Neurocomputing 456 (2021) 599–608

Contents lists available at ScienceDirect

Neurocomputing

journal homepage: www.elsevier.com/locate/neucom

Convex formulation for multi-task L1-, L2-, and LS-SVMs

Carlos Ruiz [a,*], Carlos M. Alaíz [a], José R. Dorronsoro [a,b]

[a] Dpto. Ing. Informática, Universidad Autónoma de Madrid, Madrid 28049, Spain
[b] Inst. de Ing. del Conocimiento, Campus Cantoblanco UAM, Madrid 28049, Spain

ARTICLE INFO

Article history:
Received 20 March 2020
Revised 17 December 2020
Accepted 5 January 2021
Available online 17 June 2021

2010 MSC:
00-01
99-00

Keywords:
Multi-task learning
L1-SVM
L2-SVM
LS-SVM
Convex model combination

ABSTRACT

Quite often a machine learning problem lends itself to be split in several well-defined subproblems, or tasks. The goal of Multi-Task Learning (MTL) is to leverage the joint learning of the problem from two different perspectives: on the one hand, a single, overall model, and on the other hand task-specific models. In this way, the found solution by MTL may be better than those of either the common or the task-specific models. Starting with the work of Evgeniou et al., support vector machines (SVMs) have lent themselves naturally to this approach. This paper proposes a convex formulation of MTL for the L1-, L2- and LS-SVM models that results in dual problems quite similar to the single-task ones, but with multi-task kernels; in turn, this makes possible to train the convex MTL models using standard solvers. As an alternative approach, the direct optimal combination of the already trained common and task-specific models can also be considered. In this paper, a procedure to compute the optimal combining parameter with respect to four different error functions is derived. As shown experimentally, the proposed convex MTL approach performs generally better than the alternative optimal convex combination, and both of them are better than the straight use of either common or task-specific models.

1. Introduction

Quite often a Machine Learning (ML) problem can be split in several subproblems, or tasks, according for instance to different characteristics of the underlying sample. Two opposing alternatives appear then, namely, to build either a single common model over the entire sample, or individual, task-specific models for each subproblem. However, a third and often superior option is to try to simultaneously learn the common and specific models, in such a way that they share information during the training process, so that a better joint solution is found for the overall problem. This Multi-task Learning (MTL) approach was first proposed by R. Caruana [1], and has been enormously extended since its apparition, with MTL paradigms being adapted for the main ML algorithms. Recent surveys are [2,3] and we give a brief overview of MTL in Section 2.

In the case of support vector machines (SVMs), MTL methods were first proposed by Evgeniou and Pontil [4] and subsequently expanded in works such as [5,6]. The starting point of the present work is the framework proposed by Cai and Cherkassky [7,8], where *additive* MTL was introduced for support vector classification and regression. Its primal formulation is similar to the standard SVM primal problem but now the common and task-specific weights are independently regularized and a hyperparameter $\mu > 0$ balances the contribution of these regularizers. More precisely, the Cai-Cherkassky extension [8] of the Multi-Task Learning SVM in [4] considers a SVM framework in which a model is constructed for each task $t, t = 1, \ldots, T$ according to the following primal problem

$$\begin{aligned} &\operatorname{argmin}_{w,v_t,\xi} J(w, v_t, \xi) = C\sum_{t=1}^{T}\sum_{i=1}^{m_t}\xi_i^t + \frac{1}{2}\sum_{t=1}^{T}||v_t||^2 + \frac{\mu}{2}||w||^2 \\ &\text{s.t.} \quad y_i^t\left(w\cdot\phi\left(x_i^t\right) + b + v_t\cdot\phi_t\left(x_i^t\right) + d_t\right) \geqslant p_i^t - \xi_i^t, \\ &\qquad \xi_i^t \geqslant 0;\ i = 1, \ldots, m_t,\ t = 1, \ldots, T. \end{aligned} \tag{1}$$

where a common weight w and bias b as well as their task specific counterparts v_t and d_t are considered. Observe that the contribution of the common model to the final MTL is stronger when small μ values are considered and, conversely, high μ values result on a stronger presence of the task-specific components in the final joint model. However, the value of μ only influences the strength of the common or independent models in an indirect way. To overcome this, a *convex* formulation of MTL SVMs is proposed in [9], which makes that influence explicit, as the final model considered

* Corresponding author.
E-mail addresses: carlos.ruizp@uam.es (C. Ruiz), carlos.alaiz@uam.es (C.M. Alaíz), jose.dorronsoro@uam.es (J.R. Dorronsoro).

https://doi.org/10.1016/j.neucom.2021.01.137
0925-2312/

has the form $\lambda \boldsymbol{w} + (1-\lambda)\boldsymbol{v}_t$, with $0 \leqslant \lambda \leqslant 1$, i.e. a convex combination of the common $\boldsymbol{w}$ and task-specific $\boldsymbol{v}_t$ weights. Here the hyperparameter λ makes explicit the contribution of the common and specific models and, on its end points, allows to capture a single common model when $\lambda = 1$ or just the task-specific models when $\lambda = 0$.

As shown in [9], both formulations are equivalent taking $\mu = \frac{(1-\lambda)^2}{\lambda^2}$ for $0 < \lambda < 1$. Moreover, this equivalence was also checked experimentally in [9] and, as mentioned there, the convex formulation has two clear advantages. The first one is the easier interpretation of the common vs. independent task interplay which, as mentioned, can be measured through the optimal λ^* value. The second one is that it allows a much simpler exploration of the λ hyperparameter space. In fact, while in the additive case it may be difficult to explore the entire $(0, \infty)$ potential range of the μ parameter, the convex formulation naturally constrains λ to the $[0, 1]$ interval, which can be simply explored using, for instance, a uniform grid including explicitly the common $\lambda = 1$ and independent $\lambda = 0$ task extremes.

The MTL SVMs described above are based on the L1-SVM, which is probably the most relevant one, but it is just one option among many SVM proposals for classification and regression. Specially relevant is the case of L2-SVMs [10], which use the squared hinge and ϵ-insensitive losses for classification and regression, respectively. These are differentiable, in contrast with the hinge and ϵ-insensitive losses, and open a link towards the more standard squared loss. In fact, this is done explicitly in Least Squares SVM (LS-SVM; [11]), as it is well known that LS-SVM models are equivalent to kernel ridge regression ones. These and their substantial extension via Gaussian Processes [12] yield powerful regression models which are widely used in practice and receive great attention in research. Notice that kernel regularized logistic regression can also be dealt with in an SVM setting [13], but it is outside the scope of this paper.

In any case, there is no ML model that can be considered universally best. This is essentially the content of the famous No Free Lunch theorem [14] and has been verified experimentally many times (see [15] for an early paper and, more recently, [16]). In particular, the same can be said of the previous SVM formulations and, therefore, of their possible application in an MTL setting. Following this line, a unified convex MTL approach to L1-SVM, L2-SVM and LS-SVM models is presented in this work. Note that, although MTL versions of LS-SVMs have been proposed before [17], they do not follow a convex setup, but control instead the relative importance of each task through a double set of common and task dependent parameters (increasing thus substantially the hyperparametrization costs).

It is important to stress that in this approach a λ value is first considered and then the common and task-specific components are jointly and simultaneously learned for that particular λ. Hence, these λ are model hyperparameters, and the optimal one for a given problem is then found by cross validation. In contrast with this, a natural alternative could be to build first common and task specific SVM models and then combine them in a convex way by selecting an optimal λ. More precisely, the loss associated to a concrete convex mixing λ becomes then a function $J(\lambda)$ that one could try to optimize directly. As shown in this paper, this is rather straightforward for LS-SVMs but much less so for L2-SVMs and, particularly, for L1-SVMs, since then the loss $J(\lambda)$ is not differentiable; nevertheless an approach to minimize $J(\lambda)$ in these cases can also be derived using subdifferential calculus.

In summary, and besides a short and unified review of L1-, L2- and LS-SVM models, the main contributions here are:

- A unified convex formulation of convex MTL for L1-, L2- and LS-SVM models.
- A theoretical derivation of the optimal mixing λ for the combination of independently learned common and task specific models.
- An experimental comparative study of these approaches over several regression and classification problems that demonstrates the effectiveness of the convex MTL approach against the individual common and specific SVM models or their direct convex combination.

The rest of the paper is organized as follows. After a short literature overview in Section 2 of MTL methods with an emphasis on SVM related ones, L1-, L2- and LS-SVMs will be briefly reviewed in Section 3, after which their convex MTL formulation is proposed in Section 4. As an alternative to this, Section 5 shows the optimal way to directly combine previously and independently learned common and task-specific SVM models. All these approaches are compared in Section 6 over several regression and classification problems. In most of them the proposed convex MTL approach gives statistically significant better models both within each SVM group (L1-, L2- or LS-SVM) and also when all SVM model combinations are considered. The paper ends with a brief discussion as well as pointers to further work.

