The california and boston datasets, which are obtained from the Kaggle repository, are problems in which the target is the median price of houses and the different tasks are defined using the location of these houses. The rest of the regression datasets are available in the UCI repository. The abalone dataset contains data about sea molluscs and we define as different tasks the prediction of the number of rings in the male, female and infant specimens. The target in the crime problem is to predict the number of crimes per population in different cities, and the prediction in each state is considered a different task. In the classification problems we consider the MHC-I molecule binding problem, which we call binding. In this problem the goal is to predict whether a molecule and a peptide will form a bond. We take the prediction for each MHC molecule as a different task. The landmine problem goal is the detection of landmines, and each type of landmine defines a task. Although the characteristics of these datasets have been presented in Table 5.1, to make the reading easier we give them also in Table 5.12.

To test our proposal we compare it with four alternative models, all based on the Gaussian L1-SVM, which are described next.

- **Common Task Learning SVM (CTL)**, which considers a single SVM model for all tasks, without making use of the task information.

TABLE 5.13: Hyper-parameters, grids used to select them (when appropriate) and hyperparameter selection method for each model.

| | Grid | CTL | ITL | MTL | GLMTL | cvxGLMTL |
|---|---|---|---|---|---|---|
| $C$ | $\{4^k : -2 \leq k \leq 2\}$ | CV | CV | CV | CV | CV |
| $\epsilon$ | $\left\{\frac{\sigma}{4^k} : 1 \leq k \leq 6\right\}$ | CV | CV | CV | CV | CV |
| $\gamma$ | $\left\{\frac{4^k}{d} : -2 \leq k \leq 3\right\}$ | CV | - | CTL | - | CTL |
| $\gamma_s$ | $\left\{\frac{4^k}{d} : -2 \leq k \leq 3\right\}$ | - | CV | ITL | CTL | CTL |
| $\lambda$ | $\{0.2k : 0 \leq k \leq 5\}$ | - | - | CV | - | CV |
| $\mu$ | $\{4^k : -1 \leq k \leq 3\}$ | - | - | - | CV | GLMTL |

- **Independent Task learning SVM (ITL)**, which fits an independent SVM for each task.
- **Convex Multi-Task learning SVM (MTL)**, which considers the convex MTL approach, described in Section 3.1.
- **Graph Laplacian MTL-SVM (GLMTL)**, where only the GL regularization is used to enforce the coupling between the tasks, as presented in Section 4.2; no common part is incorporated.
- **Convex Graph Laplacian MTL-SVM (cvxGLMTL)**, where we use a common and task-specific parts, and those specific parts are coupled through a GL regularization, as presented in Section 4.3.

Each of the considered models has a different set of hyperparameters, which are typically selected from a grid of possible combinations, that is, using a grid search procedure. The cost of this search, however, as explained in Section 5.1, scales exponentially with the number of hyperparameters. We will consider a grid search CV procedure, but, when the number of hyperparameters is greater than three, we will use some simplifications that we detail next. For the CTL approach, where we have 3 hyperparameters, $\{C, \epsilon, \gamma\}$, with $\gamma$ being the kernel width of this common model, we choose all of them via CV. It is the same for the parameters $\{C_r, \epsilon_r, \gamma_r\}$ of the task-specific models in the ITL approach. In the MTL we have the following set of hyperparameters: $\{C, \epsilon, \lambda, \gamma, \gamma_1, \ldots, \gamma_T\}$, which include the kernel width for the common part $\gamma$ and the task-specific widths $\gamma_r$, $r = 1, \ldots, T$, that is a total of $T + 4$ hyperparameters. We proceed as follows: we take the common and task-specific kernel widths as the optimal ones from the CTL and ITL approaches, then we perform a standard CV search for $\{C, \epsilon, \lambda\}$. For the GLMTL approach we use a similar procedure. We use the kernel width selected for CTL and do a grid search CV for $\{C, \epsilon, \mu\}$. Finally, for cvxGLMTL we use the optimal kernel width for CTL in both kernels of the model definition and select for $\mu$ the optimal value for the GLMTL approach. Then, we perform the CV search for $\{C, \epsilon, \lambda\}$. As with the other experiments, in Table 5.13 we give the grids used and illustrate how each hyperparameter is selected in the different models.

Recall that the CV consists on dividing the dataset in multiple folds; then, for each combination of hyperparameters we train with all the folds but one and test its performance on the remaining fold, the validation set. We divide our datasets in folds differently depending on the nature of the problem. In majorca and tenerife, which are time series, and, thus, we need to take into account the time dependencies, we use the data from the years 2013, 2014 and 2015 as train, validation and test sets. As in the previous experiments, for the rest of the problems we consider a nested CV, where now we have three outer folds, and we use two outer of them to select the optimal

TABLE 5.14: In the left, test MAE (top), and test R2 scores (bottom) in the regression problems.

| | maj. | ten. | boston | california | abalone | crime | landmine | binding |
|---|---|---|---|---|---|---|---|---|
| | | | | **MAE** | | | | **F1** |
| CTL | 5.265 | 5.786 | 2.254±0.035 | 41870.820± 76.723 | 1.483±0.039 | 0.078±0.001 | 0.106±0.016 | 0.868±0.002 |
| ITL | 5.119 | 5.341 | 2.779±0.134 | 37043.664±371.549 | 1.488±0.038 | 0.082±0.006 | 0.183±0.034 | 0.901±0.000 |
| MTL | 5.077 | 5.351 | 2.228±0.006 | 36848.971±242.052 | 1.466±0.028 | 0.074±0.003 | 0.150±0.023 | 0.906±0.001 |
| GLMTL | 5.291 | 5.840 | 3.070±0.391 | 37123.515±404.205 | 1.690±0.017 | 0.094±0.006 | 0.227±0.042 | 0.896±0.003 |
| cvxGLMTL | 4.917 | 5.335 | 2.230±0.038 | 36 720.854±225.335 | 1.467±0.026 | 0.074±0.003 | 0.163±0.031 | 0.908±0.001 |
| | | | | **R2** | | | | **Accuracy** |
| CTL | 0.831 | 0.902 | 0.843±0.044 | 0.638 ± 0.005 | 0.560±0.017 | 0.743±0.022 | 0.942±0.004 | 0.791±0.003 |
| ITL | 0.843 | 0.904 | 0.776±0.017 | 0.696 ± 0.005 | 0.550±0.024 | 0.711±0.006 | 0.942±0.004 | 0.850±0.000 |
| MTL | 0.845 | 0.907 | 0.850±0.045 | 0.700 ± 0.003 | 0.566±0.013 | 0.755±0.016 | 0.943±0.004 | 0.858±0.002 |
| GLMTL | 0.832 | 0.894 | 0.490±0.264 | 0.695 ± 0.007 | 0.366±0.027 | 0.596±0.033 | 0.935±0.002 | 0.844±0.005 |
| cvxGLMTL | 0.849 | 0.905 | 0.852±0.046 | 0.702 ± 0.003 | 0.566±0.013 | 0.752±0.016 | 0.944±0.004 | 0.862±0.002 |

TABLE 5.15: Top: Wilcoxon $p$-values of absolute errors of a regression model and the one following it in the MAE ranking and similar F1 score $p$ values. Bottom: with the same scheme, $p$ values of quadratic errors and the R2 score ranking and accuracy scores.

| | majorca | tenerife | boston | california | abalone | crime | classif. |
|---|---|---|---|---|---|---|---|
| | | | | **MAE** | | | **F1** |
| CTL | 0.0000 (3) | 0.0000 (4) | **0.2554 (1)** | 0.0000 (4) | 0.0002 (2) | 0.0000 (2) | 0.0277 (3) |
| ITL | 0.8131 (2) | 0.0035 (2) | 0.0001 (2) | 0.0318 (3) | 0.2546 (2) | 0.3995 (2) | **0.3454 (1)** |
| MTL | 0.0000 (2) | 0.0000 (3) | - **(1)** | 0.0000 (2) | - **(1)** | - **(1)** | 0.0277 (2) |
| GLMTL | 0.4183 (3) | 0.5962 (4) | 0.0621 (2) | 0.5658 (3) | 0.0000 (3) | 0.0000 (3) | - **(1)** |
| cvxGLMTL | - **(1)** | - **(1)** | **0.4113 (1)** | - **(1)** | **0.0771 (1)** | **0.6093 (1)** | **0.3454 (1)** |
| | | | | **R2** | | | **Accuracy** |
| CTL | 0.0032 (3) | 0.0000 (2) | **0.1791 (1)** | 0.0000 (4) | 0.0016 (3) | 0.0001 (2) | 0.3454 (4) |
| ITL | 0.6340 (2) | **0.5999 (1)** | 0.0001 (2) | 0.0035 (3) | 0.3096 (3) | 0.3972 (2) | 0.0277 (3) |
| MTL | 0.0000 (2) | **0.0815 (1)** | - **(1)** | 0.0000 (2) | - **(1)** | - **(1)** | 0.0431 (2) |
| GLMTL | 0.2040 (3) | 0.7790 (2) | 0.0384 (3) | 0.6759 (3) | 0.0000 (4) | 0.0000 (3) | 0.0277 (4) |
| cvxGLMTL | - **(1)** | - **(1)** | **0.2606 (1)** | - **(1)** | 0.0181 (2) | **0.7262 (1)** | - **(1)** |

hyperparameters and the remaining one to measure the fitness of our models. To select the optimal hyperparameters we combine the data from the two outer folds and perform a CV with three inner folds, where we use two for training and one for validation. The folds are selected randomly and stratified by task, so the task proportions are similar in all folds. As validation metrics to select the best hyperparameters we use the MAE in the regression problems and the F1 score in the classification ones. In the training and validation process, we also normalize the data feature-wise using the mean and deviation from the training set, and in the regression problems we scale the targets into the $[0, 1]$ interval with the minimum and maximum from the training set.

