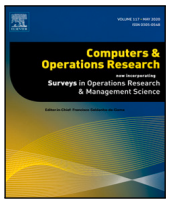
Computers & Operations Research 183 (2025) 107203

Contents lists available at ScienceDirect

Computers and Operations Research

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



Probabilistic Vector Machines

A. Pedro Duarte Silva

Universidade Católica Portuguesa, Católica Porto Business School and CEGE, Rua Diogo Botelho 1327, Porto, 4169-005, Portugal

ARTICLE INFO

MSC:
4904
49N10
6008
6204
6207
62H30
65K05

Keywords:
Support vector machines
Classification
Supervised learning
Multiclass probabilities

ABSTRACT

This paper proposes a novel Support Vector Machine (SVM) methodology for finding accurate probabilities of class memberships in supervised classification problems. Classical SVMs do not complement their class predictions with reliable confidence measures for each class assignment. For two-class problems this problem can be overcome by combining a sequence of weighted SVMs predictions into consistent class probabilities. In this work we show how a smart use of mathematical programming models can be used to extend this approach to the general multi-class classification problem. Previous attempts to tackle this problem either do not scale well with the number of different classes, or rely on sub-optimal partition strategies. Numerical experiments reveal the good scaling properties of the proposal, and the relative advantages of its class probability estimates over alternative approaches.

1. Introduction

In supervised classification problems, it is often the case that while some examples can be predicted with high confidence, for others any prediction is more uncertain and several different classes are almost as equally likely. In fact, finding reliable measures of class membership is a critical problem that complements the original, hard classification one, with necessary additional information. However, while traditional statistical methodologies typically address both problems, modern machine learning methods tend to overlook the second one and, however successful at making hard predictions, usually do not unleash their potential for building reliable class probability estimates. In this paper we will address this issue directly, focusing on Support Vector Machine (SVM) based methodologies. In particular we will show how, by smartly combining SVMs with mathematical programming techniques, the strengths of model-free machine learning approaches can be extended to the problem of class probability estimation.

SVMs were originally designed to handle two-class supervised classification problems, and have quickly established themselves as one of the most accurate methodologies for class prediction. However, this success did not translate to the related task of deriving confidence measures for these predictions. In fact, Lin (2002) has shown that, by targeting directly classification boundaries, standard SVMs do not carry much further information about class probabilities other than the predicted class by itself. Nevertheless, Lin et al. (2002) showed that, by appropriately modifying (weighting) the loss function used in standard SVMs, nonstandard SVMs can estimate consistently a theoretical Bayes rule for any arbitrary setting of class probabilities. Based on this property, Wang, Shen and Liu (2008) proposed to solve sequences of nonstandard SVMs with varying weight specifications, and to recover class probabilities from the frontiers between regions of the weights domain that lead to different predictions. This paper focus on extensions of this idea to the general k-class supervised classification problem.

There are two main alternatives to extend two-class SVMs to general k-class problems. The first one, known as the *all-in-one* approach, solves a single optimization problem that replaces the loss criterion used in two-class SVMs by an appropriate multiclass loss. The second alternative decomposes the original problem into several two-class problems using either an *one-against-one* or an *one-against-the-rest* decomposition strategy. However, the first approach is usually considered to be theoretically more sound, and there are some statistical evidence showing that it is also more stable in providing accurate classification results across a wide range of data conditions (Dogan et al., 2016).

The first proposal to extend (Wang et al., 2008) method to the general k-class problems was an *all-in-one* approach due to Wu, Zhang and Liu (WZL) (2010). However, in order to move from two-class to k-class problems, two difficulties needed to be resolved. Firstly, in general k-class problems the recovery of class probabilities from the solutions of weighted SVMs is not straightforward, and direct generalizations of the corresponding two-class procedure may lead to unstable

E-mail address: psilva@ucp.pt.

https://doi.org/10.1016/j.cor.2025.107203
Received 20 September 2024; Received in revised form 5 March 2025; Accepted 1 July 2025
Available online 15 July 2025
0305-0548/© 2025 The Author. Published by Elsevier Ltd. This is an open access article under the CC BY-NC-ND license (http://creativecommons.org/licenses/by-nc-nd/4.0/).

estimates. To correct this problem, Wu et al. (2010) proposed a semi-analytical indirect procedure based on the observed class prediction frequencies, and applied it successfully to several 3-class and 5-class problems. However, in this procedure, the number of base weighted SVMs that need to be trained increases exponentially with the number of classes, and we are not aware of any application of this approach to problems with more than 5 classes. Secondly, consistency properties are not guaranteed for general k-class classification SVMs, and many of the best known multiclass SVMs do not satisfy it. Aware of this problem (Wu et al., 2010) proposed a SVM based on a new loss function that satisfies a relevant definition of multiclass consistency. However, unlike most existing multiclass SVMs, the training of the WZL SVM requires the optimization of a non-convex problem, which increases the computational challenges, and makes their method impractical for big, or even moderate, data problems.

Multiclass probability estimation based on pairwise *one-against-the-rest* weighted SVMs were proposed by Xu and Wang (2013) (XW), and Wang, Zhang and Wu (2019) (WZW). The XW method implicitly assumes an order between the classes, and relies of estimates of the ratio between probabilities of consecutive classes. Therefore, this proposal seems to be more relevant for ordinal classification problems than for the general multinominal k-class problem that is the main focus of this paper. In contrast, the WZW method makes no such assumption, and can be regarded as a true multinomial method. Simulation results reported by Xu and Wang (2013) and Wang et al. (2019) suggest that the *one-against-the-rest* methods might have a slight statistical edge over the *all-in-one* WZL method. However, given the limited number of comparisons made, and the absence of results for the *all-in-one* approach in problems with larger k, it is difficult to make general assessments of the relative merits of the different SVM based proposals for multiclass probability estimation.

This paper proposes a novel *all-in-one* SVM approach to multiclass probability estimation, and compares its statistical performance against alternative estimators. In particular, we develop an improved method for recovering class probability estimates from weighted SVM predictions, and rely on base Fisher-consistent SVMs for which efficient training algorithms are available. In our approach, class probability estimates will be based on the solutions of linear programming models that optimize an l_1-norm measure of the agreement between predictions implied by the probability estimates and those made by weighted SVMs. One important advantage of this strategy is that the different weight specifications do not have to be uniformly distributed over a k-dimensional simplex, which allows for the creation of grids with satisfactory resolution, while ensuring that the number of required base weighted SVMs only grows linearly with the number of different classes. On the other hand, we employ base SVMs using an universal kernel without bias terms and the weighted loss proposed by Lee, Lin and Wahba (2004) (LLW). We note that the LLW loss leads to Fisher-consistent SVMs that can be trained by solving convex optimization problems. Furthermore, as remarked by Dogan et al. (2016), based on the theoretical results of Poggio et al. (2002), multiclass SVMs based on universal kernels without bias terms have similar statistical generalization properties as the corresponding SVMs with bias terms, and dropping these terms facilitates the use of state of art efficient decomposition training algorithms.

Based on these strategies, we were able the find reliable class probability estimates for problems with hundreds of examples and more than a dozen different classes. In order to achieve these results we rely on the smart use of mathematical programming tools, namely the careful choice, adaptation and improvement of the most efficient state of the art multi-class SVM training algorithms and, most importantly, the derivation of a novel way of converting weighted SVM predictions into probability estimates by the efficient optimization of relevant goodness of fit measures. We will call our proposed method Probabilistic Vector Machines, and will denote it by PVM.

Results from numerical comparisons show the computational feasibility of our method, and reveal that class probability estimation based on weighted SVMs are typically more accurate than competing distribution free machine learning approaches. Furthermore, our new estimators appear to be more reliable than those given by model based statistical methodologies when their assumptions fail. Amongst the SVM based methods, no alternative is universally superior to the others, and either *all-in-one* or pairwise *one-against-the-rest* strategies may give the best results depending on the particular data conditions at hand.

An R (R Core Team, 2025) package implementing the proposal described in this paper is currently under preparation, and we expect that it would be submitted to the Comprehensive R Archive Network (CRAN) in the near future.

The remainder of this paper is organized as follows. Section 2 introduces notation, reviews the base weighted multiclass SVM used in the paper, and presents the method of estimating class probabilities from weighted SVMs predictions. Section 3 describes the training algorithm used for the base SVMs. Section 4 presents numerical comparisons based on both controlled simulation experiments and real world benchmark data sets. Section 5 describes the most important conclusions of the paper, and gives directions for further research.