2. Related work

While in this work a support vector machine (SVM) approach to MTL will be considered, MTL proposals have been considered for all the leading ML paradigms; see [18] for a recent survey. This is particularly so for deep networks [3,19], where many architectures and optimization methods have appeared in the literature. Possibly the most straightforward approach to deep MTL is that of hard parameter sharing networks, where, in a layered network, the first layers are the same for all tasks and only the final layers specialize on individual tasks. This architecture organization is often used, for instance, in multitask computer vision [20], where the initially shared weights seek to achieve an enhanced feature representation common to all tasks that they later on take advantage of on the final layers. The alternative to hard parameter sharing are the so called soft parameter sharing models. Here weights may not be directly shared across tasks but, instead, similar models are sought for related tasks; for instance, this can be achieved [21] by adding the L2 distances of the task specific weights as part of the regularization penalty. More complex approaches to achieve knowledge sharing while the task models are being built are neural discriminative dimensionality reduction [22], sluice networks [23] or cross-stitch networks [24].

A common trait in almost all these deep MTL models is that a single input is processed to produce multiple outputs, one for each different task. However, the SVM approach to MTL considered here differs from the above deep MTL methods in that task specific features are used to build the individual models and it is their joint training with that of the global model which binds them into an overall MTL model. In other words, the role of the common model is to connect the task specific ones and to allow a degree of interaction among them.

A starting point for the SVM MTL approach in this paper is the work of Evgeniou and Pontil [4], where SVM primal and dual MTL problems that combine the common and specific tasks lead to a minimization problem which is then solved through standard SVM procedures. On the other hand, in the Learning Using Privileged Information (LUPI) paradigm proposed by Vapnik [25] it is sought that model predictions are improved using group information. Subsequently Vapnik also introduced SVM+, which builds on the LUPI approach to enhance the overall predictive capacity.

A connection between SVM + and MTL-SVMs was proposed by Cai and Cherkasski in [7]; this includes a MTL-SVM model with task-specific biases. They further extended this in [8], where a Generalized SMO (GSMO) algorithm is applied to solve a dual problem involving multiple biases; moreover, GSMO is very flexible as it considers individual common and task-specific kernels together with individual biases, with the price to pay being the need to tune multiple hyperparameters and the impossibility of using efficient, standard SMO-based SVM solvers such as LIBSVM [26]. In any case, these two papers are closely related with the proposal here; however, to alleviate the just mentioned issued, slightly simpler MTL-SVM models, as described in Section 4, will be used. Another connection between SVM + and MTL-SVMs is proposed in [27] which is successfully applied in [5] in a biomedical context. Finally, recent contributions of interest are [28] where a convex MTL approach is applied to feature learning, or [29], which applies MTL to enhance the fairness of classifiers.

3. L1-, L2-, and LS-SVMs Review

The standard formulation of L1- and L2-SVMs is reviewed first, following the approach in [30], which unifies their classification and regression versions. To do so, consider a sample $S = \{(\boldsymbol{x}_i, y_i, p_i),\ 1 \leqslant i \leqslant N\}$, where $y_i = \pm 1$, and the following primal problem for $q \in \{1, 2\}$:

$$\begin{aligned} &\operatorname{argmin}_{\boldsymbol{w},b,\xi} \quad J(\boldsymbol{w}, b, \xi) = C\textstyle\sum_{i=1}^{N} (\xi_i)^q + \frac{1}{2}\|\boldsymbol{w}\|^2 \\ &\text{s.t.} \quad y_i(\boldsymbol{w}\cdot\boldsymbol{x}_i + b) \geqslant p_i - \xi_i,\ i = 1,\ldots,N, \\ &\qquad \xi_i \geqslant 0,\ i = 1,\ldots,N. \end{aligned} \tag{2}$$

When $q = 1$ this problem reduces to the L1-SVM primal problem, and its corresponding dual problem is

$$\begin{aligned} &\operatorname{argmin}_{\boldsymbol{\alpha}} \quad \Theta(\boldsymbol{\alpha}) = \boldsymbol{\alpha}^\intercal \boldsymbol{Q}\boldsymbol{\alpha} - \boldsymbol{p}^\intercal\boldsymbol{\alpha} \\ &\text{s.t.} \quad 0 \leqslant \alpha_i \leqslant C,\ i = 1,\ldots,N,\ \textstyle\sum_{i=1}^{N} y_i\alpha_i = 0, \end{aligned} \tag{3}$$

where $\boldsymbol{\alpha}^\intercal = (\alpha_1, \ldots c, \alpha_N)$ and $\boldsymbol{p}^\intercal = (p_1, \ldots c, p_N)$. When $q = 2$ we have the L2-SVM unified formulation, and its dual problem is:

$$\begin{aligned} &\operatorname{argmin}_{\boldsymbol{\alpha}} \quad \Theta(\boldsymbol{\alpha}) = \boldsymbol{\alpha}^\intercal\left(\boldsymbol{Q} + \tfrac{1}{C}\boldsymbol{I}_N\right)\boldsymbol{\alpha} - \boldsymbol{p}^\intercal\boldsymbol{\alpha} \\ &\text{s.t.} \quad 0 \leqslant \alpha_i,\ i = 1,\ldots,N,\ \textstyle\sum_{i=1}^{N} y_i\alpha_i = 0, \end{aligned} \tag{4}$$

where $\boldsymbol{I}_N$ is the $N \times N$ identity matrix. In both L1- and L2-SVMs, $\boldsymbol{Q}$ represents the kernel matrix, which for linear SVMs is defined by the inner products $Q_{ij} = \boldsymbol{x}_i \cdot \boldsymbol{x}_j$. For kernel SVMs, $\boldsymbol{Q}$ contains the inner products $\phi(\boldsymbol{x}_i) \cdot \phi(\boldsymbol{x}_j)$ of the data mapped into a Reproducible Kernel Hilbert Space (RKHS), and the dual problems can be solved using the kernel trick without having to work with the ϕ transform. Observe also that the L2-SVM dual problem (4) is very similar to the L1-SVM one given by (3), with the main difference being that the C constant no longer appears as an upper bound of the α_i, but instead it moves to the kernel matrix.

Finally, the primal LS-SVM problem is

$$\begin{aligned} &\operatorname{argmin}_{\boldsymbol{w},b,\xi} \quad J(\boldsymbol{w}, b, \xi) = C\textstyle\sum_{i=1}^{N} (\xi_i)^2 + \frac{1}{2}\|\boldsymbol{w}\|^2 \\ &\text{s.t.} \quad y_i(\boldsymbol{w}\cdot\boldsymbol{x}_i + b) = 1 - \xi_i,\ i = 1,\ldots,N, \end{aligned} \tag{5}$$

and its corresponding dual problem can be reduced [11] to solving the following linear system:

$$\left[\begin{array}{c|c} 0 & -\mathbf{y}^\intercal \\ \hline \mathbf{y} & \mathbf{Q} + \frac{1}{C}\mathbf{I}_{N\times N} \end{array}\right] \left[\begin{array}{c} b \\ \hline \boldsymbol{\alpha} \end{array}\right] = \left[\begin{array}{c} 0 \\ \hline \mathbf{1}_N \end{array}\right], \tag{6}$$

where $\boldsymbol{Q}$ is again the kernel matrix, $\boldsymbol{y}^\intercal = (y_1, \ldots, y_N)$, and $\boldsymbol{1}_N$ is the all ones vector of dimension N.

4. Convex Multi-Task L1-, L2-, and LS-SVMs

We recall that the Cai-Cherkassky extension [8] of the Multi-Task Learning SVM proposed in [4] solves the primal Problem 1 where a weight vector w_t and a task specific bias b_t are divided into a common part w and b, and task specific additions v_t and d_t. We also recall that the regularization term penalizes independently the common and task specific deviations adjusting the influence of each part through the parameter μ. In particular, large μ values lead to a smaller common part w and, thus, to stronger task-independent models; on the other hand, a small μ would strengthen the common model. However, μ's influence is somewhat indirect; moreover, the μ range coincides with the entire positive real numbers $(0, \infty)$, which makes difficult its hyperparametrization. To overcome this, we have proposed in [9] a convex formulation which we review next and then extend it to L2- and LS-SVMs.