In the approaches incorporating a GL we need to select an adjacency matrix. The weights that compose this matrix define the degree of relationship that we expect between tasks, and we introduce this information through the GL regularization. Selecting an adequate graph is not trivial, and an expert knowledge of each specific problem would be necessary. In our experiments, since we do not have any prior knowledge about the tasks and their relations, we use an "agnostic" graph in which every task (node) is connected to all the others; that is, our adjacency matrix is the constant matrix $A = \frac{1}{T}\mathbf{1}\mathbf{1}^\intercal$.

In Table 5.14 we give the test scores for the regression problems; we consider the MAE, which is the most natural for SVMs, and the R2 score which is a normalized MSE.

To show statistical significance we apply the procedure based on the Wilcoxon test, as described before. Recall that we first rank the models, either according to MAE or R2 score, and then the first model is assigned a ranking of 1, and for the second model we

TABLE 5.16: Train MAE in the regression problems (smallest values in bold face).

| | maj. | ten. | boston | california | abalone | crime |
|---|---|---|---|---|---|---|
| | | | | MAE | | |
| CTL | 3.440 | 4.183 | $1.557 \pm 0.198$ | $40502.686 \pm 222.209$ | $1.434 \pm 0.019$ | $0.055 \pm 0.006$ |
| ITL | 3.590 | 3.914 | $1.883 \pm 0.224$ | $34403.940 \pm 83.583$ | $1.399 \pm 0.025$ | $0.050 \pm 0.004$ |
| MTL | 3.649 | 3.921 | $1.522 \pm 0.248$ | $35061.556 \pm 118.259$ | $1.399 \pm 0.027$ | $0.055 \pm 0.007$ |
| GLMTL | **2.630** | **3.728** | $2.077 \pm 0.447$ | **33984.568±151.998** | $1.594 \pm 0.023$ | **0.038±0.002** |
| cvxGLMTL | 3.344 | 4.141 | **1.516±0.270** | $34409.942 \pm 101.472$ | **1.406±0.023** | $0.057 \pm 0.007$ |

use a Wilcoxon test to check if the difference with the first one is significant. If we reject the null hypothesis, we consider that both models are different and we give the second model a ranking of 2, in other case we assign a ranking of 1. We continue proceeding like this for each pair of consecutive models to get a ranking that is somehow statistical significant. We give this ranking, alongside the $p$-values of the corresponding pairwise Wilcoxon test in Table 5.15. We reject the null hypothesis at the 5 % significance level.

It can be observed in Table 5.14 that, in terms of MAE, our proposed cvxGLMTL gets the best results in most regression problems, and even when this is not the case, we can see in Table 5.15 that the difference with the best model is not significant. Only in the case of abalone, the MTL model gets significantly better results, and just in terms of R2 score.

In the case of classification problems, we also give the accuracy and F1 test scores in Table 5.14. We notice that in landmine, which is strongly unbalanced, although the accuracy scores are generally high, the F1 scores are low. On the other hand, in a balanced problem such as binding, both the accuracy and F1 scores are similar.

As it has been pointed out already, in the classification setting we cannot compute the accuracy or F1 score at every instance, so to apply the Wilcoxon test we can only use the score values computed for each of the outer folds in each problem, that is, a total of 6 different values. Given that this is a small sample, the validity of using the Wilcoxon test is not guaranteed. However, for illustration puroposes we provide the Wilcoxon $p$-values and rankings in the last column of 5.15. Observe that now we have a single ranking for all the problems, because we are combining the scores of all the problems, as described above, to apply the Wilcoxon test.

Finally, if we compare the two approaches using the GL regularization, we can see that the GLMTL performs quite well in the classification problems but less so in regression problems. As a possible explanation, recall that a pure GL does not include individual regularizers for each $w_r$, so the models are more prone to overfitting. To support this theory, we give the training errors in Table 5.16, where GLMTL gets the best MAE scores in majorca, tenerife, california and crime, but these results are not transferred completely to the test sets, where cvxGLMTL gets better results.

## 5.5 Adaptive Graph Laplacian for Multi-Task Learning

The convex MTL or even the convex GL approach assume that all tasks are connected or related to each other. However, there might be problems where there are some characteristics shared by all tasks, others that are present in a subgroup of tasks, and others that are task-specific. To take into account this kind of dependences, we have proposed the convex adaptive GL MTL formulation in Ruiz et al. (2021a), which we have described in Section 4.4. In this section we present some experiments to test our proposal, using both synthetic and real data. To make this comparison we consider the

CTL, ITL and MTL approaches. More specifically, using the nomenclature LX-SVM to refer for either L1, L2 or LS-SVM, we will compare the following:

- CTL-LX: A CTL model based on the LX-SVM.
- ITL-LX: An ITL model based on the LX-SVM.
- MTL-LX: A Convex MTL model based on the LX-SVM, as explained in Section 3.1.
- GLMTL-LX: A Convex GL MTL model based on the LX-SVM, as presented in Section 4.3.
- AdapGLMTL-LX: An Adaptive Convex GL MTL model based on the LX-SVM, as presented in Section 4.4.

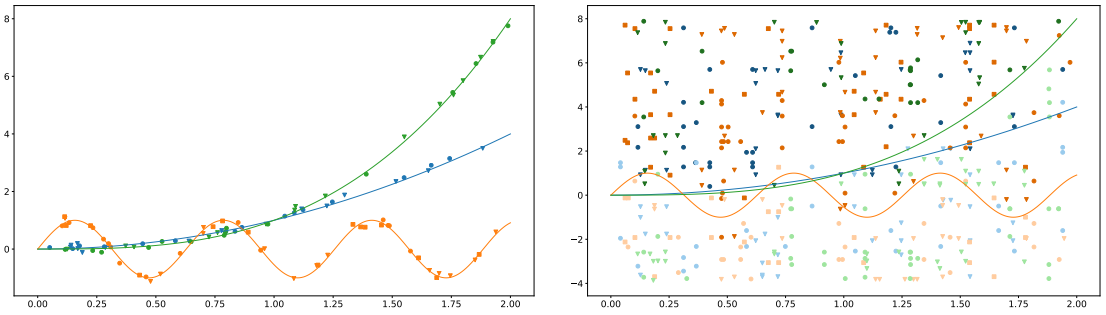
In all cases we use kernel SVMs with Gaussian kernels. We recall that in the adaptive convex GL formulation two possible kernel spaces can be used simultaneously. We conduct experiments with both synthetic and real data, each requiring a different experimental procedure because of their respective computational cost. We first describe the synthetic problems and then those with real data. In each case, we describe the characteristic of the problems, explain the procedure and discuss the corresponding results.

### 5.5.1 Synthetic Problems

We use synthetic problems to generate data where the inter-task relationships are known, so we can check if our proposed model is able to detect this underlying task structure. Also, we generate small training datasets so we can find all the hyperparameters using a grid-search procedure.

Regarding the definition of the synthetic problems, we generate three pairs of related regression and classification problems, namely regClusters0, regClusters1 and regClusters2 for regression, and clasClusters0, clasClusters1 and clasClusters2 for classification. The data in each pair, such as, for example, regClusters0 and clasClusters0, is generated using essentially the same procedure, and the underlying functions in the regression problems are also the decision boundaries in the classification ones. In each problem, we define a number of "underlying" tasks $\tau$, and for $r = 1, \ldots, \tau$ we have a function $f_r$. Then, for each $r$-th "underlying" task, with $r = 1, \ldots, \tau$, we generate $T_r$ "virtual" tasks; although their data is sampled using basically the same function, and models will see them with different task labels, these models should be able to identify their similarities and somehow group them. The total number of tasks in each problem is thus $\sum_{r=1}^{\tau} T_r$, and we denote them as $t_i^r$, where the superindex $r = 1, \ldots, \tau$ correspond to the "underlying" task and the subindex $i = 1, \ldots, T_r$ are the "virtual" tasks or task labels. Therefore, each model receives a total of $\sum_{r=1}^{\tau} T_r$ different task labels. For each task label $t_i^r$, we sample $m_i^r$ points, so the total number of points for each problem is $\sum_{r=1}^{\tau} \sum_{i=1}^{T_r} m_i^r$. More precisely, we proceed as follows:

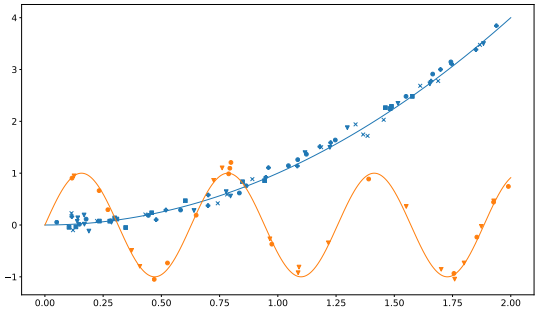
- In problems regClusters0 and clasClusters0, we define $\tau = 3$ "underlying" tasks, defined by the functions $f_1(x) = x^2$, $f_2(x) = \sin(10x)$, $f_3(x) = x^3$. Moreover, we further subdivide them in $T_1 = 2, T_2 = 3$ and $T_3 = 2$ tasks, respectively. The resulting regression dataset is shown in Figure 5.11a, and the classification one in Figure 5.11b.