2. Multiclass probability SVMs

2.1. SVMs with Fisher consistent weighted losses

Let $\mathcal{T} = \{(\mathbf{x}_1, y_1), \ldots, (\mathbf{x}_n, y_n)\}$ be a training set of n examples, where $\mathbf{x}_i$ is an attribute descriptor belonging to some domain, $\mathcal{X}$, the label y_i is an integer belonging to the set $\mathcal{Y} = \{1, \ldots, k\}$, and all pairs $(\mathbf{x}_i, y_i)$ are independently generated from some unknown, but common, probability distribution, $P(\mathbf{X}, Y)$. Based on $\mathcal{T}$, we are interested in developing estimator functions $(\mathbf{p}_c(\mathbf{x}) \; ; \; c \in \mathcal{Y})$ for the *a-posterior* class probabilities

$$\mathbf{p}_c(\mathbf{x}) = P(Y = c | \mathbf{X} = \mathbf{x}) = \frac{P(\mathbf{x}, c)}{\sum_{c' \in \mathcal{Y}} P(\mathbf{x}, c')} \tag{1}$$

These estimators are to be recovered from a sequence of nonstandard (weighted) k-class SVMs yielding decisions rules with general form

$$\hat{y} = argmax_c \; \mathbf{f_c}(\mathbf{x}) \tag{2}$$

where $\mathbf{f_c}(.)$ is the cth element of the vector function $\mathbf{f} : \mathcal{X} \mapsto \mathbb{R}^k$ that solves the optimization problem

$$min_{\mathbf{f} \in \mathcal{F}^k} \quad n^{-1} \sum_{i=1}^{n} \boldsymbol{\pi}_{y_i} L(\mathbf{f}(\mathbf{x}_i), y_i) + \lambda \, J(\mathbf{f}) \tag{3}$$

$$subject \; to \qquad \sum_{c \in \mathcal{Y}} \mathbf{f}_c = \mathbf{0} \tag{4}$$

Here, $\mathcal{F}^k$ is a cartesian product of some known functional space $\mathcal{F}$, $J : \mathcal{F}^k \mapsto \mathbb{R}_0^+$ is an operator that penalizes function complexity, $\lambda \in \mathbb{R}^+$ is a regularization parameter that controls the trade-off between the smoothness of $\mathbf{f}$ and the multiclass large margin loss $L : \mathbb{R}^k \times \mathcal{Y} \mapsto \mathbb{R}_0^+$, and the weighting vector, $\boldsymbol{\pi}$, belongs to the k-dimensional simplex, $A_k = \{\boldsymbol{\pi} \in \mathbb{R}^k : \sum_{c=1}^{k} \boldsymbol{\pi}_c = 1 \; , \; \forall \; c \; \in \mathcal{Y}, \; \boldsymbol{\pi}_c \geq 0\}$.

The balancing constraint (4), ensures that problem (3)–(4) has an unique solution. For two-class problems this is equivalent to enforcing $\boldsymbol{f}_2 = -\boldsymbol{f}_1$. Then, (2) becomes the classical two-class decision rule $\hat{y} = (3 - sign(\mathbf{f_1}(\mathbf{x})))/2$, which is usually defined in terms of the scalar valued function $\mathbf{f}_1(.)$. In this case, $|\mathbf{f}_1(\mathbf{x}_i)|$ can be interpreted as the distance between example i and a separating hyperplane, in a classification relevant feature space (see e.g. Cristianini and Shawe-Taylor (2000)). Early SVM research interpreted these distances as heuristic measures of classification confidence and there were some attempts, notably by Platt (1999b), to convert them into posterior class probability estimates. However, Lin (2002) and Wang et al. (2008) argued that only the sign of $\mathbf{f}_1(.)$ is used by classification SVMs, and their absolute values do not carry much further information about class probabilities. That

observation is the fundamental insight that justifies (Wang et al., 2008) approach, extended here to the general multi class case, of estimating class probabilities by many weighted SVMs that only consider the class predicted by each SVM, but otherwise ignore the values of the corresponding $\mathbf{f}(.)$ functions.

In this paper we will study SVMs based on universal kernels (Steinwart, 2002), where $\mathcal{F}$ is a strictly positive definite Reproducing Kernel Hilbert Space (RKHS), $\mathcal{H}_{\mathbf{K}}$, induced by some known kernel function $K : \mathcal{X} \times \mathcal{X} \mapsto \mathbb{R}$, and endowed by the norm $\|.\|_{\mathcal{H}_{\mathbf{K}}}$ (Wahba, 1998, Cristianini and Shawe-Taylor, 2000, Poggio et al., 2002). Then, the representer theorem (Kimeldorf and Wahba, 1971) implies that for all $\mathbf{x} \in \mathcal{X}$ and $c \in \mathcal{Y}$, $\mathbf{f_c}(\mathbf{x})$ and $\|f_c\|^2_{\mathcal{H}_{\mathbf{K}}}$ can be expressed as $\mathbf{f_c}(\mathbf{x}) = \sum_{i=1}^n \theta_i^c \ K(\mathbf{x_i}, \mathbf{x})$, $\|f_c\|^2_{\mathcal{H}_{\mathbf{K}}} = \sum_{i=1}^n \sum_{j=1}^n \theta_i^c \ \theta_j^c \ K(\mathbf{x_i}, \mathbf{x_j})$, $\theta^c \in \mathbb{R}^n$, and the penalty $J(.)$ is typically chosen as the sum of the squared norms of the $\mathbf{f}$ components, *i.e.*, $J(\mathbf{f}) = \sum_{c=1}^k \|\mathbf{f}_c\|^2_{\mathcal{H}_{\mathbf{K}}}$.

The formulation above is based on the insight of Girosi (1986), showing that SVMs can be presented as particular cases of estimation regularization problems in functional analysis (Tikhonov, 1963, Bertero, 2006), where generalization properties of regularized estimators are justified by asymptotic arguments. However, Evgeniou et al. (2000) have shown in the case of SVMs, Vapnick's statistical learning theory (Vapnik, 1998) can be used to strength the known large sample generalization properties of these estimators, with the help of finite sample based error bounds. Particularly relevant for the SVMs considered in this paper are results of Guermeur (2017), Guermeur (2020) and Lei et al. (2019), respectively on data-independent and data-dependent, generalization error bounds for multiclass SVMs.

We note that this framework differs from the traditional one assumed in most SVMs, in that our classification functions $\mathbf{f}_c(.)$ do not include bias terms. This choice is justified by the theoretical results of Poggio et al. (2002), showing that for strictly positive definite RKHS, an arbitrary function, $\mathbf{f}(.)$, can always be approximated, to any given precision, by expansions such as $\mathbf{f}(\mathbf{x}) = \sum_i \theta_i \ K(\mathbf{x_i}, \mathbf{x})$, that do not use such terms. We note that this property does not hold for semi-positive definite or non-positive definite RKHS. These results are also the basis for Dogan et al. (2016) recommendation of dropping bias terms from standard multiclass SVMs, since this strategy can lead to substantial computational gains without affecting the main statistical properties of the resulting classifiers. We followed Dogan's recommendation and our numerical comparisons, to be described in Section 4, suggest that the resulting weighted SVMs are not adversely affected by it.

Different weighted versions of known multiclass SVMs can be specified as particular cases of decision rule (2) and optimization model (3) (4), by choosing particular loss functions. For instance, the SVMs proposed by Crammer and Singer (2001) (CS) and Lee, Lin and Wahba (2004) (LLW) are respectively defined by the following loss functions

$$L_{CS}(\mathbf{f}, y) = (1 - \max_{c \in \mathcal{Y}\setminus\{y\}} (\mathbf{f}_y - \mathbf{f}_c))_+ \tag{5}$$

$$L_{LLW}(\mathbf{f}, y) = \sum_{c \in \mathcal{Y}\setminus\{y\}} (\mathbf{f}_c + (k-1)^{-1})_+ \tag{6}$$

where $(u)_+ := max(0, u)$.

A desirable theoretical property of weighted multi-class SVMs is weighted Fisher-consistency, which is defined as follows

Definition 1 (*Weighted Fisher-consistency (**Wu, Zhang** and **Liu**)*). A multiclass functional margin based loss is weighted Fisher-consistent if the minimizer $\mathbf{f}^*$ of $E[\boldsymbol{\pi}_Y L(\mathbf{f}(\mathbf{X}), Y)|\mathbf{X} = \mathbf{x}]$ under (4), satisfies

$$argmax_{c \in \mathcal{Y}} \ \mathbf{f}^*_c(\mathbf{x}) = argmax_{c \in \mathcal{Y}} \ \boldsymbol{\pi}_c \ \mathbf{p}_c(\mathbf{x}) \quad \forall \mathbf{x} \in \mathcal{X}, \forall \boldsymbol{\pi} \in A_k$$

Wu et al. (2010) have shown that the loss assumed by Crammer and Singer is not weighted Fisher-consistent, but some truncated versions of it are. In particular, in their simulation studies, Wu et al. (2010) used the following Fisher-consistent loss

$$L_{WZL}(\mathbf{f}, y) = max[\max_{c \in \mathcal{Y}\setminus\{y\}} (1 - \mathbf{f}_y - \mathbf{f}_c)_+ \ , \ 1 + (k-1)^{-1}] \tag{7}$$

Table 1
Number of grid points (G) in uniformly distributed grids.