In the linear case, the convex L1-SVM primal problem is the following:

$$\begin{aligned} &\operatorname{argmin}_{\boldsymbol{w},\boldsymbol{v}_r,b_r,\xi} \quad J(\boldsymbol{w}, \boldsymbol{v}_r, b_r, \xi) = C\textstyle\sum_{r=1}^{T}\sum_{i=1}^{n_r} \xi_i^r + \frac{1}{2}\|\boldsymbol{w}\|^2 + \frac{1}{2}\sum_{r=1}^{T}\|\boldsymbol{v}_r\|^2 \\ &\text{s.t.} \quad y_i^r(\lambda\boldsymbol{w}\cdot\boldsymbol{x}_i^r + (1-\lambda)\boldsymbol{v}_r\cdot\boldsymbol{x}_i^r + b_r) \geqslant p_i^r - \xi_i^r, \\ &\qquad \xi_i^r \geqslant 0,\ i = 1,\ldots,n_r,\ r = 1,\ldots,T. \end{aligned} \tag{7}$$

Here the vector $\boldsymbol{w}$ captures the common part and the vectors $\boldsymbol{v}_r$ do the same with the task-specific ones. Notice that for $\lambda = 0$ and $\lambda = 1$, the problem reduces to the task-specific and common ones respectively (denoted in what follows as ITL or CTL). In a kernel setting the constraints become

$$y_i^r\left(\lambda\boldsymbol{w}\cdot\phi(\boldsymbol{x}_i^r) + (1-\lambda)\boldsymbol{v}_r\cdot\phi_r(\boldsymbol{x}_i^r) + b_r\right) \geqslant p_i^r - \xi_i^r,\ i = 1,\ldots,n_r,\ r = 1,\ldots,T,$$

where $\phi(\boldsymbol{x})$ and $\phi_r(\boldsymbol{x})$ denote the mappings of the original $\boldsymbol{x}$ patterns into the common and specific RKHS; observe that these may be different for each task, i.e., different kernels can be used on each one. The dual problem corresponding to (7) is

$$\begin{aligned} &\operatorname{argmin}_{\boldsymbol{\alpha}} \quad \Theta(\boldsymbol{\alpha}) = \boldsymbol{\alpha}^\intercal \hat{\boldsymbol{Q}}\,\boldsymbol{\alpha} - p^\intercal\boldsymbol{\alpha} \\ &\text{s.t.} \quad 0 \leqslant \alpha_i^r \leqslant C,\ i = 1,\ldots c, n_r,\ r = 1,\ldots,T, \\ &\qquad \textstyle\sum_{i=1}^{n_r} y_i\alpha_i^r = 0,\ r = 1,\ldots,T. \end{aligned} \tag{8}$$

Here $\widehat{\boldsymbol{Q}} = \lambda^2\boldsymbol{Q} + (1-\lambda)^2\boldsymbol{K}$, where $Q_{rs} = k(\boldsymbol{x}^r, \boldsymbol{x}^s) = \phi(\boldsymbol{x}^r)\cdot\phi(\boldsymbol{x}^s)$ is the common kernel matrix and $\boldsymbol{K}$ is a block diagonal matrix where each block $\boldsymbol{K}_t$ contains the task-specific kernels $k_t(\boldsymbol{x}^r, \boldsymbol{x}^s) = \phi_t(\boldsymbol{x}^r)\cdot\phi_t(\boldsymbol{x}^s)$.

It should be pointed out that the dual problem (8) is essentially identical to (3) except for the fact that in (8) there are T task-specific equality constraints (because of the T task-specific biases of the model) and, also, that $\widehat{\boldsymbol{Q}}$ is defined in terms of several kernels. While a general SMO algorithm [8] could be used to solve (8), here a common bias for all tasks will be considered so that the very efficient SMO implementation in the LIBSVM library [26] can be used.

Analogously to the L1-SVM above, the primal problem for a convex multi-task L2-SVM can be defined as:

$$\begin{aligned} &\operatorname{argmin}_{\boldsymbol{w},\boldsymbol{v}_r,b_r,\xi} \quad J(\boldsymbol{w}, \boldsymbol{v}_r, b_r, \xi) = \tfrac{C}{2}\textstyle\sum_{r=1}^{T}\sum_{i=1}^{n_r} \left(\xi_i^r\right)^2 + \frac{1}{2}\|\boldsymbol{w}\|^2 + \frac{1}{2}\sum_{r=1}^{T}\|\boldsymbol{v}_r\|^2 \\ &\text{s.t.} \quad y_i^r(\lambda\boldsymbol{w}\cdot\phi(\boldsymbol{x}_i^r) + (1-\lambda)\boldsymbol{v}_r\cdot\phi_r(\boldsymbol{x}_i^r) + b_r) \geqslant p_i^r - \xi_i^r, \\ &\qquad \xi_i^r \geqslant 0,\ i = 1,\ldots,n_r,\ r = 1,\ldots,T, \end{aligned}$$

and its Lagrangian becomes

$$L(\boldsymbol{w}, \boldsymbol{v}_r, b_r, \boldsymbol{\xi}, \boldsymbol{\alpha}, \boldsymbol{\beta}) = \frac{C}{2}\sum_{r=1}^{T}\sum_{i=1}^{n_r}\left(\xi_i^r\right)^2 + \frac{1}{2}\|\boldsymbol{w}\|^2 + \frac{1}{2}\sum_{r=1}^{T}\|\boldsymbol{v}_r\|^2 - \boldsymbol{\beta}^\top\boldsymbol{\xi} - \sum_{r=1}^{T}\sum_{i=1}^{n_r}\alpha_i^r\left\{y_i^r\left(\lambda\boldsymbol{w}\cdot\phi(\boldsymbol{x}_i^r) + (1-\lambda)\boldsymbol{v}_r\cdot\phi_r(\boldsymbol{x}_i^r) + b_r\right) - p_i^r + \xi_i^r\right.$$

here $\boldsymbol{\alpha}, \boldsymbol{\beta} \geqslant 0$ are the Lagrange multiplier vectors. It follows from this that the dual problem for multi-task L2-SVM is

$$\begin{aligned} &\text{argmin}_{\boldsymbol{\alpha}} \quad \Theta(\boldsymbol{\alpha}) = \boldsymbol{\alpha}^\top\left(\hat{\boldsymbol{Q}} + \tfrac{1}{C}\boldsymbol{I}_N\right)\boldsymbol{\alpha} - p^\top\boldsymbol{\alpha} \\ &\text{s.t.} \quad 0 \leqslant \alpha_i^r,\ i = 1,\dots c, n_r,\ r = 1,\dots,T, \\ &\qquad \textstyle\sum_{i=1}^{n_r} y_i\alpha_i^r = 0,\ r = 1,\dots,T, \end{aligned} \tag{9}$$

where $\widehat{\boldsymbol{Q}} = \lambda^2\boldsymbol{Q} + (1-\lambda)^2\boldsymbol{K}$ is again the same multi-task kernel used in (8). Observe that the differences between the dual problems of convex multi-task L1 and L2-SVMs are the same as those between the standard single-task L1- and L2-SVM duals.

Finally, to extend the convex MTL approach to the LS-SVM case the following multi-task primal problem is defined:

$$\begin{aligned} &\text{argmin}_{\boldsymbol{w},\boldsymbol{v}_r,b_r,\boldsymbol{\xi}} \quad J(\boldsymbol{w}, \boldsymbol{v}_r, b_r, \boldsymbol{\xi}) = \tfrac{C}{2}\textstyle\sum_{r=1}^{T}\sum_{i=1}^{n_r}\left(\xi_i^r\right)^2 + \frac{1}{2}\|\boldsymbol{w}\|^2 + \frac{1}{2}\sum_{r=1}^{T}\|\boldsymbol{v}_r\|^2 \\ &\text{s.t.} \quad y_i^r\left(\lambda\boldsymbol{w}\cdot\phi(\boldsymbol{x}_i^r) + (1-\lambda)\boldsymbol{v}_r\cdot\phi_r(\boldsymbol{x}_i^r) + b_r\right) = 1 - \xi_i^r, \\ &\qquad i = 1,\dots,n_r,\ r = 1,\dots,T, \end{aligned}$$

whose corresponding Lagrangian is

$$L(\boldsymbol{w}, \boldsymbol{v}_r, b_r, \boldsymbol{\xi}, \boldsymbol{\alpha}, \mu) = \frac{C}{2}\sum_{r=1}^{T}\sum_{i=1}^{n_r}\left(\xi_i^r\right)^2 + \frac{1}{2}\|\boldsymbol{w}\|^2 + \frac{1}{2}\sum_{r=1}^{T}\|\boldsymbol{v}_r\|^2 - \sum_{r=1}^{T}\sum_{i=1}^{n_r}\alpha_i^r\left\{y_i^r\left(\lambda\boldsymbol{w}\cdot\phi(\boldsymbol{x}_i^r) + (1-\lambda)\boldsymbol{v}_r\cdot\phi_r(\boldsymbol{x}_i^r) + b_r\right) - 1 + \xi_i^r\right.$$

Since there are only equality primal constraints, notice that the Lagrange multipliers α_i^r can now be either positive or negative. As for standard LS-SVMs, the dual problem reduces to the following system of linear equations:

$$\left[\begin{array}{c|c} \boldsymbol{0}_{T\times T} & \boldsymbol{A} \\ \hline \boldsymbol{A}^\top & \widehat{\boldsymbol{Q}} + \frac{1}{C}\boldsymbol{I}_N \end{array}\right]\begin{bmatrix} b_1 \\ \vdots \\ b_T \\ \boldsymbol{\alpha} \end{bmatrix} = \begin{bmatrix} 0 \\ \vdots \\ 0 \\ \boldsymbol{1}_N \end{bmatrix},\ \underset{T\times N}{\boldsymbol{A}} = \begin{pmatrix} \overbrace{y_1^1\dots y_{n_1}^1}^{n_1} & \dots & \overbrace{0\dots 0}^{n_T} \\ \ddots & \ddots & \ddots \\ 0\dots 0 & \dots & y_1^T\dots y_{n_T}^T \end{pmatrix}.$$

Here $\widehat{\boldsymbol{Q}}$ is the same multi-task kernel used in (8). Again, the only differences between the standard LS-SVM dual problem (6) and the convex multi-task LS-SVM problem are the use of multiple biases and of the multi-task kernel.