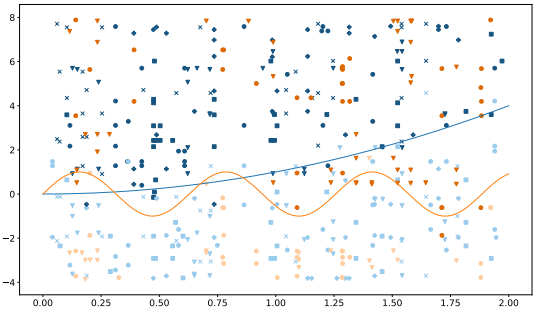
(A) regClusters0.



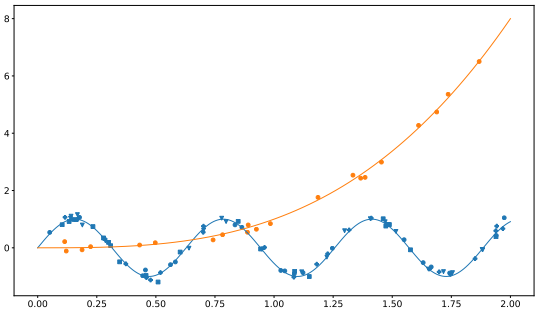
(B) clasClusters0.



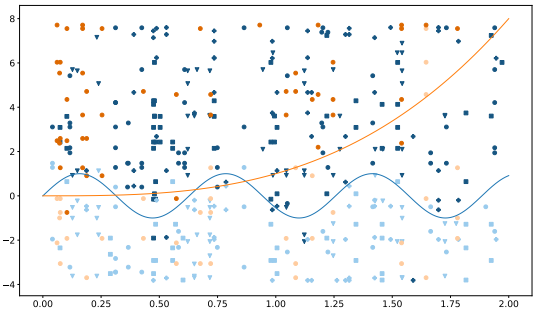
(C) regClusters1.



(D) clasClusters1.



(E) regClusters2



(F) clasClusters2

FIGURE 5.11: Representation of the synthetic problems. We use a different color for each "underlying" task and a different marker for each "virtual" task. In the case of classification, we take two tones for each color: the darker one corresponds to the positive labels, while the lighter one corresponds to the negative labels.

- In problems regClusters1 and clasClusters1 we define $\tau = 2$ "underlying" tasks, defined by the functions $f_1(x) = x^2$, $f_2(x) = \sin(10x)$, and we further subdivide them in $T_1 = 5$ and $T_2 = 2$ tasks, respectively. The resulting regression dataset is shown in Figure 5.11c, and the classification one in Figure 5.11d.

- Finally, in problems regClusters2 and clasClusters2 we define $\tau = 2$ "underlying" tasks in terms of the functions $f_1(x) = \sin(10x)$, $f_2 = x^3$, and we further subdivide them in $T_1 = 4$ and $T_2 = 1$ tasks, respectively. The resulting regression dataset is shown in Figure 5.11e, and the classification one in Figure 5.11f.

In the regression problems we sample 150 values $x_i$ for each task label ("virtual" task) and using the corresponding function we compute the corresponding target values as $t_i = f(x_i) + \epsilon_i$, where $\epsilon_i \sim N(0, 0.1)$. For the classification problems we sample 500 pairs $(x_{i,1}, x_{i,2})$ and get the class values in terms of the corresponding function as

TABLE 5.17: Hyperparameters, grids used to select them (when appropriate) and hyperparameter selection method for each model in the synthetic and real problems. The parameter $\epsilon$ is only suitable for regression with L1 and L2-SVM.

| | Grid | CTL-LX | ITL-LX | MTL-LX | GLMTL-LX | AdapGLMTL-LX |
|---|---|---|---|---|---|---|
| **Synthetic Problems** | | | | | | |
| $C$ | $\left\{10^k : 0 \leq k \leq 5\right\}$ | ✓ | ✓ | ✓ | ✓ | ✓ |
| $\epsilon$ | $\left\{\frac{1}{10^k} : 1 \leq k \leq 3\right\}$ | ✓ | ✓ | ✓ | ✓ | ✓ |
| $\gamma$ | $\left\{\frac{10^k}{d} : 0 \leq k \leq 3\right\}$ | ✓ | ✓ | ✓ | ✓ | ✓ |
| $\lambda$ | $\{0.2k : 0 \leq k \leq 5\}$ | - | - | ✓ | ✓ | ✓ |
| $\nu$ | $\left\{10^k : -2 \leq k \leq 3\right\}$ | - | - | - | ✓ | ✓ |
| $\mu$ | $\left\{10^k : 0 \leq k \leq 5\right\}$ | - | - | - | - | ✓ |
| **Real Problems** | | | | | | |
| $C$ | $\left\{10^k : 0 \leq k \leq 4\right\}$ | ✓ | ✓ | ✓ | - | - |
| $\epsilon$ | $\left\{\frac{1}{10^k} : 1 \leq k \leq 4\right\}$ | ✓ | ✓ | ✓ | - | - |
| $\gamma$ | $\left\{\frac{10^k}{d} : -3 \leq k \leq 2\right\}$ | ✓ | ✓ | ✓ | - | - |
| $\gamma_s$ | $\left\{\frac{10^k}{d} : -3 \leq k \leq 2\right\}$ | - | - | ✓ | - | - |
| $\lambda$ | $\{0.1k : 0 \leq k \leq 10\}$ | - | - | ✓ | - | - |
| $\nu$ | $\{0\} \cup \left\{10^k : -4 \leq k \leq 3\right\}$ | - | - | - | ✓ | ✓ |
| $\mu$ | $\{0\} \cup \left\{10^k : -4 \leq k \leq 3\right\}$ | - | - | - | - | ✓ |

$c_i = \text{sign}(x_{i,2} - f(x_{i,1}) + \epsilon_i)$, where now $\epsilon_i \sim N(0, 1)$. Since we sample 150 pairs for each task, in this problem 300, 450 and 300 patterns come from the underlying tasks 1, 2 and 3, respectively. Therefore, in regClusters0, for example, we have a total of $7 \times 150 = 1050$ pairs $(x_i, t_i)$.

The experimental procedure to test the performance of our proposal is the following one. First, we keep 10 % of the data for training and validation, while the remaining data is used as test set. Thus, in the regression problems we use 15 patterns for train and validation, and in the classification problems we use 50 examples. We remark the small size of the training and validation sets; however this is done consciously; otherwise, if the one-dimensional problems we generate are not challenging enough, we would get perfect score with all the models, which is no useful. We use thus the scarcity of data to make the problems more difficult.

The optimal hyperparameters for each model in each problem are selected using a 5-fold CV grid search; as with the rest of the experiments, the corresponding grids are given Table 5.17. Once these optimal hyperparameters are selected, we use the associated hyperparametrized models and refit them using the entire train and validation set; then we compute the test scores on the test set. For the regression problems we use the MAE as the validation metric, while we use the F1 score for classification ones. To get more stable results, the full procedure, i.e. the problem generation, hyperparameter selection and test scores computation for each model, is repeated 10 times with every problem.