$d_{\boldsymbol{\pi}}$	$k = 3$	$k = 5$	$k = 10$	$k = 20$
0.05	231	10626	1.00×10^7	6.89×10^{10}
0.02	1326	316251	1.26×10^{10}	4.63×10^{16}
0.01	5151	4598126	4.26×10^{12}	4.91×10^{21}

On the other hand, Lee et al. (2004) proved that for any choice of an $\mathbf{C} \in (\mathbb{R}_0^+)^{k \times k}$ cost matrix, the minimizer of $\sum_{c \in \mathcal{Y}\setminus\{y\}} \mathbf{C}(y, c) \ (\mathbf{f}_c + (k-1)^{-1})_+$ under (4), is given by the vector f^{*1} with components

$$\mathbf{f}^{*\mathbf{1}}_{\mathbf{j}} = \begin{cases} 1 & \text{if } j = argmin_{c \in \mathcal{Y}} \sum_{c' \in \mathcal{Y}\setminus\{c\}} \mathbf{C}(c', c) \ p_{c'}(\mathbf{x}) \\ -(k-1)^{-1} & \text{otherwise} \end{cases}$$

In particular, noting that $argmin_{c \in \mathcal{Y}} \sum_{c' \in \mathcal{Y}\setminus\{c\}} \boldsymbol{\pi}_{c'} \ p_{c'}(\mathbf{x}) = argmin_{c \in \mathcal{Y}} \sum_{c' \in \mathcal{Y}} \boldsymbol{\pi}_{c'} \ p_{c'}(\mathbf{x}) - \boldsymbol{\pi}_c \ p_c(\mathbf{x}) = argmax_{c \in \mathcal{Y}} \ \boldsymbol{\pi}_c \ p_c(\mathbf{x})$, the choice of $\mathbf{C}$ as a matrix satisfying $\mathbf{C}(c', c) = \boldsymbol{\pi}_{c'} \ \forall c \neq c'$, leads to the minimizer, f^{*2}, defined as

$$\mathbf{f}^{*\mathbf{2}}_{\mathbf{j}} = \begin{cases} 1 & \text{if } j = argmax_{c \in \mathcal{Y}} \ \boldsymbol{\pi}_c \ p_c(\mathbf{x}) \\ -(k-1)^{-1} & \text{otherwise} \end{cases}$$

which implies the weighted Fisher-consistency of the LLW loss (6) in the Wu, Zhang and Liu sense.

In this paper we propose to estimate class probabilities from the class predictions given by weighted SVMs based on loss (6). One advantage of using this loss, is that the training of the resulting SVMs leads to convex optimization models, which are faster to solve than the non-convex models resulting from the WZL loss (7). In Section 3.1 we will describe an efficient algorithm to train the weighted SVMs proposed here.

2.2. SVM probability estimation

In order to estimate class probabilities from the predictions given by weighted SVMs, one needs to train several weighted SVMs with different weight specifications. Let $\mathcal{G}$ be the grid of $\boldsymbol{\pi}$ specifications, and $\boldsymbol{\pi}^g$ a generic element of $\mathcal{G}$. Wu et al. (2010) proposed to estimate $\mathbf{p}(\mathbf{x})$ by the $\mathbf{p}^*(\mathbf{x}) = (\mathbf{p}^*_1(\mathbf{x}), \mathbf{p}^*_2(\mathbf{x}), \ldots, \mathbf{p}^*_k(\mathbf{x}))$ solution of the system $h_c(\mathbf{p}_1(\mathbf{x}), \mathbf{p}_2(\mathbf{x}), \ldots, \mathbf{p}_k(\mathbf{x})) = prop_c(\mathbf{x})$; $c = 1, 2, \ldots, k$, where $h_c(\mathbf{p}(\mathbf{x}))$ represents the volume proportion of the A_k simplex where $\boldsymbol{\pi}_c \ \mathbf{p}_c(\mathbf{x}) \geq \boldsymbol{\pi}_{c'} \ \mathbf{p}_{c'}(\mathbf{x}) \forall c' \neq c$, and $prop_c(\mathbf{x})$ the observed proportion of weighted SVMs that assign an example described by $\mathbf{x}$ to class c. This method requires the $\boldsymbol{\pi}^g$ to be uniformly distributed over A_k which, for a grid step size $d_{\boldsymbol{\pi}}$, implies a number of grid elements equal to $G = \#\mathcal{G} = \sum_{c=1}^{k-1} \binom{d_{\boldsymbol{\pi}}^{-1}+1}{c} \binom{k-2}{c-1}$ (Casado et al., 2016). If follows that the number of base weighted SVMs to be trained increases exponentially with k. Table 1 shows some values of G for different combinations of $d_{\boldsymbol{\pi}}$ and k. It is clear that this method becomes computationally intractable for problems with moderate or large k, and we are not aware of any application of this approach to problems with more than 5 classes.

In this paper, we propose an alternative method to recover class probabilities from class predictions, where the grid of weigh specifications does not have to be uniformly distributed over A_k. Our approach is based on optimizing an heuristic l_1-norm measure of the agreement between the weighted SVMs predictions and the optimal classification rules implied by the $\mathbf{p}_c(\mathbf{x})$ estimates. In particular, consider the weight specification $\boldsymbol{\pi}^g$ and the corresponding weighted SVM prediction $\hat{y}_g$. Assuming the theoretical cost weighted classification problem, $min_{\mathbf{f}} \ E[\boldsymbol{\pi}^g_Y L(\mathbf{f}(\mathbf{X}), Y)|\mathbf{X} = \mathbf{x}] \ s.t. \sum_{c \in \mathcal{Y}} \mathbf{f}_c = \mathbf{0}$, the SVM prediction agrees with the optimal Bayes rule for a vector of *a-posteriori* probabilities, $\mathbf{p}_c(\mathbf{x})$, iff

$$\boldsymbol{\pi}^g_{\hat{y}_g} \mathbf{p}_{\hat{y}_g}(\mathbf{x}) \geq \boldsymbol{\pi}^g_c \mathbf{p}_c(\mathbf{x}) \qquad \forall c \in \mathcal{Y}\setminus\{\hat{y}_g\} \tag{8}$$

When it is possible to find $\mathbf{p}(\mathbf{x})$ vectors such that (8) is satisfied for all $\boldsymbol{\pi}^g \in \mathcal{G}$, a reasonable estimation criterion is to choose amongst such vectors, the one that maximizes the desirable sum of the l_1-norm

margin deviations $\boldsymbol{\pi}^g_{\hat{y}_g}\mathbf{p}_{\hat{y}_g}(\mathbf{x}) - \boldsymbol{\pi}^g_c\mathbf{p}_c(\mathbf{x})$. On the other hand, when it is not possible to find a vector estimate that always satisfies (8), one may search for one the minimizes the undesirable l_1-norm deviations $\boldsymbol{\pi}^g_c\mathbf{p}_c(\mathbf{x}) - \boldsymbol{\pi}^g_{\hat{y}_g}\mathbf{p}_{\hat{y}_g}(\mathbf{x})$. Putting these two goals together, we propose to search for the probability estimate that solves the following probability estimation problem

$$min_{\mathbf{p}(\mathbf{x})\in A_k} \sum_{\boldsymbol{\pi}^g\in\mathcal{G}} \sum_{c\in\mathcal{Y}\setminus\{\hat{y}_g\}} \eta\,(\boldsymbol{\pi}^g_c\,\mathbf{p}_c(\mathbf{x}) - \boldsymbol{\pi}^g_{\hat{y}_g}\mathbf{p}_{\hat{y}_g}(\mathbf{x}))_+ - \\ - \;(\boldsymbol{\pi}^g_{\hat{y}_g}\mathbf{p}_{\hat{y}_g}(\mathbf{x}) - \boldsymbol{\pi}^g_c\mathbf{p}_c(\mathbf{x}))_+ \tag{9}$$

$$subject\ to \quad \mathbf{p}_c(\mathbf{x}) \geq \epsilon \qquad \forall c\in\mathcal{Y} \tag{10}$$

where $\eta > 1$ is an hyper-parameter that controls the trade-off between the desirable and undesirable deviations, and ϵ is either zero or a small positive constant. In all the experiments reported in this paper we set $\epsilon = d_{\boldsymbol{\pi}}/2$, reflecting the numerical precision implied by the grid resolution. Setting $\epsilon = 0$ is equivalent of assuming that, for same values of the attribute vector $\mathbf{x}$, membership in some classes may be impossible. In general, that is not a sensible assumption, and we recommend that, unless there are good reasons to believe that under some conditions one (or several) classes can never occur, ϵ should be set to a strictly positive constant, enforcing the strict positivity of $\mathbf{p}(\mathbf{x})$.