5. Optimal convex combination of SVMs

In the Convex Multi-Task SVM proposals of Section 4, the λ coefficient defines a concrete combination of common and specific models that are jointly learned once a λ is fixed. It thus acts as a hyperparameter whose optimal value has to be found through cross validation or a similar procedure, with the corresponding computational costs. This suggests the alternative of building first and independently the common and task-specific SVM models and afterwards combining them with a certain λ to be optimal in some sense.

Notice that, in this approach, the common and specific models are no longer jointly learned and, thus, do not cooperate to solve the problem at hand. On the other side, the overall cost is much smaller, as just one set of common and specific SVM models is to be learned and, as shown below, finding the mixing λ has a low computational cost. Thus, one has to balance a possibly worst performance, as there is no joint multitask learning anymore, with the computational savings of this alternative approach, called the Optimal Convex Combination of SVMs and denoted as cvxCMB.

The procedure to choose the optimal λ for cvxCMB is derived next. Assume that common f and task-specific g_r SVM models have been built. For a pattern $\boldsymbol{x}^r$ of task r, their predictions $f(\boldsymbol{x}^r)$ and $g_r(\boldsymbol{x}^r)$ are combined to yield

$$f_r(\boldsymbol{x}^r) = \lambda f(\boldsymbol{x}^r) + (1-\lambda)g_r(\boldsymbol{x}^r).$$

If $L(\hat{y}, y)$ is the loss to be minimized, the optimal λ can be found by solving the following optimization problem:

$$\text{argmin}_{0\leqslant\lambda\leqslant 1} \quad J(\lambda) = \sum_{r=1}^{T}\sum_{i=1}^{n_r} L\left(\lambda f(\boldsymbol{x}_i^r) + (1-\lambda)g_r(\boldsymbol{x}_i^r), y_i^r\right). \tag{10}$$

In what follows, simple necessary and sufficient conditions for optimality are derived, making possible to find the optimal λ^* with $O(N\log N)$ cost for the absolute and squared errors for regression, and the squared error, hinge and squared hinge losses for classification. While each loss L results in a different cost function J and has to be treated separately, these J also have some elements in common, so it is useful to rewrite them to make those elements clearer. Focusing first on the absolute error loss for regression L1-SVMs, and on the squared error loss for regression L2-SVMs and classification and regression LS-SVMs, the combination can be written as:

$$\begin{aligned} \lambda f(\boldsymbol{x}_i^r) + (1-\lambda)g_r(\boldsymbol{x}_i^r) - y_i^r &= \lambda\left\{f(\boldsymbol{x}_i^r) - g_r(\boldsymbol{x}_i^r)\right\} + \left\{g_r(\boldsymbol{x}_i^r) - y_i^r\right\} \\ &= \lambda c_i^r + d_i^r, \end{aligned}$$

where

$$c_i^r = f(\boldsymbol{x}_i^r) - g_r(\boldsymbol{x}_i^r),\ d_i^r = g_r(\boldsymbol{x}_i^r) - y_i^r.$$

Then, for the absolute error loss, (10) becomes:

$$\text{argmin}_{0\leqslant\lambda\leqslant 1} \quad J(\lambda) = \sum_{r=1}^{T}\sum_{i=1}^{n_r}\left|\lambda c_i^r + d_i^r\right|, \tag{11}$$

while for the squared error the problem is:

$$\text{argmin}_{0\leqslant\lambda\leqslant 1} \quad J(\lambda) = \sum_{r=1}^{T}\sum_{i=1}^{n_r}\left(\lambda c_i^r + d_i^r\right)^2. \tag{12}$$

Similarly, using the hinge and squared hinge losses for classification L1- and L2-SVMs, respectively, the combination is written as:

$$\begin{aligned} 1 - \left\{\lambda f(\boldsymbol{x}_i^r) + (1-\lambda)g_r(\boldsymbol{x}_i^r)\right\}y_i^r &= \lambda\left\{y_i^r\left(g_r(\boldsymbol{x}_i^r) - f(\boldsymbol{x}_i^r)\right)\right\} + 1 \\ &\quad - y_i^r g_r(\boldsymbol{x}_i^r) \\ &= \lambda c_i^r + d_i^r, \end{aligned}$$

where here the c_i^r and d_i^r variables are defined as:

$$c_i^r = y_i^r\left(g_r(\boldsymbol{x}_i^r) - f(\boldsymbol{x}_i^r)\right),\ d_i^r = 1 - y_i^r g_r(\boldsymbol{x}_i^r).$$

Now, for the hinge loss, (10) becomes:

$$\operatorname{argmin}_{0\leqslant\lambda\leqslant 1} \quad J(\lambda) = \sum_{r=1}^{T}\sum_{i=1}^{n_r}\left[\lambda c_i^r + d_i^r\right]_+, \tag{13}$$

while for the squared hinge loss the resultant problem is:

$$\operatorname{argmin}_{0\leqslant\lambda\leqslant 1} \quad J(\lambda) = \sum_{r=1}^{T}\sum_{i=1}^{n_r}\left[\lambda c_i^r + d_i^r\right]_+^2. \tag{14}$$

The simplest case is that of the squared error, as the J function in (12) is differentiable, and solving $J'(\lambda)=0$ results in

$$\lambda' = -\frac{\sum_{r=1}^{T}\sum_{i=1}^{n_r} d_i^r c_i^r}{\sum_{r=1}^{T}\sum_{i=1}^{n_r}\left(c_i^r\right)^2},$$

and the optimum is hence $\lambda^* = \max(0,\min(1,\lambda'))$. This applies to LS-SVMs and to regression L2-SVMs.

The $J(\lambda)$ function is also differentiable for the squared hinge loss but only piecewise differentiable for the absolute error and hinge losses, where differentiability breaks down at the "elbows", i.e., the points where $\lambda c_i^r + d_i^r = 0$ or, equivalently, $\lambda_{i,r} = \max(0,\min(1,-d_i^r/c_i^r))$. There are N such elbows and sorting them in increasing order and reindexing the pairs $(i,r), 1\leqslant i\leqslant n_r, 1\leqslant r\leqslant T$, as $1\leqslant j\leqslant N$, it can be assumed that $0\leqslant\lambda_1\leqslant\lambda_2\leqslant\ldots\leqslant\lambda_N\leqslant 1$. Notice also that the cost functions (11) and (13) are convex, and thus by the generalized Fermat theorem:

$$\lambda^* = \operatorname{argmin}_\lambda J(\lambda) \iff 0\in\partial J(\lambda^*),$$

where ∂ denotes the subdifferential operator. Moreover these cost functions can be written as:

$$J(\lambda) = \sum_{r=1}^{T}\sum_{i=1}^{n_r}\ell\left(\lambda c_i^r + d_i^r\right),$$

and since all functions $\ell_i^r(\lambda) = \ell\left(\lambda c_i^r + d_i^r\right)$ share the same domain $[0,1], \partial J = \sum_r\sum_i \partial\ell_i^r$. Now it is easy to see the following result.

Proposition 1. For the absolute error and hinge losses, $\partial J(\lambda)$ is single valued and constant between the elbows λ_j. Moreover, if $\partial J(\lambda) = J'(\lambda) = \gamma$ for some λ between two consecutive distinct elbows, i.e., $\lambda_j<\lambda<\lambda_{j+1}$, then $\gamma\in\partial J(\lambda_j)$ and $\gamma\in\partial J(\lambda_{j+1})$.

An immediate consequence of this is that for these two losses it is sufficient to search the optimal solution at the elbows λ_j. More precisely, next proposition is satisfied for them and the squared hinge loss.

Proposition 2. With the previous notation:

1. A value λ^* is optimal for the absolute error loss of problem (11) iff λ^* is an elbow, that is, $\lambda^* = \lambda_k$ for some $k = 1,\ldots,N$, and

$$-\sum_{j=1}^{k-1}|c_j| + \sum_{j=k+1}^{N}|c_j| \in [-|c_k|,|c_k|]. \tag{15}$$

2. A value λ^* is optimal for the hinge loss of problem (13) iff λ^* is an elbow, that is, $\lambda^* = \lambda_k$ for some $k = 1,\ldots,N$, and

$$-\sum_{j=1}^{k-1}\max(0,c_j) - \sum_{j=k+1}^{N}\min(0,c_j) \in [\min(0,c_k),\max(0,c_k)]. \tag{16}$$

3. For the squared hinge loss of (14), set first $\lambda_0 = 0$ and $\lambda_{N+1} = 1$, and for each λ_k value, $0\leqslant k\leqslant N$, define $\widehat{\lambda}_k$ as

$$\widehat{\lambda}_k = -\frac{\sum_{j=1}^{k-1}\max(0,c_j)d_j + \sum_{j=k+1}^{N}\min(0,c_j)d_j}{\sum_{j=1}^{k-1}\max\left(0,c_j\right)^2 + \sum_{j=k+1}^{N}\min\left(0,c_j\right)^2}. \tag{17}$$

Then, when $\widehat{\lambda}_0<0$, problem (14) has a minimum at $\lambda^*=0$. On the other hand, if $\lambda_k\leqslant\widehat{\lambda}_k\leqslant\lambda_{k+1}$ for some $\widehat{\lambda}_k$, then $\lambda^* = \widehat{\lambda}_k$ is a minimum of (14). Finally, if neither of the previous conditions holds, (14) has a minimum at $\lambda = 1$.