In Table 5.18 we give the test results for the regression problems where we consider the MAE as the metric to measure the performance of the models. Here, we observe that our proposal, AdapGLMTL, gets the best test scores in all problems and for all the variants of the SVM. To better understand these results we can look at the adjacency matrices learned by the AdapGLMTL models, which are depicted in Figure 5.12 for problem regClusters0, in Figure 5.13 for problem regClusters1 and in Figure 5.14 for problem regClusters2. In general, it can be observed that our method detects the underlying task structure and exploits it. See, for instance, the adjacency matrices learned for the problem regCluster0, where there are three distinct clusters, one for each of the three "underlying" tasks of such problem, corresponding to the task-defining functions $f_1(x) = x^2$, $f_2(x) = \sin(10x)$ and $f_3(x) = x^3$. A similar behavior can be found in all the

TABLE 5.18: MAE scores for synthetic regression problems and F1 scores for synthetic classification ones. In bold we highlight the best models of each group: L1, L2 or LS-SVMs.

| | regClusters0 | regClusters1 | regClusters2 | clasClusters0 | clasClusters1 | clasClusters2 |
|---|---|---|---|---|---|---|
| | | MAE | | | F1 | |
| CTL-L1 | 0.989 | 0.512 | 0.541 | 0.901 | 0.912 | 0.904 |
| ITL-L1 | 0.221 | 0.212 | 0.159 | 0.922 | 0.923 | 0.910 |
| MTL-L1 | 0.213 | 0.176 | 0.135 | **0.924** | 0.925 | 0.914 |
| GLMTL-L1 | 0.212 | 0.173 | 0.138 | 0.920 | 0.926 | 0.912 |
| AdapGLMTL-L1 | **0.152** | **0.116** | **0.107** | **0.924** | **0.929** | **0.916** |
| CTL-L2 | 0.990 | 0.642 | 0.768 | 0.904 | 0.912 | 0.906 |
| ITL-L2 | 0.213 | 0.201 | 0.154 | **0.928** | 0.928 | 0.910 |
| MTL-L2 | 0.209 | 0.168 | 0.131 | 0.925 | 0.927 | 0.913 |
| GLMTL-L2 | 0.204 | 0.169 | 0.131 | 0.921 | 0.923 | **0.915** |
| AdapGLMTL-L2 | **0.141** | **0.115** | **0.103** | 0.924 | **0.929** | **0.915** |
| CTL-LS | 0.989 | 0.642 | 0.766 | 0.895 | 0.908 | 0.894 |
| ITL-LS | 0.212 | 0.209 | 0.149 | 0.914 | 0.915 | 0.904 |
| MTL-LS | 0.206 | 0.167 | 0.131 | 0.917 | 0.917 | **0.905** |
| GLMTL-LS | 0.207 | 0.169 | 0.132 | 0.919 | **0.921** | 0.897 |
| AdapGLMTL-LS | **0.136** | **0.115** | **0.106** | **0.920** | **0.921** | 0.901 |

regession problems with all the SVM variants, as shown in Figures 5.13 and 5.14. The improvement on the test scores can then be explained by the implicit data augmentation that takes place when the tasks, corresponding to the same "underlying" task, are grouped and learned together. For example, in problem regClusters1, after the structure of the "underlying" tasks has been identified, our proposal can use $0.1 \times 5 \times 150 = 75$ examples, instead of the 15 that would be considered when each task is to be learned separately.

In the classification problems, the results are not that clear-cut. We give in Table 5.18 the F1 test scores for each model, and, in most cases, the proposed AdapGLMTL gets the best results. However, there are cases where our proposal ties with other approaches, or even where other models obtains the largest F1 score. In Figures 5.15, 5.16 and 5.17, we give the adjacency matrix for problems clasClusters0, clasClusters1 and clasClusters2, respectively. These matrices, although they show that some information about the task structure is inferred from the problems, do not perfectly capture the task relationships, and, therefore, the learning process does not improve task identification that much. This is specially true for the results of the LS-SVM variant, where the matrices learned are very close to the constant matrix, and the results are clearly inferior than those of the L1 or L2-SVMs. One possible reason can be found in the nature of the LS-SVM for classification, which builds a regression function for the positive class and another for the negative one, which usually leads to worse results than those of models designed directly for classification.

### 5.5.2 Real Problems

Apart from the synthetic problems, we also use real data to test our proposal. We consider two regression datasets, computer and parkinson, and a classification dataset, landmine, which has already been described. The computer dataset (Lenk et al., 1996), which has been used in several works (Agarwal et al., 2010; Argyriou et al., 2008; Jeong and Jun, 2018; Kumar and Daumé, 2012), represents the likelihood of purchasing different computer models, as gathered from a survey of 190 people, who consider 13 binary attributes of each computer model. There are 20 different models that the users have

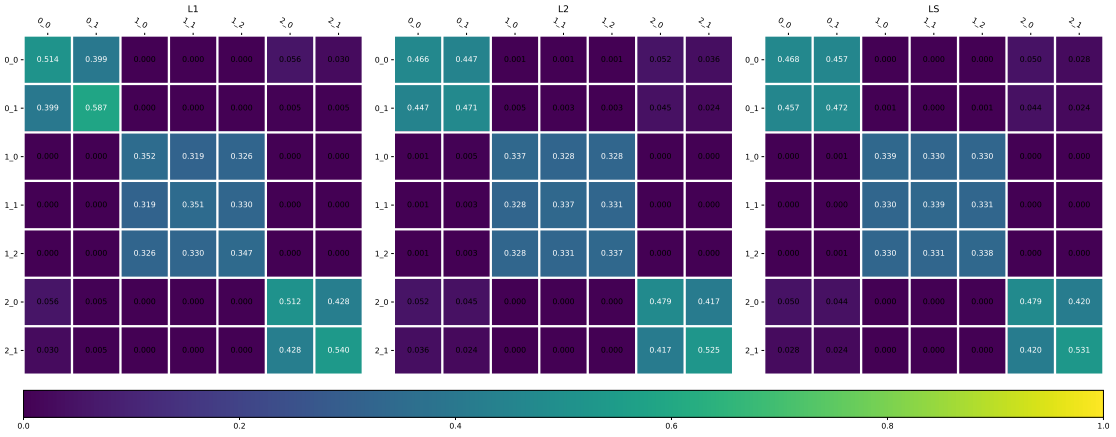



FIGURE 5.12: Adjacency matrices learned for Adaptive GL MTL L1, L2 and LS-SVRs in problem regClusters0.

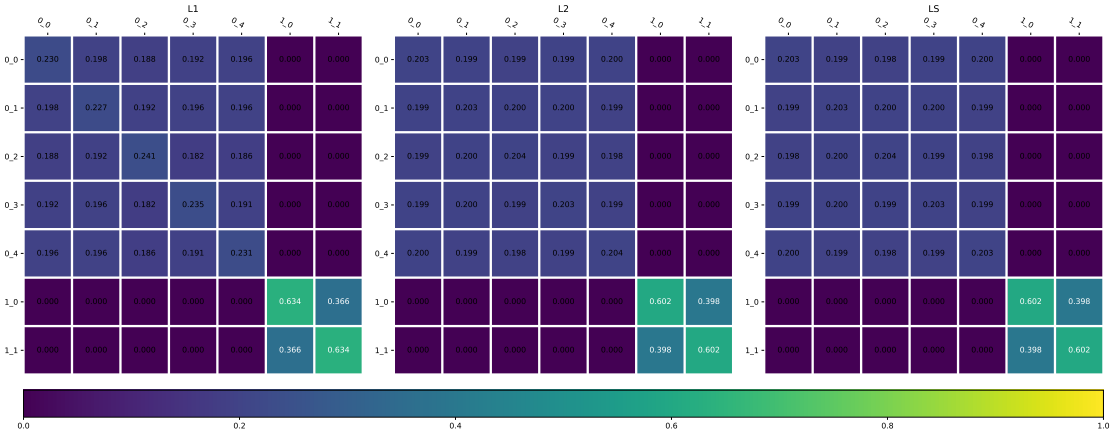



FIGURE 5.13: Adjacency matrices learned for Adaptive GL MTL L1, L2 and LS-SVRs in problem regClusters1.

to evaluate; thus, we have 190 tasks with 20 examples each. The parkinson dataset[10] has data of the voice of 42 people in early stage of Parkinson's disease. They represent each recording with 26 attributes and the target is the UPDRS score, a measurement of the movement, of the patient corresponding to each example. In total, there are 5875 examples, from 42 different tasks. We point out that their size, particular landmine, make them computationally challenging, specially for kernel methods.

Due to the computational limitations, here we consider a different approach for the experimental procedure to the one used for synthetic problems and again we reutilize hyperparameters, as we have done in the others experiments with real data. To compute different test scores for more stable results, we use 5 external folds, where four folds

[10]Available at https://archive.ics.uci.edu/ml/datasets/Parkinsons+Telemonitoring.

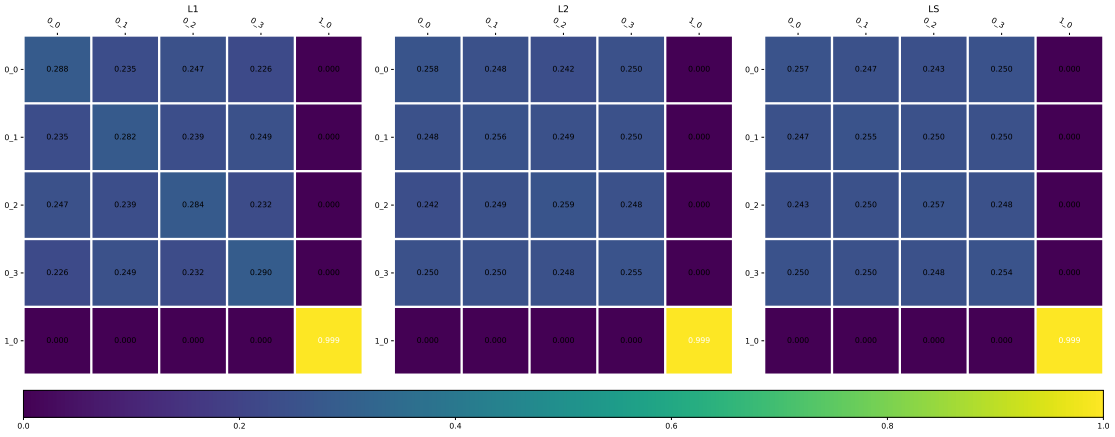



FIGURE 5.14: Adjacency matrices learned for Adaptive GL MTL L1, L2 and LS-SVRs in problem regClusters2.