Whether or not it is possible to find a vector estimate that always satisfies (8) depends on the relation between the grid resolution and the sample size. For a fixed grid, as the sample size grows that probability approaches one from below. However, as the precision of the $\mathbf{p}(\mathbf{x})$ estimates also depends on the grid resolution, there is a trade-off between this resolution and the likelihood of always enforcing (8). We recommend that $d_{\boldsymbol{\pi}}$ should be chosen as a function of the sample size, so that larger samples can use grids with better resolution, even if that choice leads to some (8) violations by a few examples. In the experiments reported in this paper we set $d_{\boldsymbol{\pi}} = 0.2/\sqrt{n}$, which appears to be a good compromise regarding this trade-off.

The optimization of l_1-norm measures similar to the one used in (9)–(10) has been widely studied in the Operations Research literature on supervised classification (see Duarte Silva (2017)), where it has been shown that the resulting problems can be solved by optimizing straightforward linear programming models. In particular, the $\mathbf{p}_c(\mathbf{x})$ estimates resulting from (9)–(10) can be recovered from the solutions of its dual problem, which is given by:

Proposition 1. *The dual of problem*(9)–(10) *is*

$$max_{\boldsymbol{\mu}\in\Re^{G\times(k-1)},\,\xi\in\Re^k,\,\nu,\in\Re} \left(\nu + \epsilon\sum_{c\in\mathcal{Y}}\xi_c\right) \tag{11}$$

subject to

$$\sum_{g=1:\hat{y}_g\neq c}^{G}\boldsymbol{\pi}^g_c\boldsymbol{\mu}_{gh_g(c)} - \sum_{g=1:\hat{y}_g=c}^{G}\sum_{c'\in\mathcal{Y}\setminus\{\hat{y}_g\}}\boldsymbol{\pi}^g_{\hat{y}_g}\,\boldsymbol{\mu}_{gh_g(c')} + \nu + \xi_c \leq 0 \quad \forall c\in\mathcal{Y} \tag{12}$$

$$1 \leq \boldsymbol{\mu}_{gh_g(c)} \leq \eta \qquad g = 1,\ldots,G; c\in\mathcal{Y}\setminus\{\hat{y}_g\} \tag{13}$$

$$\xi_c \geq 0 \qquad \forall c\in\mathcal{Y} \tag{14}$$

where

$$h_g(c) = \begin{cases} c, & c < \hat{y}_g \\ c-1, & c > \hat{y}_g \end{cases}$$

Proof. see Appendix A. □

We note that problem (11)–(14) has only k explicit linear constraints (excluding simple variable bounds), and can be solved efficiently by the simplex algorithm for bounded variables (Bazaraa et al., 2011).

One important advantage of this method is the fact that it does not require $\boldsymbol{\pi}^g$ to be uniformly distributed over A_k, which allows for alternative ways of defining representative $\mathcal{G}$ grids with a considerable smaller number of grid points. We will describe one such alternative, where we look at one component of $\boldsymbol{\pi}$, say $\boldsymbol{\pi}_c$, at the time, and ensure that a resulting set of $\boldsymbol{\pi}^g$ specifications gives an adequate representation of the $(0,1)$ interval, while the remaining components of $\boldsymbol{\pi}$ are assigned at random. The details are provided in algorithm 1, where it is shown that the representation of the $(0,1)$ line is satisfied by setting $\boldsymbol{\pi}_c$ in turn to each of the $d_{\boldsymbol{\pi}}^{-1}$-1 uniformly distributed points of the $(0,1)$ line, with a distance of $d_{\boldsymbol{\pi}}$ between each point. The resulting number of grid points equals $G = \#\mathcal{G} = k\,d_{\boldsymbol{\pi}}^{-1} - k$, and only increases linearly with k. Table 2 shows the number of base SVMs that need to be trained in this approach for representative values of $d_{\boldsymbol{\pi}}$ and k. The contrast with the corresponding figures shown before in Table 1 is remarkable.

Summing it up, algorithm 2 summarizes the major steps required to implement the proposal made in this paper.

Algorithm 1 Generating grid of weight vectors

Parameters: k, $d_{\boldsymbol{\pi}}$
Output: A $\mathcal{G}$ grid of weight vectors

Let $pbc = d_{\boldsymbol{\pi}}^{-1} - 1$.
Let $\mathcal{G} = \emptyset$, $g = 0$.
for $c \in \mathcal{Y}$ **do**
  for $a = 1$ to pbc **do**
    Let $g = g + 1$.
    Initialize the k-dimensional vector $\boldsymbol{\pi}^g$.
    Let $\boldsymbol{\pi}^g_c = a\,d_{\boldsymbol{\pi}}$.
    Generate independently k-1 uniform random numbers $(U_{c'}, c' \in \mathcal{Y}\setminus\{c\})$ on the unit interval.
    Let $SU = \sum_{c'\in\mathcal{Y}\setminus\{c\}} U_{c'}$.
    for $c' \in \mathcal{Y}\setminus\{c\}$ **do**
      Let $\boldsymbol{\pi}^g_{c'} = U_{c'}(1 - \boldsymbol{\pi}^g_c)/SU$.
    end for
    Add $\boldsymbol{\pi}^g$ to $\mathcal{G}$.
  end for
end for
return $\mathcal{G}$.

Table 2
Number of grid points required by PVM.

$d_{\boldsymbol{\pi}}$	$k=3$	$k=5$	$k=10$	$k=20$
0.05	57	95	190	380
0.02	147	245	450	980
0.01	297	495	990	1980

Algorithm 2 Main probability estimation algorithm

Input: A training set $\mathcal{T}$, and a estimation set $\mathcal{E}$
Output: The set $\mathcal{P}$, of probability vectors for $\mathcal{E}$.

Let $\mathcal{P} = \emptyset$.
Define a grid $\mathcal{G}$ of weight specifications using algorithm 1.
for $\boldsymbol{\pi}^g \in \mathcal{G}$ **do**
  Solve optimization model (3)-(4) using loss (6).
  for $e \in \mathcal{E}$ **do**
    Predict and save a class for entity e based on the $\boldsymbol{\pi}^g$ weighted SVM.
  end for
end for
for $e \in \mathcal{E}$ **do**
  Solve optimization model (9) - (10) and add its solution to $\mathcal{P}$.
end for
return $\mathcal{P}$.

Algorithm 3 Canonical convex quadratic SVM training algorithm

Input: A vector $\mathbf{v}$, and matrix $\mathbf{Q}$, of coefficients, and two vectors, $\mathbf{l}$ and $\mathbf{u}$, of lower and upper bounds.
Output:The vector, $\boldsymbol{\gamma}$, that maximizes (18) subject to (19).

1: Set $\boldsymbol{\gamma}$ to an initial feasible solution.
2: **repeat**
3: Choose a working set formed by a small number of $\boldsymbol{\gamma}$ components, that promise the greatest increase in (18) subject to (19).
4: Find the analytical solution of problem (18)-(19), restricted to the components of the working set.
5: **until** No further working set selection leads to an improvement of (18) subject to (19).
6: **return** $\boldsymbol{\gamma}$.

3. Learning algorithms

3.1. SVM optimization

The state of art algorithms for training SVMs are based on decomposition strategies to solve dual formulations of the associated optimization problems. In the case of two-class problems, a standard reference is Platt's Sequential Minimal Optimization (SMO) algorithm (Platt, 1999a) which decomposes a large convex quadratic optimization problem into a sequence of two-dimensional quadratic problems that can be solved analytically. This algorithm was adapted by Dogan et al. (2011) to solve the most common k-class SVMs, including those based on the unweighted LLW loss. This adaptation was implemented in the public-domain C++ software library Shark (Igel et al., 2008). It is straightforward to show that this approach also applies to SVMs using the weighted LLW loss. In particular:

Proposition 2. *The dual of problem (3)–(4) with loss (6) is*

$$max_{\boldsymbol{\alpha}\in\mathbb{R}^{n\times k}} \frac{1}{k-1}\sum_{i\in\mathcal{T}}\sum_{c\in\mathcal{Y}\backslash\{y_i\}}\boldsymbol{\alpha}_{ic} - \\ -\frac{1}{2\lambda}\sum_{i,i'\in\mathcal{T}}K(\mathbf{x}_i,\mathbf{x}_{i'})\sum_{c,c'\in\mathcal{Y}}(\delta_{cc'}-\frac{1}{k})\,\boldsymbol{\alpha}_{ic}\,\boldsymbol{\alpha}_{i'c'} \quad (15)$$

$$subject\ to \quad 0 \le \boldsymbol{\alpha}_{ic} \le \frac{\boldsymbol{\pi}_{y_i}}{n} \qquad i\in\mathcal{T},\ c\in\mathcal{Y}\backslash\{y_i\} \quad (16)$$

$$\boldsymbol{\alpha}_{iy_i} = 0 \qquad i\in\mathcal{T} \quad (17)$$

where $\delta_{cc'} = I(c=c')$ is the Kronecker delta.