Proof.

1. For the absolute error loss, the subdifferential of $J(\lambda)$ at an elbow λ_k is given by the interval

$$\partial J(\lambda_k) = -\sum_{j=1}^{k-1}|c_j| + \sum_{j=k+1}^{N}|c_j| + [-|c_k|,|c_k|].$$

Therefore, λ_k is optimal iff $0\in\partial J(\lambda_k)$, that is

$$0\in\partial J(\lambda_k) \iff -\sum_{j=1}^{k-1}|c_j| + \sum_{j=k+1}^{N}|c_j| \in [-|c_k|,|c_k|].$$

2. Similarly, for the hinge loss, the subdifferential at an elbow λ_k is given by the interval

$$\partial J(\lambda_k) = \sum_{j=1}^{k-1}\max(0,c_j) + \sum_{j=k+1}^{N}\min(0,c_j) + [\min(0,c_k),\max(0,c_k)],$$

that is, the condition for λ_k to be optimal, $0\in\partial J(\lambda_k)$, becomes

$$-\sum_{j=1}^{k-1}\max(0,c_j) - \sum_{j=k+1}^{N}\min(0,c_j) \in [\min(0,c_k),\max(0,c_k)].$$

3. Finally, the objective function $J(\lambda)$ of the squared hinge loss is differentiable and its continuous and piecewise linear derivative, for λ such that $\lambda_k\leqslant\lambda<\lambda_{k+1}$, is given by

$$J'(\lambda) = \sum_{j=1}^{k-1}2\max(0,c_j)\left(\lambda c_j + d_j\right) + \sum_{j=k+1}^{N}2\min(0,c_j) \times\left(\lambda c_j + d_j\right).$$

Solving $J'(\lambda)=0$ in each piece $\lambda_k\leqslant\lambda<\lambda_{k+1}$ results in the λ_k defined by (17); however, their possible location in the $[0,1]$ interval has to be analyzed. To do so, observe that, since $J(\lambda)$ is strictly convex and differentiable, $J'(\lambda)$ is increasing. If $\widehat{\lambda}_0<0$, then by (17):

$$\sum_{j=1}^{k-1}\max(0,c_j)d_j + \sum_{j=k+1}^{N}\min(0,c_j)d_j > 0,$$

in other words, $J'(0)>0$ and the minimum of $J(\lambda)$ in the interval $[0,1]$ is achieved at $\lambda^*=0$. Next, if for some k we have $0\leqslant\lambda_k\leqslant\widehat{\lambda}_k\leqslant\lambda_{k+1}\leqslant 1$, then $J'\left(\widehat{\lambda}_k\right)=0$, i.e., $\lambda^* = \widehat{\lambda}_k$ is a minimum of $J(\lambda)$ in $[0,1]$. Finally, if the two previous conditions are not met, then $\widehat{\lambda}_N>1$, which implies

$$\sum_{j=1}^{k-1}\max(0,c_j)\left(d_j+c_j\right) + \sum_{j=k+1}^{N}\min(0,c_j)\left(d_j+c_j\right) < 0,$$

that is, $J'(1)<0$, and the minimum of $J(\lambda)$ in $[0,1]$ is achieved at $\lambda^*=1$.

As a consequence, Eqs. (15)–(17) characterize the optimal λ^* for cvxCMB using the absolute error, hinge and squared hinge losses with the $O(N\log N)$ cost of computing the elbow values and sorting them.

6. Experiments

This section starts with a description of the models to be compared, the classification and regression problems used and the experimental methodology followed; after that, the obtained results are shown and briefly discussed.

The following four models are considered:

Table 1
Sample sizes, dimensions and number of tasks of the datasets used.

Dataset	Size	No. feat.	No. tasks	Avg. task size	Min. t. s.	Max. t. s.
majorca	15330	765	14	1095	1095	1095
tenerife	15330	765	14	1095	1095	1095
california	19269	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	32302	184	47	687	59	3089
landmine	14820	10	28	511	445	690
adult_(G)	48842	106	2	24421	16192	32650
adult_(R)	48842	103	5	9768	406	41762
adult_(G, R)	48842	101	10	4884	155	28735
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

- Common Task Learning LX-SVM (CTL-LX): A single LX-SVM which uses data from all the tasks and does not make use of the task information.
- Independent Task Learning LX-SVM (ITL-LX): Multiple task-specific LX-SVMs, each of which only uses 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 5.
- Convex Multi-Task Learning LX-SVM (cvxMTL-LX): The extensions proposed here of the L1-SVM convex MTL model in [9].

In the preceding, LX stands for either L1, L2 or LS.

The characteristics of the regression and classification datasets used are given in Table 1. The regression problems used in this work are majorca, tenerife, boston, california, abalone and crime. In majorca and tenerife our goal is to predict the photovoltaic energy produced in parks installed in the islands of Majorca and Tenerife, respectively, and the tasks are defined as the energy prediction at each daylight hour. In the boston and california problems, obtained from the Kaggle repository, the goal is to predict house prices; here the tasks are defined according to a geographical division of the corresponding areas. Finally, the regression problems abalone and crime are taken from the UCI repository. The goal of the first is to predict the number of rings of a certain kind of marine molluscs and the three tasks considered are those predictions for male, female and infant specimens. In the crime problem the crime rates per 100.000 habitants in different cities of the U.S.A. are to be predicted. We define the tasks as the prediction of this crime rate in each state.

The classification problems are landmine, binding, adult and compas. In landmine the goal is to detect landmines, with different mine types defining the different tasks. In binding the goal is to predict whether a given molecule will bind to a peptide, and each molecule defines a task. In adult the goal is to predict whether a person has an income per year larger than $ 50.000. The compas dataset is derived from the application of the COMPAS tool, developed by the Northpointe company, to predict recidivism of convicted defendants. We consider it as a two class problem, with the positive class being those defendants to which the COMPAS algorithm assigns a "low" score and, hence, are considered to have little risk of recidivism. In both adult and compas the tasks are defined according to a person's characteristic of gender, race or both. In the compas problem, we have removed patterns from the original Native-American and Asian groups, as they are too small to fit independent models for them. We note that, in order to work with Gaussian kernels, input features are scaled for all problems to a $[0, 1]$ range.

Table 2 shows the hyperparameters of each model and the grids we will use to estimate their best values. The C, γ and (for regression) ϵ hyperparameters for the CTL-LX and ITL-LX models are selected by CV, as described below. The common γ_c and specific γ_s^r kernel scales used for the cvxMTL-LX models are the same than those obtained for the CTL-LX and ITL-LX models, respectively. Their remaining hyperparameters C, λ and possibly ϵ, are then selected using again CV. Finally, recall that the cvxCMB-LX works with the already selected CTL-LX and ITL-LX models, and the optimal mixing λ^* is computed directly using the results from Section 5.

Data from the years 2013, 2014 and 2015 are used in the majorca and tenerife problems for training, validation and test, respectively. The other datasets do not have predefined splits, and all grid searches are performed using a nested CV with 3 external and 3 internal random folds, stratifying the data by task so that all folds have a similar task distribution. In more detail, we first randomly obtain three external folds and we cyclically keep one of them as a test set and apply standard three fold CV on the other two to estimate optimal model hyperparameters. The corresponding optimal models are then applied on each test fold and we report the mean (and, for regression, the standard deviation) of these three test values. Moreover, to have comparable results, the same folds are used for the CTL-LX, ITL-LX, cvxMTL-LX and cvxCMB-LX models.

Table 2
Hyperparameters, grids used to select them (when appropriate) and hyperparameter selection method for each model.