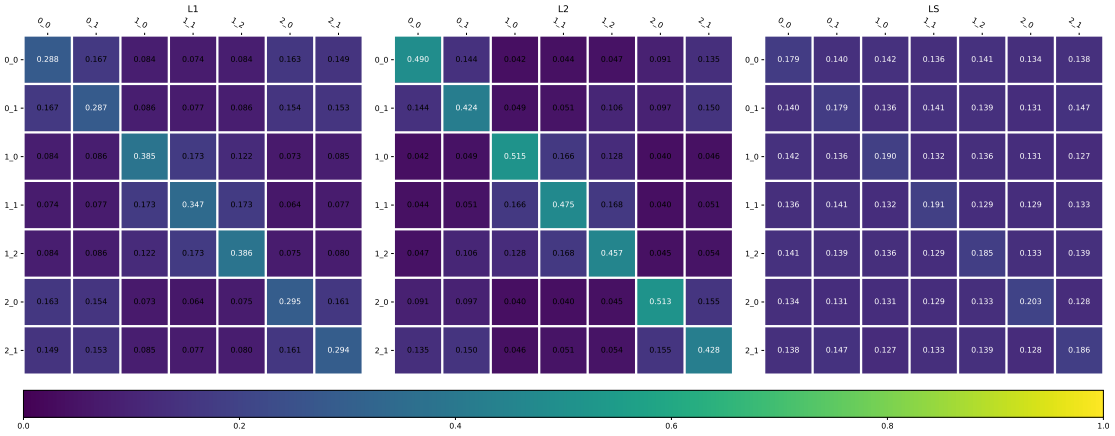



FIGURE 5.15: Adjacency matrices learned for Adaptive GL MTL L1, L2 and LS-SVCs in problem clasClusters0.

are used for train and validation and the remaining one as test set. That is, we get 5 different test scores and average them for a better estimation of the performance of the models. For searching the optimal hyperparameters, with each outer train-validation set we perform a randomized search with an internal 5-fold CV. That is, we define a discrete grid, for each hyperparameter and then a maximum of 50 points over the entire combined grid are evaluated on the inner train-validation folds. We select the hyperparameter combination with the best average validation score, and the corresponding model is refitted on the entire outer train-validation set, and then applied to the outer test fold to compute the test score. In Table 5.19 we give the averages of the 5 test scores that we obtain with this procedure. We remark that the number of hyperparameters in the GLMTL and AdapGLMTL models, which include the GL regularization, is 5 and 6, which

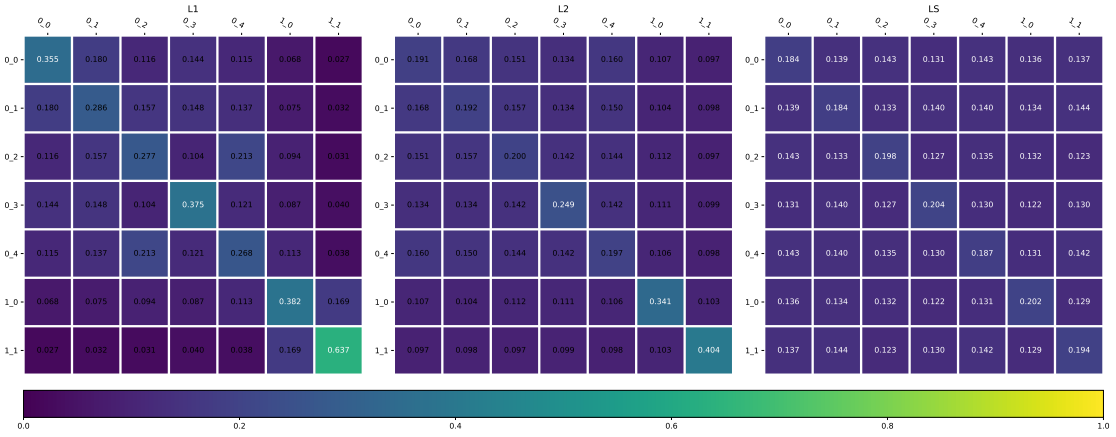



FIGURE 5.16: Adjacency matrices learned for Adaptive GL MTL L1, L2 and LS-SVCs in problem clasClusters1.

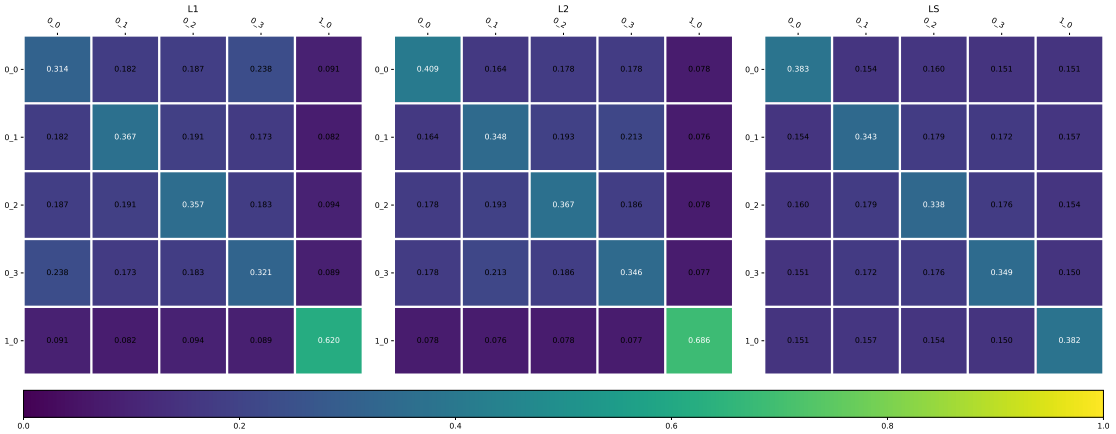



FIGURE 5.17: Adjacency matrices learned for Adaptive GL MTL L1, L2 and LS-SVCs in problem clasClusters2.

define spaces too large to be covered with only 50 points. To alleviate this, we first pre-select the hyperparameters $C, \epsilon, \gamma, \gamma_s$ with those values that are optimal for the MTL model. Recall that we are using two different kernel widths, as discussed on the GL MTL Section in Chapter 4. In Table 5.17 we present the grids considered for the CV search in these experiments, and we also indicate which parameters are included in this random search for each model. Here, observe that we include the values $\mu = 0$ and $\nu = 0$, which ensures that the the GLMTL model contains the MTL one, with $\nu = 0$, and also AdapGLMTL contains GLMTL with an ientity matrix (minimal entropy rows) when $\mu = 0$.

In Table 5.19 of the appendix we show the test results for the real problems, where we report the MAE for the regression problems and the F1 score for landmine, the classification one. In computer, our proposal, AdapGLMTL obtains different results

TABLE 5.19: Test results for real problems. We show the average and standard deviation of MAE scores for the regression problems: computer and parkinson; and average and standard deviation of F1 scores for the classification problem: landmine. In bold we highlight the best models of each group: L1, L2 or LS-SVMs.

| | computer | parkinson | landmine |
|---|---|---|---|
| CTL-L1 | $1.985 \pm 0.039$ | $4.306 \pm 0.209$ | $0.202 \pm 0.047$ |
| ITL-L1 | $1.623 \pm 0.028$ | $1.949 \pm 0.021$ | $0.243 \pm 0.014$ |
| MTL-L1 | $\mathbf{1.526 \pm 0.052}$ | $\mathbf{1.942 \pm 0.019}$ | $0.266 \pm 0.022$ |
| GLMTL-L1 | $\mathbf{1.526 \pm 0.052}$ | $1.945 \pm 0.024$ | $0.249 \pm 0.039$ |
| AdapGLMTL-L1 | $1.545 \pm 0.083$ | $1.945 \pm 0.024$ | $\mathbf{0.272 \pm 0.033}$ |
| CTL-L2 | $2.001 \pm 0.028$ | $4.211 \pm 0.093$ | $0.235 \pm 0.045$ |
| ITL-L2 | $2.105 \pm 0.070$ | $1.936 \pm 0.026$ | $\mathbf{0.267 \pm 0.019}$ |
| MTL-L2 | $1.506 \pm 0.043$ | $1.928 \pm 0.037$ | $0.260 \pm 0.041$ |
| GLMTL-L2 | $1.507 \pm 0.045$ | $\mathbf{1.927 \pm 0.037}$ | $0.251 \pm 0.048$ |
| AdapGLMTL-L2 | $\mathbf{1.501 \pm 0.047}$ | $1.928 \pm 0.038$ | $0.249 \pm 0.055$ |
| CTL-LS | $2.002 \pm 0.029$ | $4.217 \pm 0.108$ | $0.186 \pm 0.040$ |
| ITL-LS | $1.609 \pm 0.015$ | $1.928 \pm 0.017$ | $0.182 \pm 0.007$ |
| MTL-LS | $1.507 \pm 0.042$ | $1.929 \pm 0.026$ | $0.186 \pm 0.041$ |
| GLMTL-LS | $\mathbf{1.502 \pm 0.047}$ | $\mathbf{1.917 \pm 0.022}$ | $\mathbf{0.197 \pm 0.033}$ |
| AdapGLMTL-LS | $1.506 \pm 0.046$ | $1.924 \pm 0.033$ | $0.196 \pm 0.031$ |

depending on the SVM variant. For L2-SVM, it gets the best result; however, in the L1 and LS variants, although it has the best validation scores, this does not generalize well to the test set, where the non-adaptive GLMTL approach has the best results. In summary, in computer, the proposed AdapGLMTL is best overall for the L2 variant, essentially ties with MTL for second place in the LS variant, and is third for the L1-SVM. In the parkinson problem, our proposal is very close to the best models for the L1 and L2 variants, and gets the second best result in the L2-SVM. Finally, in the landmine, the classification problem, AdapGLMTL is the best approach for L1, essentially ties with the non-adaptive version for the first place for LS, but it is fourth for the L2-SVM. In any case, we can highlight some general patterns: the MT models get the best results in all cases except in landmine for the L2 variant, the GL approaches either tie or surpass the MTL approach in all problems, and AdapGLMTL have the best score in two occasions, while being competitive in almost all others.