Proof. see Appendix B. □

Problem (15)–(17) can be expressed as a particular case of the canonical convex quadratic SVM training model studied by Dogan et al. (2011), which has the following general form

$$max_{\boldsymbol{\gamma}\in\mathbb{R}^m} \quad \boldsymbol{v}^T\,\boldsymbol{\gamma} - \frac{1}{2}\boldsymbol{\gamma}^T\,\boldsymbol{Q}\,\boldsymbol{\gamma} \quad (18)$$

$$subject\ to \quad \mathbf{l} \le \boldsymbol{\gamma} \le \mathbf{u} \quad (19)$$

For that purpose, first notice that, given (17), the matrix $\boldsymbol{\alpha}$ has only $m = n(k-1)$ free elements. Therefore, $\boldsymbol{\gamma}$ can be defined as the m-dimensional vector collecting these elements, $\mathbf{l} = \mathbf{0}_m$ as a null vector, and $\mathbf{u}, \mathbf{v}$ and $\mathbf{Q}$ as vectors and matrix of upper bounds and coefficients, whose values can be easily derived from (15)–(17). Dogan et al. (2011) proposed to solve Problem (18)–(19) by a decomposition strategy outlined in algorithm 3.

The efficiency of this algorithm depends on some implementation details such as the choice of the initial solution and the working set selection. Although (Dogan et al., 2011) do not specifically mentioned how to choose the initial solution, an inspection of the Shark (Igel et al., 2008) source code reveals that in that implementation the initial $\boldsymbol{\gamma}$ is set to the lower bound vector $\mathbf{l}$. Regarding the working set selection, Dogan et al. (2011) discuss the trade-offs involved with the working set size, noting that smaller working sets allow for fast iterations with small improvements at each iteration, while larger sets allow for larger progress at each iteration, at the price of making each iteration computationally more expensive. They argued that the best trade-offs can be achieved for working sets that lead to relatively straightforward analytic solutions for each iteration subproblem, which restricted their choices to sets with just a few $\boldsymbol{\gamma}$ components. They recommended two-dimensional working sets as a good compromise in this regard. For the first element of this set, they choose the component, j, with maximal absolute value on the objective (18) gradient, $\mathbf{g} = \mathbf{v} - \mathbf{Q}\,\boldsymbol{\gamma}$. For the second element choice, they maximized the unconstrained gain, i.e. they chose the component that, knowing j, leads to the maximal increase in (18), ignoring the box constraints (19). Additional details can be found in Dogan et al. (2011), while in Dogan et al. (2016) it is argued that, under reasonable assumptions, the asymptotic time complexity of this algorithm equals the one required to solve Crammer and Singer's SVM. We note that the popularity of Crammer and Singer SVM is mostly due to the fact that this classifier is believed to be the fastest to train amongst all the *all-in-one* multiclass SVMs.

We have developed an independent implementation of algorithm 3, and tested several alternative implementation choices in order to improved its efficiency. In particular, we have found that setting all the components of the initial $\boldsymbol{\gamma}$ solution to their upper bounds $\mathbf{u}$, tends to improve the algorithm, often leading to a dramatic reduction in the number of required iterations. Furthermore, while we have confirmed that two-dimensional working sets with the first element chosen by the maximal absolute gradient gives good results, we have found a more effective rule for the choice of its second element. In particular, we have noticed that in most of the algorithm iterations the unconstrained subproblem maximum is infeasible, and the overwhelming majority of the iterations lead to jumps between upper and lower bounds (or vice-versa) of the $\boldsymbol{\gamma}$ components. That being the case, we chose to select the second element of the working set by maximizing the constrained gain, which is obtained by checking the variation on (18) when the $\boldsymbol{\gamma}$ components are changed to their lower and/or upper bounds, instead of moving them to the unrestricted maximizer of (18). This choice leads to further reductions in the overall training computational effort. The C++ source code of our implementation will be made publicly available, as part of an R package providing the proposal made in this paper, that we intend to submit to the official R repository (CRAN) in the near future.

3.2. Parameter tuning

The algorithm described in the previous subsection depends on the value of several hyper-parameters, namely the regularization parameter λ in model (3)–(4), the desirable-undesirable deviations trade-off parameter η in model (9)–(10), and any parameters associated with the chosen kernel. The traditional way to tune hyper-parameters in machine learning algorithms is to search amongst a grid of possible values for those that optimize a relevant measure of performance in one, or several, tuning data sets. Following (Wu et al., 2010), in this work we will use the tuning log-likelihood, $\ln L = \sum_i \ln \hat{\mathbf{p}}_{y_i}(\mathbf{x}_i)$ as performance criterion.

The strategy described above is particularly important for critical hyper-parameters, such as λ, whose optimal values are strongly dependent on the data characteristics. However, sensible values for other hyper-parameters might not be so data dependent, and a multidimensional search for a large number of hyper-parameters can be computationally too demanding. Furthermore, for some parameters, data dependency can sometimes be incorporated by pre-computed functions that do not require expensive searches. In particular, for the case of classification SVMs based on the Gaussian kernel, $K(\mathbf{x}_i,\mathbf{x}_{i'}) = exp^{-\frac{1}{2}\|\mathbf{x}_i,-\mathbf{x}_{i'}\|_2^2/\sigma^2}$, Caputo et al. (2002) have argued that sensible choices

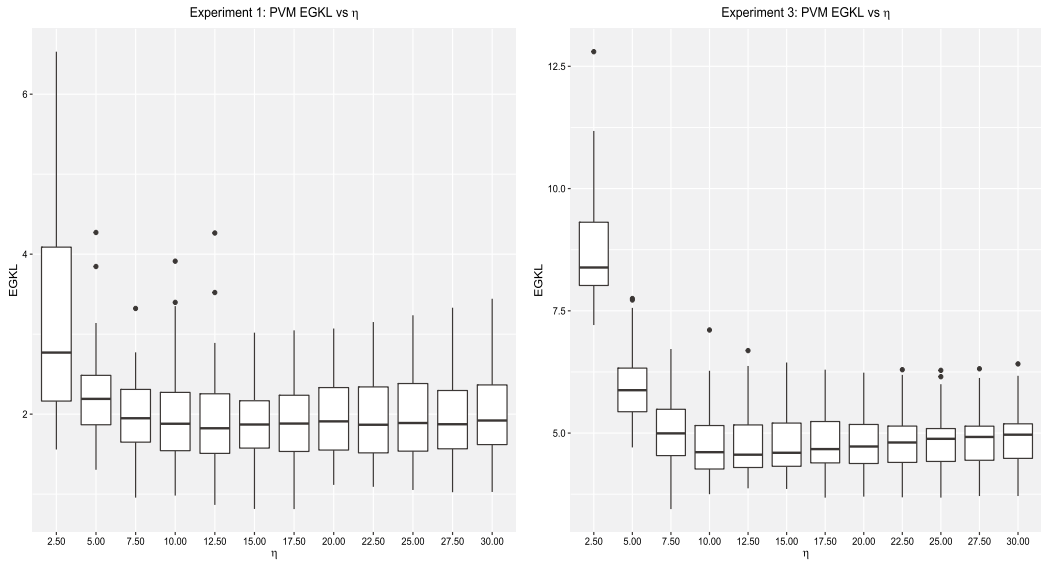


Fig. 1. PVM dependence on η.

for σ^2 depend mostly on the quantiles of the distribution of the pairwise distances $d^2_{ii'} = \|\mathbf{x}_i - \mathbf{x}_{i'}\|^2_2 \quad i, i' \in \mathcal{T} \quad i \neq i'$, with any value between the corresponding 10th and 90th percentiles yielding essentially the same results. We propose, for SVMS with Gaussian kernels, to choose σ^2 as the median of the $d^2_{ii'}$ distribution. Furthermore, we conducted some simulation experiments to study the dependence of our proposal to the value of the η parameter, and found that for $\eta \geq 10$ our method is not particularly sensitive to this choice, with any constant ranging from 10 to 20 leading to sensible, and similar, results. Fig. 1 illustrates this point with boxplots of the Empirical Generalized Kullback–Leibler measure, $EGKL = \frac{1}{\#\mathcal{E}} \sum_{i \in \mathcal{E}} \sum_{c \in \mathcal{Y}} \mathbf{p}_c(\mathbf{x}_i) \ln \frac{\mathbf{p}_c(\mathbf{x}_i)}{\hat{\mathbf{p}}_c(\mathbf{x}_i)}$, distribution (over 50 independent replications) on validation data, for two of the data conditions considered in the experiments described in Section 4.1. Results for other conditions are similar. Therefore, we propose that η should be fixed to a constant within the $10 - 20$ range, and in the numerical comparisons presented in Section 4 we always set $\eta = 15$. With these choices, the only hyper-parameter that requires a data driven search is the λ regularization constant.