	Grid	CTL-L1,2	ITL-L1,2	cvxMTL-L1,2	CTL-LS	ITL-L,S	cvxMTL-LS
C	$\left\{4^k : -2 \leqslant k \leqslant 6\right\}$	CV	CV	CV	CV	CV	CV
ϵ	$\left\{\frac{\sigma}{4^k} : 1 \leqslant k \leqslant 6\right\}$	CV	CV	CV	-	-	-
γ_c	$\left\{\frac{4^k}{d} : -2 \leqslant k \leqslant 3\right\}$	CV	-	CTL-L1,2	CV	-	CTL-LS
γ_s^r	$\left\{\frac{4^k}{d} : -2 \leqslant k \leqslant 3\right\}$	-	CV	ITL-L1,2	-	CV	ITL-LS
λ	$\{0.1k : 0 \leqslant k \leqslant 10\}$	-	-	CV	-	-	CV

Table 3
Test MAE (top) and R2 score (bottom) and Wilcoxon-based ranking for the models selected using the MAE for hyperparametrization. The best models are shown in bold.

	maj.		ten.		boston			california			abalone			crime		
MAE																
ITL-L1	5.087	(6)	5.743	(3)	2.341	0.229	**(1)**	36883.582	418.435	(2)	1.481	0.051	(3)	0.078	0.001	(2)
CTL-L1	5.175	(7)	5.891	(5)	**2.192**	**0.244**	**(1)**	41754.337	270.908	(6)	1.482	0.050	(3)	0.078	0.001	(2)
cvxCMB-L1	**5.047**	(5)	**5.340**	**(1)**	2.239	0.255	**(1)**	36880.238	420.417	**(1)**	1.470	0.052	(2)	0.077	0.002	(2)
cvxMTL-L1	5.050	(5)	5.535	(2)	2.206	0.292	**(1)**	**36711.383**	**343.333**	**(1)**	**1.454**	**0.048**	**(1)**	**0.074**	**0.002**	**(1)**
ITL-L2	4.952	(3)	**5.629**	(3)	2.356	0.300	**(1)**	37374.618	433.511	(5)	1.498	0.054	(4)	0.079	0.002	(2)
CTL-L2	5.193	(7)	6.107	(8)	**2.083**	**0.136**	**(1)**	42335.612	163.773	(8)	1.503	0.047	(5)	0.080	0.002	(2)
cvxCMB-L2	4.869	(3)	5.963	(6)	2.089	0.128	**(1)**	37374.618	433.511	(4)	1.494	0.050	(4)	0.077	0.003	(2)
cvxMTL-L2	**4.854**	(2)	5.784	(4)	2.089	0.134	**(1)**	**37202.603**	**419.166**	(3)	**1.482**	**0.049**	(3)	**0.077**	**0.002**	(2)
ITL-LS	4.937	(3)	5.649	(3)	2.204	0.116	**(1)**	37348.347	441.240	(4)	1.496	0.051	(4)	0.079	0.002	(2)
CTL-LS	5.193	(7)	6.005	(7)	**2.072**	**0.143**	**(1)**	42259.492	146.825	(7)	1.502	0.052	(5)	0.079	0.002	(2)
cvxCMB-LS	4.977	(4)	**5.593**	(3)	2.081	0.146	**(1)**	37339.179	430.288	(4)	1.486	0.049	(4)	0.079	0.002	(2)
cvxMTL-LS	**4.824**	**(1)**	5.754	(4)	2.077	0.152	**(1)**	**37231.043**	**420.992**	(4)	**1.478**	**0.050**	(3)	**0.076**	**0.002**	(2)
R2																
ITL-L1	0.845	(6)	0.901	(7)	0.821	0.041	(2)	0.699	0.009	(7)	0.543	0.022	(8)	0.732	0.021	(3)
CTL-L1	0.837	(9)	0.901	(6)	0.854	0.036	**(1)**	0.639	0.006	(10)	0.559	0.014	(6)	0.740	0.027	(3)
cvxCMB-L1	0.844	(6)	0.905	(4)	0.845	0.053	**(1)**	0.699	0.009	(6)	0.555	0.018	(7)	0.741	0.029	(3)
cvxMTL-L1	**0.846**	(4)	**0.908**	(2)	**0.858**	**0.057**	**(1)**	**0.703**	**0.007**	(6)	**0.568**	**0.012**	(5)	**0.760**	**0.024**	(2)
ITL-L2	0.846	(5)	0.906	(3)	0.836	0.045	(2)	0.707	0.009	(5)	0.565	0.025	(6)	0.743	0.017	(3)
CTL-L2	0.840	(8)	0.901	(8)	**0.889**	**0.017**	**(1)**	0.645	0.005	(9)	0.574	0.013	(4)	0.744	0.028	(3)
cvxCMB-L2	0.850	(3)	0.900	(9)	0.885	0.013	**(1)**	0.707	0.009	(4)	0.571	0.018	(4)	0.755	0.024	(3)
cvxMTL-L2	**0.863**	(2)	**0.908**	**(1)**	0.888	0.015	**(1)**	**0.709**	**0.008**	**(1)**	**0.580**	**0.014**	(3)	**0.762**	**0.028**	**(1)**
ITL-LS	0.849	(3)	0.907	(3)	0.856	0.008	**(1)**	0.707	0.009	(3)	0.573	0.015	(4)	0.743	0.022	(3)
CTL-LS	0.838	(9)	0.904	(5)	**0.894**	**0.015**	**(1)**	0.646	0.005	(8)	0.576	0.016	(4)	0.746	0.032	(3)
cvxCMB-LS	0.843	(7)	0.907	(2)	0.886	0.024	**(1)**	0.707	0.009	(2)	0.581	0.012	(2)	0.746	0.021	(3)
cvxMTL-LS	**0.863**	**(1)**	**0.910**	**(1)**	0.890	0.016	**(1)**	**0.709**	**0.008**	(2)	**0.581**	**0.015**	**(1)**	**0.763**	**0.028**	**(1)**

For classification problems the F1 score is used in CV, as the imbalance ratio of the `landmine` dataset is a large 200/13, and test results for both F1 and accuracy are reported. Notice that the classification losses do not aim to directly minimize either one of these scores, in contrast to what happens in regression, where the ϵ-insensitive loss is close to the mean absolute error (MAE) while the squared ϵ-insensitive loss is closer to the mean squared error. This implies that using the MAE as the CV score is likely to favor the performance of the ϵ-insensitive loss-based regression models and, to the contrary, using the MSE as the CV score might penalize them and favor the squared loss models. Because of this, the results are reported using both the MAE and the MSE as different CV scores and test performances are given in terms of both the MAE and the R2 score, which is very closely related to the MSE.

Starting with regression, Table 3 gives the test MAE and R2 values when the hyperparameters are chosen using the MAE as the CV score, and Table 4 shows the same test measures when MSE is the CV score. When nested 3-fold CV has been used, the tables give the mean test values and their standard deviation. The tables are split in three blocks, one for each of the L1-, L2- and LS-SVM models. To

Table 4
Test MAE (top) and R2 score (bottom) and Wilcoxon-based ranking for the models selected using the MSE for hyperparametrization. The best models are shown in bold.

	maj.		ten.		boston			california			abalone			crime		
MAE																
ITL-L1	5.087	(7)	5.743	(3)	2.437	0.281	(3)	36941.516	450.767	(1)	1.480	0.058	(3)	0.079	0.002	(3)
CTL-L1	5.175	(8)	5.891	(7)	2.315	0.192	(2)	41857.602	235.021	(6)	1.479	0.047	(3)	0.078	0.000	(2)
cvxCMB-L1	**4.920**	(4)	5.743	(4)	2.315	0.192	(3)	**36941.476**	**450.711**	**(1)**	1.471	0.057	(2)	0.079	0.002	(2)
cvxMTL-L1	5.050	(6)	**5.535**	**(1)**	**2.244**	**0.150**	(1)	36999.003	360.445	(2)	**1.455**	**0.046**	**(1)**	**0.074**	**0.001**	**(1)**
ITL-L2	4.924	(5)	5.752	(5)	2.437	0.324	(3)	37407.929	461.878	(5)	1.497	0.050	(5)	0.079	0.002	(2)
CTL-L2	5.193	(8)	6.107	(9)	2.096	0.112	(1)	42335.612	163.773	(7)	1.504	0.048	(6)	0.079	0.002	(2)
cvxCMB-L2	**4.813**	**(1)**	**5.623**	(3)	2.116	0.131	(1)	37398.940	449.498	(5)	1.495	0.051	(5)	0.078	0.003	(2)
cvxMTL-L2	4.854	(4)	5.784	(6)	**2.082**	**0.130**	(1)	**37356.599**	**390.629**	(4)	**1.481**	**0.041**	(4)	**0.076**	**0.000**	(2)
ITL-LS	4.937	(5)	5.649	(3)	2.326	0.231	(3)	37385.244	403.331	(4)	1.495	0.045	(5)	0.079	0.002	(2)
CTL-LS	5.193	(8)	6.005	(8)	**2.072**	**0.143**	**(1)**	42339.063	156.624	(7)	1.504	0.043	(6)	0.078	0.002	(2)
cvxCMB-LS	**4.820**	(2)	5.578	(2)	2.136	0.106	(1)	37377.005	391.694	(4)	1.491	0.048	(5)	0.078	0.002	(2)
cvxMTL-LS	4.824	(3)	**5.754**	(6)	2.090	0.090	(1)	**37232.918**	**397.866**	(3)	**1.478**	**0.042**	(3)	**0.076**	**0.000**	(2)
R2																
ITL-L1	0.845	(6)	0.901	(9)	0.800	0.050	(3)	0.703	0.009	(8)	0.534	0.053	(10)	0.732	0.017	(4)
CTL-L1	0.837	(7)	0.901	(8)	0.860	0.026	(2)	0.642	0.006	(10)	0.564	0.011	(8)	0.748	0.017	(3)
cvxCMB-L1	**0.852**	(4)	0.901	(10)	0.860	0.026	(3)	0.703	0.009	(7)	0.550	0.036	(9)	0.733	0.018	(3)
cvxMTL-L1	0.846	(5)	**0.908**	(5)	**0.871**	**0.019**	(1)	**0.705**	**0.008**	(6)	**0.573**	**0.011**	(7)	**0.764**	**0.019**	(1)
ITL-L2	0.850	(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)
cvxMTL-L2	**0.863**	(2)	0.908	(4)	**0.887**	**0.015**	(1)	**0.708**	**0.007**	(2)	**0.581**	**0.011**	(2)	**0.768**	**0.020**	**(1)**
ITL-LS	0.849	(4)	0.907	(5)	0.841	0.028	(3)	0.707	0.009	(5)	0.577	0.012	(4)	0.743	0.021	(3)
CTL-LS	0.838	(7)	0.904	(7)	**0.894**	**0.015**	**(1)**	0.645	0.005	(9)	0.575	0.012	(4)	0.754	0.022	(3)
cvxCMB-LS	0.856	(3)	0.909	(3)	0.877	0.009	(1)	0.707	0.009	(3)	0.580	0.013	(3)	0.750	0.024	(3)
cvxMTL-LS	**0.863**	**(1)**	**0.910**	(2)	0.890	0.014	(1)	**0.710**	**0.008**	**(1)**	**0.582**	**0.011**	**(1)**	**0.763**	**0.019**	(2)