Now, we also point out the difference in performance of the AdapGLMTL between the real and synthetic problems. We can try to find an explanation for these different behaviors. Note that our adaptive proposal relies on pushing together tasks whose model parameters are close. In the synthetic problems, the clusters of tasks are well-defined and, thus, smart-grouping lead to better results. However, in the real problems this is not the case. The task structure is not clear-cut, and even more, the information that is inferred from the training set might not reflect the real underlying structure.

## 5.6 Conclusions

In this Chapter we have presented several experiments to test our MTL proposals. We test the performance of the convex formulations and GL-based strategies, presented in the previus chapters, on multiple classificationa and regression problems.

In Chapter 3 we have described the convex MTL approach, where a common and task-specific parts are combined. Instead of adding the common and the corresponding task-specific part, which we call the additive formulation, we use a convex combination of the two parts. In Section 5.1, using models based on the L1-SVM, we have shown empirically that the convex MTL formulation is equivalent to the additive one. Next, we have extended this convex MTL approach in two main ways. We have applied it

to the three SVM varians, L1, L2 and LS-SVM, and we have also given numerical results, with multiple real datasets, that show that our proposed methods outperforms the corresponding CTL and ITL strategies.

Although the convex MTL formulation with kernel methods gives very competitive results, these models are not the most adequate for other type of data, such as images. To tackle the challenges that non-tabular data brings, the NNs are models that are better prepared, so we have presented in Chapter 3 an extension of the convex MTL for NNs. Regarding this Chapter of experiments, in Section 5.3 we provide experiments with four different image datasets to test the convex MTL approach with NNs. In these results we have observed that our proposal obtains better results than using a common NN or considering task-specific ones. Also, in our experiments, the convex approach surpasses the models using a hard sharing strategy, with shared hidden layers and task-specific output ones, which is the most natural approximation for MTL with NNs.

Furthermore, in Chapter 4 we have proposed an MTL strategy based on the GL formulation, which we combine with the convex approach; we apply this formulation, which we call convex GL, to kernel methods, such as the L1, L2 and LS-SVM. Recall that the GL formulation imposes a regularization that penalizes the pairwise distances between tasks, weighted by a given adjacency matrix. In Section 5.4 we have compared the convex GL with the baselines CTL and ITL approaches, as well as with the convex models. We have observed that, with an appropiate selection of hyperparameters, the convex GL methods with a fixed adjacency matrix can get good results, and outperform in many cases the convex ones. We remark that the GL approach can obtain the maximum leverage when an appropiate adjacency matrix is taken. In Chapter 4 we have provided a procedure to automatically learn this matrix from the data, this is the adaptive GL algorithm. To test this algorithm we have provided in Section 5.5 several synthetic problems, both in a regression and classification setting, to grasp a better understanding of such algorithm. We have observed that in the synthetic problems, where the task relations are well defined, the adaptive GL methods are superior to non-adaptive ones. Here, the adjacency matrices learned by our proposals usually reflect the real underlying relations between tasks, and it provide an insightful information about the task structure. Moreover, we see that our proposal is also useful with real problems, where the results are competitive even where the task relations are not that clear-cut.

In summary, our proposals, based on the convex combination of common and task-specific parts, and the application of the GL regularization, yield good results over several datasets. We can observe that the convex formulation is quite general, and with it we can define a broad range of models, from kernel methods such as multiple SVM variants to NNs. Therefore, we can select the model that is better suited for our problem at hand and apply a convex MTL strategy. The GL regularizer has been developed in this work for kernel models, and, when a fixed adjacency matrix is considered, it can be interpreted as a complement to our convex MTL formulation. The convex approximation and the convex GL one can be equivalent under an adequate selection of hyperparameters and some restrictions; we need to restrict the independent models to a common space, which results in less expressive models but also in less hyperparameters to select. We have observed that the interesting part of the GL regularization is that we can learn the task relationship and exploit this information to get better results. There is, thus, a tradeoff between the convex and convex GL approaches; in the first one we need to make a computational effort to select the most adequate hyperparameters and task-specific spaces (especially with kernel methods), and in the second one this effort must be made to learn the task relations. Therefore, depending on the problem, we should select one

or the other: if we believe that the tasks are very different we can use the full convex approach with all its hyperparameters, and, if we believe that the tasks are closely related but not in a trivial way (all share a common part), we should use the convex GL models to control the task dependencies.

Chapter 6

# Conclusions And Future Work

## 6.1 Conclusions

Machine Learning (ML) has the goal of automatizing the learning process so machines can be programmed to learn. For this purpose, several approaches have been considered, but all of them ultimately try to estimate some function from data inductively, i.e. from particular realizations of some process, ML methods try to infer the underlying function generating this data. Some of the most relevant ML methods are kernel methods (such as the Support Vector Machine (SVM)) and the NN.

Multi-Task Learning (MTL) is a field of ML that considers learning multiple tasks jointly with the goal of improving the learning process, that is, multiple functions are to be estimated and the goal is to obtain leverage by learning them together. The trivial approximations for these scenarios would be Common-Task Learning (CTL), where a common model is applied for all tasks, and Independent-Task Learning (ITL), where task-specific models are considered, without any relation between them. However, MTL considers another way, where there is one model for each task, but there exist a connection between them.

For the goals of MTL, several strategies have been considered; in this work, in Chapter 2, we have categorized them in three groups: feature-based, parameter-based and regularization-based. In short, feature-based methods try to find a shared representation of the data that is useful for all tasks, parameter-based methods try to couple different tasks using regularization schemes, and combination-based ones combine a common and task-specific parts. Different ML methods, based on their characteristics, are more easily adaptable to one strategy or another, as described in Chapter 2. In particular, Neural Networks (NNs) are most compatible with feature-based approaches: a very natural way to implement MTL is to define a network with shared hidden layers and task-specific output neurons. Other example is linear models, which are also compatible with feature-based strategies; however, the parameter-based ones are more interesting in this case, where different properties can be enforced on the linear models (such as different coupling schemes, sparsity...) through specialized regularizers. Looking at the kernel methods, although they bring desirable properties, they present a more rigid framework to adapt MTL strategies for them. Since the features depend on the choice of the kernel, feature-based strategies cannot be easily applied. Moreover, the parameters of these models are elements of a Reproducing Kernel Hilbert Space (RKHS), so designing specialized regularizers for them is more difficult; however, combination-based strategies

fit well with kernel methods. In this work we develop novel MTL strategies for kernel methods, which are also applicable to other models such as NNs.

First, we describe a convex combination approach in Chapter 3. One of the main challenges of combination-based strategies is selecting the optimal hyperparameters for our models, where it is necessary to select the weight of the common and task-specific part in each model. In particular, we propose a convex formulation, where the combination parameters $\lambda_r \in [0, 1]$ are task-specific, have a clear interpretation and can recover the trivial CTL and ITL cases. For a deeper understanding, we develop this convex combination approach for three SVM variants: L1, L2 and LS-SVM; we also discuss its properties both in the dual and primal formulations. In addition, we consider a natural alternative, which applies the convex combination of pre-trained common and task-specific models. We describe how the optimal combination parameters can be obtained in this case. In the experiments of Chapter 5 we show how, in most cases, this convex approach outperforms the CTL and ITL strategies, as well as the convex combination of pre-trained models. The experiments concerning energy prediction are specially relevant; here, we observe that defining tasks in our data to build more specialized models can improve our results.

Furthermore, this convex formulation is a general one, and it is applicable to other models; to the best of our knowledge, we propose in this work the first combination-based MTL strategy for NNs. In Chapter 5, using image datasets, we observe how this approach gets the best result when compared with its CTL and ITL counterparts, and also when compared with the commonly used approach of sharing weights in the hidden layers.

As we have already seen, parameter-based strategies are more difficult to implement for kernel methods; however, in Chapter 4 we propose a formulation based on tensor products of Reproducing Kernel Hilbert Spaces (RKHSs) that enables us to apply some of these strategies. As a result of this formulation, we observe that we can define kernels that split the task and feature information in two parts, which are multiplied.