Another issue in hyper-parameter tuning is the selection of the tuning data where different parameter values are to be evaluated. The simplest, and computationally more efficient, way to proceed is to divide the original data into three roughly equally sized sets, and use one for training, another for parameter tuning, and the third one for algorithm evaluation. Two major shortcomings of this strategy are the reduction on the number of examples available for training, and an increased instability in the evaluation measure values, due to the use of a single tuning data set. These two problems are particularly serious if the number of original examples is small, but become milder for larger data sets. A common strategy to mitigate the above problems is to use s-fold cross-validation techniques, for instance, randomly dividing the data not used for evaluation into s different folds, and for $f = 1, \ldots, s$, use the fth fold for tuning and the remaining $s - 1$ folds for training. Then, the proposed algorithms can be evaluated by the average of the performance measure over these s different analyses. This strategy increases the size of the training sets, and improves the stability of the tuning procedure, at the price of increasing its computational effort. Trying to balance statistical performance against computational efficiency, we recommend that for small and moderate data sets an s-fold cross-validation strategy should be employed, while for large data sets a simple single division into training, tuning, and evaluation data may suffice. The concrete data size threshold that would determine the choice between these two strategies is application dependent, and should take into account the relative costs of the available computer resources versus the expected precision and stability losses associated with the use of smaller training sets, and an unique tuning set.

4. Numerical comparisons

In this section we illustrate the performance of this proposal, comparing it with seven alternative methods for 4 simulation scenarios and 4 benchmark data sets previously discussed in the literature. In all these comparisons we will assume that $\mathcal{X} = \mathbb{R}^p$, *i.e.* all examples can be described by p-dimensional vectors of real numbers.

The methods under comparison are: (i) three model based statistical methods, namely Multinomial Logistic Regression (MLR), Multiple Linear Discriminant Analysis (MLDA) and Multinomial Generalized Additive Models (MGAM); (ii) three SVM based methods, namely our Probability Vector Machines (PVM) proposal, and the two pairwise methods proposed by Wang, Zhang and Wu (WZW) and Xu and Wang (XW); (iii) two standard machine learning methods, namely classification trees (TREE) and Random Forests (RF).

Multinomial Logistic Regression is arguably the most important reference method in the classical statistical literature, and relies on mild assumptions about the distribution of $\mathbf{p}(\mathbf{x})$. In particular it assumes that $\ln \mathbf{p}_c(\mathbf{x})/\mathbf{p}_1(\mathbf{x}) = \beta^c_0 + \boldsymbol{\beta^c}^T \mathbf{x}$, $\beta^c{}_0 \in \mathbb{R}$, $\boldsymbol{\beta^c} \in \mathbb{R}^p$, $c = 2, \ldots, k$, estimating β^c_0 and $\boldsymbol{\beta^c}$ by maximum likelihood. Multiple Linear Discriminant Analysis relies on the somehow more stringent assumption of an homoscedastic Gaussian model, $\mathbf{x}|Y \sim N(\boldsymbol{\mu}_Y; \boldsymbol{\Sigma})$, which implies the same (linear) functional form for $\ln \mathbf{p}_c(\mathbf{x})/\mathbf{p}_1(\mathbf{x})$, but now with $\beta^c_0 = -1/2(\boldsymbol{\mu}_c + \boldsymbol{\mu}_1)^T \boldsymbol{\Sigma}^{-1}(\boldsymbol{\mu}_c - \boldsymbol{\mu}_1)$ and $\boldsymbol{\beta^c} = \boldsymbol{\Sigma}^{-1}(\boldsymbol{\mu}_c - \boldsymbol{\mu}_1)$. Under this assumption, β^c_0 and $\boldsymbol{\beta^c}$ are typically estimated by replacing population means, variances, and covariances by their sample counterparts in the formulae above (McLachlan, 2005). Modern approaches include Multinomial Generalized Additive Models (Yee and Wild, 1996) which assume instead that $\ln \mathbf{p}_c(\mathbf{x})/\mathbf{p}_1(\mathbf{x}) = \beta^c_0 + \sum_{j=1}^p f^c_j(\mathbf{x}_j)$, $c = 2, \ldots, k$ with $f^c_j(\mathbf{x}_j)$, $j = 1, \ldots, p$ being smooth functions, usually estimated by spline methods. Classification Trees (Breiman et al., 1984) and Random Forests (Breiman, 2001) are two well established machine learning methodologies for supervised classification.

We estimated the MLR and MLDA models using, respectively, the *multinom* and the *lda* functions of the *nnet* and *MASS* R packages (Venables and Ripley, 2002). In the MGAM models we employed cubic splines for all continuous attributes, linear functions for the discrete attributes, and relied on the *vgam* function of the *VGAM* package (Yee, 2010), with all remaining arguments at their default values. In the SVM based methods we used R code gently ceded by Xu and Wang, for XW

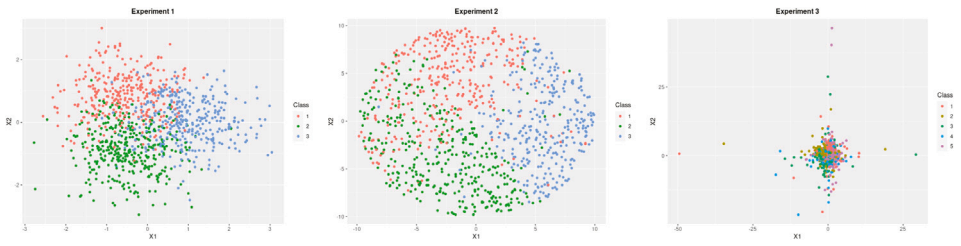


Fig. 2. Sample Data for Experiments 1, 2 and 3.

method, and our own implementations for the PVM and WZW methods. In these three methods we always employed a Gaussian kernel with the σ^2 hyper-parameter chosen as the median of the $d^2_{ii'}$ distribution, while the value of λ hyper-parameter was found by a two step search that tried to optimize, on tuning data, the performance measure suggested in each proposal. In the first step we searched for the λ_0 value that optimized this measure over the set $\{2^{5j} | j = -2, -1, 0, 1, 2\}$, and in the second step we refined the search, looking for its optimizer over $\{2^j \lambda_0 | j = -2, -1, 0, 1, 2\}$. For the PVM method, the grid step size was always set at $d_{\pi} = 0.2/\sqrt{n}$, and the η hyper-parameter at $\eta = 15$. For the Tree and Random Forest methods we relied on the *TREE* and *RF* functions of the *rpart* (Therneau and Atkinson, 2018) and *randomForest* (Liaw and Wiener, 2002) R packages, with all arguments set at their default values.

4.1. Controlled simulations

We consider four different experiments with different data conditions. The first two experiments use setups initially considered in Wu et al. (2010). Both these experiments are three-class problems, and the first one illustrates conditions in which the assumptions of MLR are satisfied, while in the second one the data was generated by highly non-linear functions that lead to a gross violation of these assumptions. The third and fourth experiments use setups initially considered in Xu and Wang (2013) in which the data was generated by heavy-tailed distributions that also violate MLR assumptions. These two experiments consider, respectively a five-class (3rd experiment) and a ten-class (4th experiment) problem. The details of the data generation are described in the paragraphs below.

Experiment 1. The first experiment uses the data conditions described in Example 1 of Wu et al. (2010), namely training samples of 400 examples with the Y class labels generated uniformly from $\mathcal{Y} = \{1, 2, 3\}$, and two-dimensional predictors generated conditionally on Y, from a Gaussian distribution with mean vector $\boldsymbol{\mu}(y) = [cos(2y\pi/3), sin(2y\pi/3)]^T$ and covariance matrix $\Sigma = 0.7^2\ \mathbf{I}_2$, with $\mathbf{I}_2$ being the two-dimensional identity matrix. The left panel of Fig. 2 displays a validation sample with 1000 examples generated from this data condition.