Table 5
Test F1 (top) and accuracy (bottom) scores, global and block-wise Wilcoxon-based rankings for classification problems. The best models in each block are shown in bold.

	comp_(G)	comp_(R)	comp_(G,R)	ad_(G)	ad_(R)	ad_(G,R)	landmine	binding	**mean**	**rank**	**Wil.**
					F1						
ITL-L1	0.625	**0.639**	0.630	**0.659**	0.653	0.657	0.231	0.867	0.620	10	1
CTL-L1	0.623	0.638	0.638	0.657	0.650	0.653	0.255	0.901	0.627	7	1
cvxCMB-L1	0.616	0.638	0.638	0.658	0.650	0.653	**0.270**	0.901	**0.628**	6	1
cvxMTL-L1	**0.627**	0.636	**0.640**	**0.659**	**0.655**	**0.659**	0.242	**0.907**	**0.628**	5	1
ITL-L2	0.636	0.623	0.607	**0.668**	**0.666**	**0.668**	0.256	0.867	0.624	8	3
CTL-L2	**0.640**	0.647	**0.651**	0.665	0.661	0.659	**0.270**	0.903	0.637	2	2
cvxCMB-L2	0.629	0.640	0.645	0.666	0.662	0.661	**0.270**	0.903	0.634	3	2
cvxMTL-L2	0.634	**0.651**	0.650	**0.668**	**0.666**	**0.668**	0.263	**0.909**	**0.639**	1	1
ITL-LS	**0.631**	0.622	0.608	**0.659**	**0.659**	**0.660**	0.243	0.867	0.619	12	2
CTL-LS	0.628	**0.644**	**0.649**	0.650	0.653	0.647	0.230	0.853	0.619	11	2
cvxCMB-LS	0.630	0.635	0.642	0.657	0.658	0.654	0.238	0.873	0.623	9	2
cvxMTL-LS	0.630	0.641	0.648	**0.659**	**0.659**	0.659	**0.257**	**0.906**	**0.632**	4	1
					Accuracy						
ITL-L1	0.750	0.749	0.746	0.852	0.851	**0.853**	**0.941**	0.790	0.817	11	3
CTL-L1	**0.757**	0.759	**0.763**	0.852	0.847	0.849	0.938	0.850	0.827	6	2
cvxCMB-L1	0.754	0.759	**0.763**	0.852	0.847	0.849	0.935	0.850	0.826	7	2
cvxMTL-L1	0.753	**0.760**	**0.763**	**0.853**	**0.852**	**0.853**	0.933	**0.861**	**0.829**	5	1
ITL-L2	0.754	0.762	0.751	**0.856**	**0.855**	**0.856**	**0.942**	0.791	0.821	8	2
CTL-L2	**0.762**	0.765	**0.767**	0.854	0.853	0.851	0.933	0.853	0.830	3	1
cvxCMB-L2	0.757	0.764	0.766	0.854	0.853	0.853	0.934	0.853	0.829	4	1
cvxMTL-L2	0.753	**0.766**	0.766	**0.856**	**0.855**	**0.856**	0.933	**0.864**	**0.831**	1	1
ITL-LS	0.754	0.761	0.750	**0.851**	**0.850**	**0.851**	0.943	0.791	0.819	9	2
CTL-LS	**0.757**	**0.764**	0.766	0.845	0.847	0.842	0.914	0.750	0.811	12	3
cvxCMB-LS	0.754	**0.764**	0.765	0.849	**0.850**	0.848	0.925	0.793	0.818	10	3
cvxMTL-LS	**0.757**	**0.764**	**0.767**	**0.851**	**0.850**	**0.851**	**0.944**	**0.858**	**0.830**	2	1

test for statistical significance, a Wilcoxon signed rank test is applied to the absolute test error distributions of the models. Doing so over all possible model pairs would result in a hard to represent 12×12 (symmetric) table. Because of this, the test results are first sorted for each problem in ascending order for MAE and in descending order for R2, and then Wilcoxon test is applied for each consecutive model pair. The rankings in parenthesis of the figures represent these results, so a model precedes immediately another if the null hypothesis of both error distributions being the same is rejected at the 5% level. For instance, the MAE results in Table 3 show that for the `majorca` dataset, the best model is the convex cvxMTL-LS proposal and the second best is the cvxMTL-L2 model, while the cvxMTL-L1 ties for fifth place with cvxCMB-L1.

While the tables do not show a single model as the overall winner, it is clear that convex MTL models usually perform better. For instance, the MAE results in Table 3 show that a convex MTL model is best in four of the six datasets, ties for first place for the `boston` problem and only for `tenerife` the first convex model, cvxMTL-L1, occupies the second place. The situation is similar for the R2 results in Table 3, as well as for MAE and R2 scores in Table 4.

Table 5 shows the F1 and accuracy scores of the classification problems computed by the nested three fold CV described above. Here we omit the standard deviations, first to make the Table easier to read and, also, as their values are rather small. The Table also gives for each model the means across all problems of the F1 and accuracy scores as well as a ranking of all models according to these mean values. This ranking is given basically for illustration purposes, as it does not have a statistical significance. In any case, it can be observed than, for both F1 and accuracy, the best mean score in each of the L1, L2 and LS groups is obtained by the multi-task model. Moreover the best mean scores overall are those of the cvxMTL-L2 model.

In contrast with the situation in regression, where we applied Wilcoxon tests on the absolute error distribution of each model, such an approach cannot be used here. In order to provide some significance related results, we have proceed as follows. Recall that the scores are computed as the mean of the three test folds considered in our nested CV approach. When these are considered for the eight classification problems considered, we have for each model a sample of twenty-four F1 or accuracy scores. We have applied a Wilcoxon test within each model group over these samples, pairing consecutively the models' scores according to previous ranking as considered on each group (that is, for the F1 score in the L1 group, we pair first the samples of cvxMTL-L1 and cvxCMB-L1, then those of cvxCMB-L1 and CTL-L1 and, finally, those CTL-L1 and ITL-L1). As it can be seen in Table 5, cvxMTL models always appear in the first position, sometimes tied with another model. In any case, and again, this must be taken as an illustration, as here the application of the Wilcoxon test is not completely justified.

7. Conclusions

In this work, a convex multi-task L1-SVM approach has been extended to consider also two other widely used SVM models, the L2-SVM and the LS-SVM. As it was the case with the L1-SVM, both multi-task models can be reduced to single-task ones with a special multi-task kernel (provided that only a common bias is used); hence they can be directly solved with the state-of-the-art approaches for L1-, L2- and LS-SVMs. On the other side, a baseline multi-task model has been proposed: the optimal convex combination of the prediction of a common SVM, trained over all the task at the same time, with those of independent task-specific models. In particular, a procedure to optimize the parameter to combine any two models is defined to minimize the L1, L2, hinge and squared-hinge error functions.

The three proposed multi-task SVMs have been compared with the optimal convex combination, and the common and the independent models, over both regression and classification datasets. These experiments show how the multi-task approach performs generally better than the other methods.

In any case, there is room for further work. For instance, one of the drawbacks of these MTL models is the number of hyperparameters to be tuned. Indeed, although in this work a single mixing hyperparameter λ is used for the multi-task model, different values λ_r could be used for the different tasks, hence allowing for more flexible models, at the expense of adjusting all these hyperparam-

eters. Therefore, an immediate line of extension of this work consists in finding alternative ways to explore these task-specific hyperparameters in such a way that doesn't make the cross validation effort too expensive. As a first approach, using the optimal combining parameter λ_r^* as the hyperparameter of the multi-task models should be studied for which it could be interesting to use other black-box optimization methods to determine all these hyperparameters, such as Bayesian searches or evolutionary algorithms. As a more elaborated alternative, a way of optimizing the mixing hyperparameter jointly with the model, hence converting it into a standard model parameter, could be explored. Finally, and as mentioned for the soft sharing approach to deep MTL models, the distances between each task weights could be added to the regularization term so that more direct interactions between task models can be achieved.