In particular, taking this tensor-based formulation, in Chapter 4 we develop a Graph Laplacian (GL) approach for the L1, L2 and LS-SVM. In this strategy, the tasks are seen as nodes in a graph, and the pairwise distance between task parameters are penalized, where the graph adjacency matrix is used to weight these distances. Our formulation enables us to write a kernel that splits the task relatedness, as indicated in the adjacency matrix, and the features similarities. Additionally, we combine the GL approach with the convex formulation, where we use a convex part and task-specific parts coupled through a graph regularization. The results of our experiments in Chapter 5 show that this convex GL strategy is an interesting alternative to the plain convex formulation, each capturing different properties of the problems. We observe that the choice of the best strategy depends on the characteristics of each problem.

As we have explained, the GL models strongly depend on the selection of an adjacency matrix, so, in Chapter 4, we develop an algorithm to automatically learn this matrix from the data. We provide multiple experimental results in Chapter 5 that show how this approach enables us to learn the task relations and, in consequence, improve obtain better models.

## 6.2 Future Work

Our proposed methods show how we can adapt different MTL strategies principally to kernel methods, but also to other model such as NNs. In this work, we have proposed a general convex formulation for MTL, which we apply on kernel methods and NNs.

Also, we have presented a GL approach that we have applied to the L1, L2 and LS-SVM. These proposals offer the possibility of new extensions, and they also bring challenges for which we can develop novel solutions. Precisely, to conclude this work, we describe some of the ideas that can be studied as future work.

One of the main difficulties of using the combination-based MTL strategies with kernel methods is the large number of hyperparameters that have to be selected. Recall that, with these approaches, it is possible to apply a common kernel and task-specific ones, which can all be different. In addition, we have other hyperparameters, such as the regularization ones or, in our convex formulation, those that govern the convex combinations. Therefore, to fully exploit the expressive power of these models we would like to select the optimal values for each one of these hyperparameters. Standard methods, such as using a grid search, are not feasible for this kind of models; however, we can consider alternative strategies, such as Bayesian Optimization or Simulated Annealing, to fully hyperparametrize these models. In the case of kernel hyperparameters, such as the kernel width in the Gaussian kernel, strategies related to kernel learning can be applied and extended for the MTL case. In our proposal, the convex MTL formulation, the selection of the convex combination parameters is particularly relevant. One approximation is to act with them as with the rest of hyperparameters and rely on general strategies (such as Bayesian Optimization), but we can also develop specific methods that learn these parameters from data, similarly to what we have done with the adjacency matrix of the GL proposal. Furthermore, in the case of the convex MTL NN it is not necessary to develop any novel algorithm, we can treat these convex combination hyperparameters as parameters of the network and optimize them using standard gradient descent techniques.

Shifting our attention to the GL approaches, a possible extension of our work would be to consider applying the GL regularization to NNs. The regularization penalizing the distances between the weights of the task-specific networks (or subnetworks) is differentiable, so gradient descent would be a valid technique to optimize these models.

Appendix A

# List of Publications.

This appendix contains a relation of the articles published during the development of this thesis. The full list of journal articles is:

1. **Adaptive Graph Laplacian MTL L1, L2 and LS-SVMs.**
   Ruiz, C., Alaíz, C. & Dorronsoro, J.. *Logic Journal of the IGPL*. 2022.
   Under review.
   SJR Rank: Q1

2. **Multitask SVR Models for Solar and Wind Energy.**
   Ruiz, C., Alaíz, C. & Dorronsoro, J.. *Energies*. 2020.
   DOI: 10.3390/en13236308
   SJR Rank: Q3

3. **Convex Formulation for Multi-Task L1-, L2-, and LS-SVMs.**
   Ruiz, C., Alaíz, C. & Dorronsoro, J.. *Neurocomputing*. 2020.
   DOI: 10.1016/j.neucom.2021.01.137
   SJR Rank: Q2

The full list of conference articles is:

1. **Convex Multi-Task Learning with Neural Networks.**
   Ruiz, C., Alaíz, C. & Dorronsoro, J.. *Hybrid Artificial Intelligent Systems*. 2022.
   DOI: 10.1007/978-3-031-15471-3_20
   Core Rank: National

2. **Adaptive Graph Laplacian for Convex Multi-Task Learning SVM.**
   Ruiz, C., Alaíz, C. & Dorronsoro, J.. *Hybrid Artificial Intelligent Systems*. 2021.
   DOI: 10.1007/978-3-030-86271-8_19
   Core Rank: National

3. **Convex Graph Laplacian Multi-Task Learning SVM.** Ruiz, C., Alaíz, C. & Dorronsoro, J.. *International Conference on Artificial Neural Networks*. 2020.
   DOI: 10.1007/978-3-030-61616-8_12
   Core Rank: A

4. **A Convex Formulation of SVM-based Multi-Task Learning.**
   Ruiz, C., Alaíz, C. & Dorronsoro, J.. *Hybrid Artificial Intelligent Systems*. 2019.
   DOI: 10.1007/978-3-030-29859-3_35
   Core Rank: C

5. **Flexible Kernel Selection in Multitask Support Vector Regression.**
   Ruiz, C., Catalina, A., Alaíz, C. & Dorronsoro, J.. *International Joint Conference on Neural Networks*. 2019.
   DOI: 10.1109/IJCNN.2019.8852297
   Core Rank: A

Appendix B

# Additional Material

## B.1 Vector-Valued Reproducing Kernel Hilbert Spaces

To give different definitions of vector-valued kernels, we will follow the work of Micchelli and Pontil (2005). 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

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

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.

The first approach considered is that of 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

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

If these functionals are continuous, we can apply Riesz representation theorem (Riesz and Nagy, 2012). 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 $f \in \mathcal{H}$,

$$L_{x,y} f = [y, f(x)]_{\mathcal{Y}} = \langle g_{x,y}, f \rangle_{\mathcal{H}} . \tag{B.1}$$

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

**Definition B.1** (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

$$\begin{array}{rccc} g_x: & \mathcal{Y} & \to & \mathcal{H} \\ & y & \to & g_x y = g_{x,y} \end{array} . \tag{B.2}$$

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

$$\begin{array}{rccc} K(x,\hat{x}): & \mathcal{Y} & \to & \mathcal{Y} \\ & y & \to & K(x,\hat{x})y = (g_{\hat{x}}y)(x) \end{array} \tag{B.3}$$

for every $x, \hat{x} \in \mathcal{X}$. It is possible then to prove that $K(x,\hat{x})$ is a reproducing kernel in $\mathcal{H}$ as seen in Micchelli and Pontil (2005, Propositon 2.1).

**Proposition B.2.** *If $K(x,\tilde{x})$ is defined for every $x, \hat{x} \in \mathcal{X}$ as in (B.3) and $g_x$ is defined as in (B.2), then*

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

*is a kernel that for every $x, \hat{x} \in \mathcal{X}$ satisfies:*

1. *For every $y, \hat{y} \in \mathcal{Y}$, we have*

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

2. *$K(x,\tilde{x}) = K(\tilde{x},x)^{\#}$ and $K(x,x)$ is positive definite for every $x \in \mathcal{X}$.*

3. *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. *$\|g_x\| = \|K(x,x)\|^{\frac{1}{2}}$.*

5. *$\|K(x,t)\| \leq \|K(x,x)\|^{\frac{1}{2}} \|K(t,t)\|^{\frac{1}{2}}$.*

6. *For every $f \in \mathcal{H}$ and $x \in \mathcal{X}$, we have that*

$$\|f(x)\| \leq \|f\| \|K(x,x)\|^{\frac{1}{2}} .$$

The second approach of Micchelli and Pontil (2005) to define vector-valued kernels 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 Theorem:

**Theorem B.3.** *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 Theorem, considering the space of the completion of the span of $\{K_x = K(\cdot,x), x \in \mathcal{X}\}$.

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

**Lemma B.4.** *Given some Hilbert space $\mathcal{W}$, any continuous feature map $\Phi : \mathcal{X} \to \mathcal{L}(\mathcal{W}, \mathcal{Y})$ defines a kernel as*

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

*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 $x_1, \ldots, x_n \in \mathcal{X}, y_1, \ldots, y_n \in \mathcal{Y}$

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

□

These definitions related to vector-valued RKHSs are useful to get a better understanding of how to apply operator-valued kernels for learning multi-target or multi-task functions. Concretely, we would like to develop some result similar to representer theorem Schölkopf et al. (2001), which is a crucial result in optimization and ML. 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 B.5.** *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*

$$E(f) = V((f(x_1), \ldots, f(x_n)), \|f\|^2)$$

*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 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 B.6.** *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

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

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

## B.2 Examples of MTL Kernels

Using the framework for MTL with kernel methods just discussed we can choose different regularizations, induced by the matrix $E$, which lead to different MT approaches.

One trivial example that we can illustrate using this framework is that of independent tasks. This is the case when $E = I_T$ and therefore $B = I_T$ , that is

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

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.