Experiment 2. The second experiment uses the data conditions described in Example 3 of Wu et al. (2010), namely training samples of 600 observations with two-dimensional predictors generated uniformly over the disk $\{\mathbf{x} : x_1^2 + x_2^2 \leq 100\}$, and class probabilities generated conditionally on $\mathbf{x}$ from $p_c(\mathbf{x}) = \exp(g_c(\mathbf{x}))/\sum_{c'=1}^{3} \exp(g_{c'}(\mathbf{x}))$, $c \in \mathcal{Y} = \{1, 2, 3\}$, where $g_1(\mathbf{x}) = \Phi^{-1}(T_2(-5\mathbf{x}_1\sqrt{3}+5\mathbf{x}_2))$, $g_2(\mathbf{x}) = \Phi^{-1}(T_2(-5\mathbf{x}_1\sqrt{3}-5\mathbf{x}_2))$, $g_3(\mathbf{x}) = 0$, and $\Phi(.)\ T_2(.)$ denote the univariate cumulative standard normal and student t with two degrees of freedom (t_2) distributions. The center panel of Fig. 2 displays a validation sample with 1000 examples generated from this data condition.

Experiments 3 and 4. The third and fourth experiments use the data conditions described in Examples 1 and 2 of Xu and Wang (2013), namely training samples of 400 observations with the Y class labels generated uniformly from $\mathcal{Y} = \{1, 2, \ldots, k\}$, and two-dimensional predictors generated conditionally on Y, from a t_2 distribution with mean $\boldsymbol{\mu}(y) = [cos(2y\pi/k), sin(2y\pi/k)]^T$ and covariance $\Sigma = diag(1, 2)$. In experiment 3, $k = 5$ while in experiment 4, $k = 10$. The right panel of Fig. 2 displays a validation sample with 1000 examples generated from experiment 3. The pattern for the condition with ten classes (experiment 4) is similar.

Table 3
Error rates for Experiment 1.

	MLDA	MLR	MGAM	PVM	WZW	XW	TREE	RF	WZL
l2err									
mean	0.29	0.36	0.53	0.57	1.62	6.26	8.23	5.18	0.90
(stderr)	(0.03)	(0.03)	(0.04)	(0.04)	(0.09)	(0.13)	(0.15)	(0.11)	–
median	0.21	0.30	0.40	0.52	1.59	6.26	8.26	5.13	–
EGKL									
mean	0.59	0.77	1.12	1.53	3.47	∞	∞	∞	2.56
(stderr)	(0.05)	(0.07)	(0.09)	(0.09)	(0.15)	–	–	–	–
median	0.46	0.64	0.91	1.44	3.38	∞	∞	∞	–

Table 4
Error rates for Experiment 2.

	MLDA	MLR	MGAM	PVM	WZW	XW	TREE	RF	WZL
l2err									
mean	6.90	6.62	5.21	2.50	5.01	9.44	10.13	5.26	4.47
(stderr)	(0.05)	(0.03)	(0.06)	(0.09)	(0.31)	(0.22)	(0.27)	(0.08)	–
median	6.81	6.58	5.19	2.38	4.71	9.03	9.75	5.35	–
EGKL									
mean	12.59	12.33	10.69	5.97	8.49	∞	∞	∞	11.79
(stderr)	(0.06)	(0.06)	(0.07)	(0.11)	(0.40)	–	–	–	–
median	12.51	12.18	10.59	5.81	7.70	∞	∞	∞	–

In all four experiments, we trained the eight estimation methods on 50 different, independently generated, training samples, tuned the λ hyper-parameter on independently generated tuning data, and evaluated them by computing the l_2 norm error, $l_2\ err = \frac{1}{\#\mathcal{E}} \sum_{i\in\mathcal{E}} \sum_{c\in\mathcal{Y}} (\hat{\mathbf{p}}_c(\mathbf{x}_i) - \mathbf{p}_c(\mathbf{x}_i))^2$, and Empirical Generalized Kullback–Leibler (EGKL) measure, $EGKL = \frac{1}{\#\mathcal{E}} \sum_{i\in\mathcal{E}} \sum_{c\in\mathcal{Y}} \mathbf{p}_c(\mathbf{x}_i) \ln \frac{\mathbf{p}_c(\mathbf{x}_i)}{\hat{\mathbf{p}}_c(\mathbf{x}_i)}$, in an independently generated validation data set ($\mathcal{E}$) with 1000 examples. Figs. 3 through 6 show sinaplots (Sidiropoulos et al., 2018) of the l_2-norm and EGKL errors for all simulation experiments, while Tables 3 through 6 present the corresponding means, mean standard errors, and medians. Tables 3 and 4 also show, under the WZL acronym, the reported average of these measures (see Wu et al. (2010)) for the original Wu, Zhang and Liu proposal. We note that, unlike in our experiments, Wu et al. (2010) did not use an universal kernel but a linear one instead, taking advantage of the known form of the optimal classification boundaries for these conditions. However, in real data problems the true form of these boundaries is always unknown.

For some combinations of experiments, replications, and classes, the XL, TREE or RF methods estimate class probabilities exactly by 0, which leads to an infinite value of the EGKL measure. This problem does not occur for the l_2-norm error measure, nor for the remaining methods including our PVM proposal, which always enforce strictly positive probability estimates. Nevertheless, we tend to prefer the EGKL loss as an global measure of estimation performance because, unlike the l_2 norm error, it does not treat equal absolute errors as equally important, regardless of being associated with probabilities close to 0.5, or to the 0 and 1 boundaries of their domain. However, we note

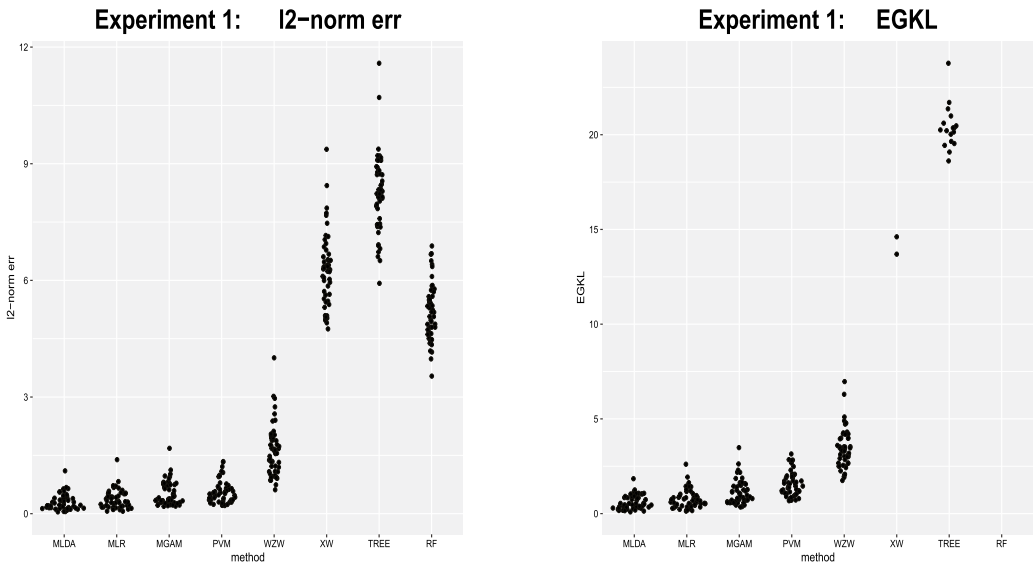


Fig. 3. l2-norm and EGKL error rates of Experiment 1.