CRediT authorship contribution statement

Carlos Ruiz: Conceptualization, Methodology, Validation, Formal analysis, Investigation, Writing - original draft, Writing - review & editing. **Carlos M. Alaíz:** Conceptualization, Methodology, Validation, Formal analysis, Investigation, Writing - original draft, Writing - review & editing. **José R. Dorronsoro:** Conceptualization, Methodology, Validation, Formal analysis, Investigation, Writing - original draft, Writing - review & editing.

Declaration of Competing Interest

The authors declare that they have no known competing financial interests or personal relationships that could have appeared to influence the work reported in this paper.

Acknowledgments

With partial support from Spain's grant TIN2016-76406-P. Work supported also by the UAM–ADIC Chair for Data Science and Machine Learning. We thank Red Eléctrica de España for making available solar energy data and the Agencia Estatal de Meteorología, AEMET, and the ECMWF for access to the MARS repository. We also gratefully acknowledge the use of the facilities of Centro de Computación Científica (CCC) at UAM.

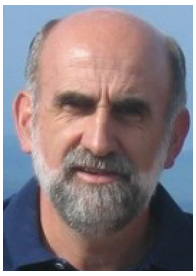
References

[1] R. Caruana, Multitask learning, Mach. Learn. 28 (1) (1997) 41–75.
[2] Y. Zhang, Q. Yang, A survey on multi-task learning, CoRR abs/1707.08114..
[3] S. Ruder, An overview of multi-task learning in deep neural networks, CoRR abs/1706.05098..
[4] T. Evgeniou, M. Pontil, Regularized multi–task learning, in, in: Proceedings of the Tenth ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, ACM, 2004, pp. 109–117.
[5] L. Liang, F. Cai, V. Cherkassky, Predictive learning with structured (grouped) data, Neural Networks 22 (5–6) (2009) 766–773.
[6] M. Donini, D. Martinez-Rego, M. Goodson, J. Shawe-Taylor, M. Pontil, Distributed variance regularized multitask learning, in: 2016 International Joint Conference on Neural Networks (IJCNN), IEEE, 2016, pp. 3101–3109.
[7] F. Cai, V. Cherkassky, SVM+ regression and multi-task learning, in: Proceedings of the 2009 International Joint Conference on Neural Networks, IEEE Press, Piscataway, NJ, USA, 2009, pp. 503–509.
[8] F. Cai, V. Cherkassky, Generalized SMO algorithm for SVM-based multitask learning, IEEE Trans. Neural Netw. Learning Syst. 23 (6) (2012) 997–1003.
[9] C. Ruiz, C.M. Alaíz, J.R. Dorronsoro, A convex formulation of svm-based multitask learning, in: Proceedings of the Hybrid Artificial Intelligent Systems - 14th International Conference, HAIS 2019. Lecture Notes in Computer Science, Vol. 11734, Springer, 2019, pp. 404–415..
[10] C.J.C. Burges, A tutorial on support vector machines for pattern recognition, Data Min. Knowl. Discov. 2 (2) (1998) 121–167.
[11] J.A.K. Suykens, T.V. Gestel, J.D. Brabanter, B.D. Moor, J. Vandewalle, Least Squares Support Vector Machines, World Scientific, 2002.
[12] C.E. Rasmussen, C.K.I. Williams, Gaussian processes for machine learning, in: Adaptive computation and machine learning, MIT Press, 2006.
[13] S.S. Keerthi, K. Duan, S.K. Shevade, A.N. Poo, A fast dual algorithm for kernel logistic regression, Mach. Learn. 61 (1–3) (2005) 151–165.
[14] D.H. Wolpert, W.G. Macready, No free lunch theorems for optimization, IEEE Trans. Evol. Comput. 1 (1) (1997) 67–82.
[15] M. Kubat, R.C. Holte, S. Matwin, Learning when negative examples abound, in: M. van Someren, G. Widmer (Eds.), Machine Learning: ECML-97, 9th European Conference on Machine Learning, Prague, Czech Republic, April 23–25, 1997, Proceedings, Vol. 1224 of Lecture Notes in Computer Science, Springer, 1997, pp. 146–153..
[16] M.F. Delgado, E. Cernadas, S. Barro, D.G. Amorim, Do we need hundreds of classifiers to solve real world classification problems?, J. Mach. Learn. Res. 15 (1) (2014) 3133–3181.
[17] S. Xu, X. An, X. Qiao, L. Zhu, Multi-task least-squares support vector machines, Multim. Tools Appl. 71 (2) (2014) 699–715.
[18] Y. Zhang, Q. Yang, A survey on multi-task learning, CoRR abs/1707.08114..
[19] M. Crawshaw, Multi-task learning with deep neural networks: A survey, CoRR abs/2009.09796..
[20] Z. Zhang, P. Luo, C.C. Loy, X. Tang, Facial landmark detection by deep multi-task learning, in: D.J. Fleet, T. Pajdla, B. Schiele, T. Tuytelaars (Eds.), Computer Vision - ECCV 2014–13th European Conference, Zurich, Switzerland, September 6–12, 2014, Proceedings, Part VI, Vol. 8694 of Lecture Notes in Computer Science, Springer, 2014, pp. 94–108..
[21] L. Duong, T. Cohn, S. Bird, P. Cook, Low resource dependency parsing: Cross-lingual parameter sharing in a neural network parser, in: Proceedings of the 53rd Annual Meeting of the Association for Computational Linguistics and the 7th International Joint Conference on Natural Language Processing of the Asian Federation of Natural Language Processing, ACL 2015, July 26–31, 2015, Beijing, China, Volume 2: Short Papers, The Association for Computer Linguistics, 2015, pp. 845–850..
[22] Y. Gao, J. Ma, M. Zhao, W. Liu, A.L. Yuille, NDDR-CNN: layerwise feature fusing in multi-task cnns by neural discriminative dimensionality reduction, in: IEEE Conference on Computer Vision and Pattern Recognition, CVPR 2019, Long Beach, CA, USA, June 16–20, 2019, Computer Vision Foundation/ IEEE, 2019, pp. 3205–3214..
[23] S. Ruder, J. Bingel, I. Augenstein, A. Søgaard, Sluice networks: Learning what to share between loosely related tasks, CoRR abs/1705.08142..
[24] I. Misra, A. Shrivastava, A. Gupta, M. Hebert, 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, IEEE Computer Society, 2016, pp. 3994–4003..
[25] V. Vapnik, A. Vashist, A new learning paradigm: Learning using privileged information, Neural networks 22 (5–6) (2009) 544–557.
[26] C.-C. Chang, C.-J. Lin, LIBSVM: a library for support vector machines, ACM Trans. Intell. Systems Technol. (TIST), 2, 2011, p. 27.
[27] L. Liang, V. Cherkassky, Connection between SVM+ and multi-task learning, in: Neural Networks, 2008. IJCNN 2008. (IEEE World Congress on Computational Intelligence). IEEE International Joint Conference on, IEEE, 2008, pp. 2048–2054..
[28] A. Argyriou, T. Evgeniou, M. Pontil, Convex multi-task feature learning, Mach. Learn. 73 (3) (2008) 243–272.
[29] L. Oneto, M. Donini, A. Elders, M. Pontil, Taking advantage of multitask learning for fair classification, in: V. Conitzer, G.K. Hadfield, S. Vallor (Eds.), Proceedings of the 2019 AAAI/ACM Conference on AI, Ethics, and Society, AIES 2019, Honolulu, HI, USA, January 27–28, 2019, ACM, 2019, pp. 227–237..
[30] C.-J. Lin, On the convergence of the decomposition method for support vector machines, IEEE Trans. Neural Networks 12 (6) (2001) 1288–1298

.

Carlos Ruiz is currently a PhD candidate at the Autonomous University of Madrid. He obtained the Degrees in Computer Science and Mathematics at the Autonomous University of Madrid in 2017, and the Master's Degree in Master's Degree in ICT Research and Innovation in 2018. His main research interests are Support Vector Machines and Multi-Task Learning.

Carlos M. Alaíz (PhD, Universidad Autónoma de Madrid; Spain) is Assistant Professor at Universidad Autónoma de Madrid, Spain. He received from this university his degrees in Computer Engineering and in Mathematics in 2008, his MSc degrees in Computer Science and Telecommunications, and in Mathematics and Applications, in 2010, and his PhD in 2014. He has also been Visiting Lecturer Professor at the Universidad Carlos III de Madrid in 2015, a postdoctoral researcher at KU Leuven (Belgium) between 2015 and 2017, and a post-doctoral researcher at Universidad Autónoma de Madrid between 2017 and 2018. His main research is focused on convex optimization, regularized learning and kernel methods, but also covers additional fields in machine learning and pattern recognition.



J.R. Dorronsoro received the Licenciado en Matemáticas degree from the Universidad Complutense de Madrid in 1977 and the Ph.D. degree in Mathematics from Washington University in St. Louis in 1981.

Currently he is Full Professor in the Computer Engineering Department of the Universidad Autónoma de Madrid, of which he was head from 1993 to 1996. He has also been UAM's vicerector for Innovation and Technology Transfer from 2009 to 2013.

He is also a Senior Researcher at UAM's Instituto de Ingeniería del Conocimiento.

His research interest are in Support Vector Machines, Neural Networks, Sparse Models and their applications in Data Science.