Other important example that we can model using this framework is that of a combination-based MTL, where we use the combination of a common part and task-specific parts. For this case, we select the matrix $B$ such that its columns are

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

the corresponding multi-task kernel is:

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

This is equivalent to the approach presented in the work of Evgeniou and Pontil (2004), that we have presented in (3.3). Here, for each task we use the vector

$$w_r = w + v_r,$$

to define the model of the $r - th$ task, where $w$ is common to all tasks and $v_r$ is task-specific.

Moreover, in Evgeniou and Pontil (2004) it is shown that this is a problem equivalent to

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

Now, only the $w_r$ variables are included, and it is clear that $\mu$ penalizes the variance of the $w_r$ vectors, so all models $w_r$ will tend to a common model as $\mu$ 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+ of the LUPI approach from Vapnik and Izmailov (2015).

Finally, we can take the case where 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. Here, the matrix $E$ can be chosen as a Laplacian matrix $L = D - A$, where $A$ 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

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

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

Appendix C

# Multi-Task Datasets

In this appendix, we present a table with some of the most commonly used Multi-Task Learning (MTL) datasets. In Table C.1 we give the number of samples, dimension and number of tasks of each dataset.

TABLE C.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.

| Name | $n$ | $d$ | $T$ | Nature | References |
|---|---|---|---|---|---|
| school | 15 362 | 27 | 139 | MT single-reg | Evgeniou and Pontil (2004)<br>Evgeniou et al. (2005)<br>Argyriou et al. (2006, 2008, 2007)<br>Bonilla et al. (2007)<br>Zhang and Yeung (2010)<br>Agarwal et al. (2010)<br>Chen et al. (2011)<br>Zhou et al. (2011)<br>Gong et al. (2012b)<br>Kumar and Daumé (2012)<br>Zhang and Yeung (2013)<br>Han and Zhang (2016)<br>Jeong and Jun (2018) |
| 20-newsgroup | 18 000 | t.v. | 20 | multi-clas | Ando and Zhang (2005)<br>Daumé (2009) |
| Reuters-RCV1 | 800 000 | t.v. | 103 | multi-clas | Yu et al. (2005)<br>Ando and Zhang (2005) |
| computer | 3600 | 13 | 180 | MT single-reg | Argyriou et al. (2006, 2008, 2007)<br>Evgeniou et al. (2005)<br>Agarwal et al. (2010)<br>Kumar and Daumé (2012)<br>Jeong and Jun (2018) |
| landmine | 14 820 | 10 | 29 | MT bin-clas | Xue et al. (2007)<br>Jebara (2011)<br>Daumé (2009)<br>Jawanpuria and Nath (2012) |
| MHC-I | 32 302 | 184 | 47 | MT bin-clas | Jacob et al. (2008)<br>Jawanpuria and Nath (2012) |
| dermatology | 366 | 33 | 6 | multi-clas | Jebara (2004)<br>Argyriou et al. (2008) |
| sentiment | 2000 | 473856 | 4 | MT bin-clas | Daumé (2009)<br>Zhang and Yeung (2010)<br>Crammer and Mansour (2012)<br>Zhang and Yeung (2013) |

Continued on next page

**Table C.1 – continued from previous page**

| Name | $n$ | $d$ | $T$ | Nature | References |
|---|---|---|---|---|---|
| | | | | | Barzilai and Crammer (2015) |
| sarcos | 44 484 | 21 | 7 | multi-reg | Zhang and Yeung (2010)<br>Chen et al. (2011)<br>Zhou et al. (2011)<br>Jawanpuria and Nath (2012)<br>Zhang and Yeung (2013)<br>Ciliberto et al. (2015) |
| isolet | 7797 | 617 | 5 | MT multi-clas | Parameswaran and Weinberger (2010)<br>Gong et al. (2012a) |
| mnist | 70 000 | 400 | 10 | multi-clas | Kang et al. (2011)<br>Kumar and Daumé (2012)<br>Zweig and Weinshall (2013)<br>Jeong and Jun (2018) |
| usps | 9298 | 256 | 10 | multi-clas | Kang et al. (2011)<br>Kumar and Daumé (2012)<br>Zweig and Weinshall (2013)<br>Jeong and Jun (2018) |
| adni | 675 | 306 | 6 | MT single-reg | Gong et al. (2012a)<br>Gong et al. (2012b) |
| microarray | 131 | 21 | 19 | multi-reg | Lozano and Swirszcz (2012)<br>Han and Zhang (2016) |
| cifar10 | 50 000 | 1024 | 10 | multi-clas | Zweig and Weinshall (2013)<br>Han and Zhang (2016) |
| parkinson | 5875 | 19 | 42 | MT single-reg | Jawanpuria and Nath (2012)<br>Jeong and Jun (2018) |

# Bibliography


Agarwal, A., Daumé, 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.

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.

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.

Chang, C. and Lin, C. (2011). LIBSVM: A library for support vector machines. *ACM Trans. Intell. Syst. Technol.*, 2(3):27:1–27:27.

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

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.

Chernoff, H. (1952). A measure of asymptotic efficiency for tests of a hypothesis based on the sum of observations. *The Annals of Mathematical Statistics*, 23(4):493–507.

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.

Cristianini, N. and Shawe-Taylor, J. (2000). *An Introduction to Support Vector Machines and Other Kernel-based Learning Methods*. Cambridge University Press.

Daumé, 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, 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.

Goodfellow, I. J., Pouget-Abadie, J., Mirza, M., Xu, B., Warde-Farley, D., Ozair, S., Courville, A. C., and Bengio, Y. (2014). In *Advances in Neural Information Processing Systems 27: Annual Conference on Neural Information Processing Systems 2014, December 8-13 2014, Montreal, Quebec, Canada*, pages 2672–2680.

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.

Hoeffding, W. (1963). Probability inequalities for sums of bounded random variables. *Journal of the American Statistical Association*, 58(301):13–30.

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.

Kadison, R. V. and Ringrose, J. R. (1983). In *Fundamentals of the Theory of Operator Algebras*, volume 100 of *Pure and Applied Mathematics*. Elsevier.

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.

Keerthi, S. S., Shevade, S. K., Bhattacharyya, C., and Murthy, K. R. K. (2001). Improvements to platt's SMO algorithm for SVM classifier design. *Neural Comput.*, 13(3):637–649.

Kelley, J. L. (2017). *General topology*. Courier Dover Publications.

Kingma, D. P. and Ba, J. (2015). Adam: A method for stochastic optimization. In Bengio, Y. and LeCun, Y., editors, *3rd International Conference on Learning Representations, ICLR 2015, San Diego, CA, USA, May 7-9, 2015, Conference Track Proceedings*.

Krizhevsky, A., Sutskever, I., and Hinton, G. E. (2012). Imagenet classification with deep convolutional neural networks. In *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 1106–1114.

Kumar, A. and Daumé, 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.

Lenk, P., Desarbo, W., Green, P., and Young, M. (1996). Hierarchical bayes conjoint analysis: Recovery of partworth heterogeneity from reduced experimental designs. *Marketing Science*, 15:173–191.

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.

Lin, C. (2001). On the convergence of the decomposition method for support vector machines. *IEEE Trans. Neural Networks*, 12(6):1288–1298.

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. (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.

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.

Pedregosa, F., Varoquaux, G., Gramfort, A., Michel, V., Thirion, B., Grisel, O., Blondel, M., Prettenhofer, P., Weiss, R., Dubourg, V., Vanderplas, J., Passos, A., Cournapeau, D., Brucher, M., Perrot, M., and Duchesnay, E. (2011). Scikit-learn: Machine learning in Python. *Journal of Machine Learning Research*, 12:2825–2830.

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.

Riesz, F. and Nagy, B. (2012). *Functional Analysis*. Dover Books on Mathematics. Dover Publications.

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. (2020a). Convex graph laplacian multi-task learning SVM. In *Artificial Neural Networks and Machine Learning - ICANN 2020 - 29th International Conference on Artificial Neural Networks, Bratislava, Slovakia, September 15-18, 2020, Proceedings, Part II*, volume 12397 of *Lecture Notes in Computer Science*, pages 142–154. 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*.

Ruiz, C., Alaíz, C. M., and Dorronsoro, J. R. (2020b). Multitask support vector regression for solar and wind energy prediction. *Energies*, 13(23).

Schölkopf, B., Herbrich, R., and Smola, A. J. (2001). A generalized representer 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.

Schölkopf, B. and Smola, A. J. (2002). *Learning with Kernels: support vector machines, regularization, optimization, and beyond*. Adaptive computation and machine learning series. MIT Press.

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.

Vaswani, A., Shazeer, N., Parmar, N., Uszkoreit, J., Jones, L., Gomez, A. N., Kaiser, L., and Polosukhin, I. (2017). Attention is all you need. In *Advances in Neural Information Processing Systems 30: Annual Conference on Neural Information Processing Systems 2017, December 4-9, 2017, Long Beach, CA, USA*, pages 5998–6008.

Weinberger, K. Q. and Saul, L. K. (2009). Distance metric learning for large margin nearest neighbor classification. *J. Mach. Learn. Res.*, 10:207–244.

Whittaker, D. J. (1991). A course in functional analysis. *The Mathematical Gazette*, 75:488 – 489.

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 Yang, Q. (2017). A survey on multi-task learning. *CoRR*, abs/1707.08114.

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.