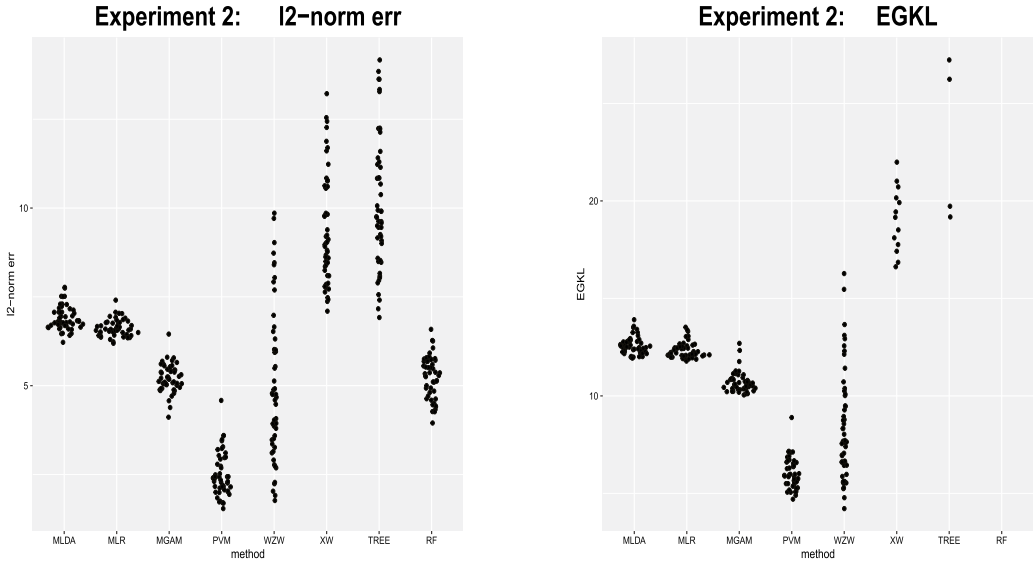


Fig. 4. l2-norm and EGKL error rates of Experiment 2.

Table 5
Error rates for Experiment 3.

	MLDA	MLR	MGAM	PVM	WZW	XW	TREE	RF
l2err								
mean	5.60	5.01	3.59	4.17	2.47	2.82	4.73	13.19
(stderr)	(0.10)	(0.10)	(0.07)	(0.08)	(0.05)	(0.04)	(0.17)	(0.11)
median	5.56	4.91	3.61	4.13	2.39	2.81	4.57	13.23
EGKL								
mean	15.30	15.80	12.17	11.28	6.00	∞	∞	∞
(stderr)	(0.15)	(0.20)	(0.28)	(0.18)	(0.12)	–	–	–
median	15.08	15.66	11.53	11.26	5.81	6.36	∞	∞

Table 6
Error rates for Experiment 4.

	MLDA	MLR	MGAM	PVM	WZW	XW	TREE	RF
l2err								
mean	3.34	3.09	–	2.63	1.81	1.81	3.93	14.89
(stderr)	(0.05)	(0.05)	–	(0.06)	(0.03)	(0.03)	(0.13)	(0.10)
median	3.34	3.07	–	2.53	1.81	1.78	3.82	14.81
EGKL								
mean	17.33	18.47	–	11.83	8.64	8.03	∞	∞
(stderr)	(0.20)	(0.29)	–	(0.19)	(0.16)	(0.15)	–	–
median	17.36	18.14	–	11.42	8.54	7.76	∞	∞

that, in these experiments, when results for the EGKL loss are available the resulting rankings of the eight estimation methods tends to agree with the rankings given by the l_2 error. In Experiment 4 (ten-classes with heavy-tailed distributions) we were not able to fit MGAM models in many replications because of numerical difficulties that were not resolved by the usual remedies of trying different starting conditions and data scalings. Therefore, we not present results for the MGAM method under this condition.

Overall these results confirmed previous findings in the literature, and gave new evidence on the competitivity of the SVM approach to probability estimation. In particular, the MLR and MLDA methods gave the best results when the assumptions of MLR were met (Experiment 1), but when they were grossly violated one of the SVM based methods

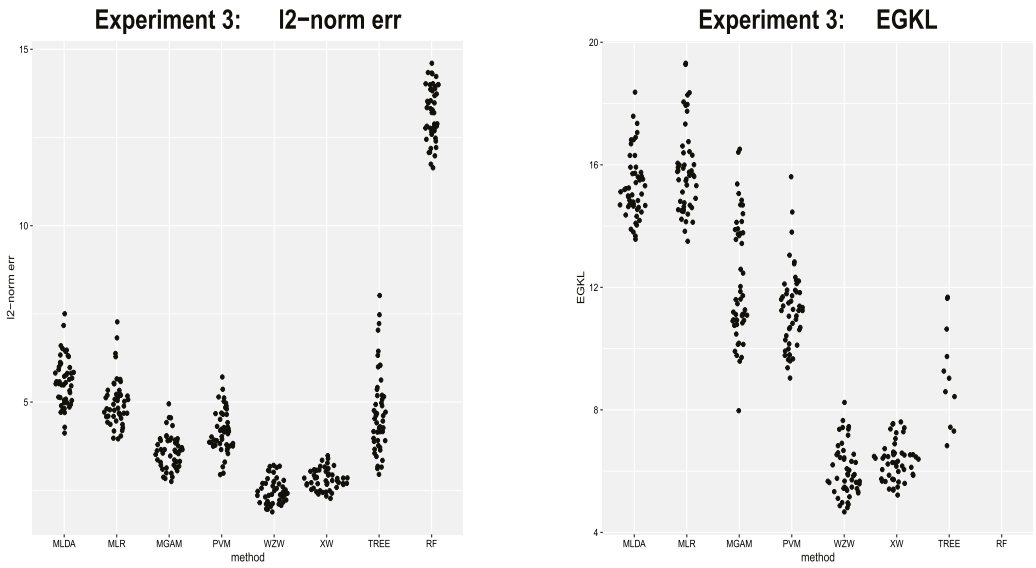


Fig. 5. l2-norm and EGKL error rates of Experiment 3.

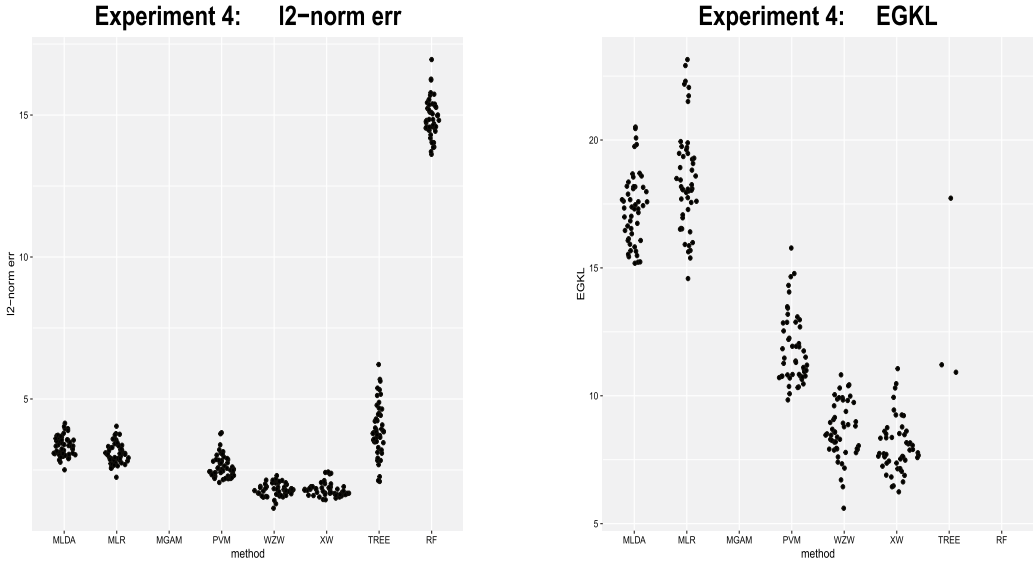


Fig. 6. l2-norm and EGKL error rates of Experiment 4.

performed the best. The MGAM performed worse than MLR and MLDA under MLR assumptions, and better than these two methods otherwise. However, in the conditions where MGAM beat the two classical statistical methods, it was always inferior to at least one of the SVM based methods. On the other hand, the performance of the Tree and Random Forest methods was disappointing, suggesting that in spite of their merits for pure supervised classification problems, they are not as reliable at providing estimates of class probabilities.

Amongst the three SVM methods we did not find a clear overall winner. In these experiments, the pairwise *one-against-the-rest* methods performed the best for the conditions characterized by heavy-tailed distributions (Experiments 3 and 4), while the *all-in-one* PVM gave the best results for the non-linearly transformed data (Experiment 2). Furthermore, both the PVM and MGAM methods seemed relatively stable across different data conditions, coming often as second best when they were not the ideal methods and, in particular, performing better than the SVM pairwise methods when the assumptions of MLR are satisfied (Experiment 1), and better than MLR and MLDA when the data has severe outliers (Experiments 3 and 4). Amongst the pairwise *one-against-the-rest* SVM methods the WZW performed better than the XW, confirming previous findings reported in Wang et al. (2019).

Remarkably, in the experiments for which results for the original *all-in-one* (Wu et al., 2010) proposal were available, those results were improved by our PVM proposal, even though we used only 297 (Experiment 1) and 364 (Experiment 2) grid points, while Wu et al. used a total of 1326 grid points in each of these experiments. We believe that this surprising result might be explained by the fact the recovery of probabilities by optimizing (9) uses more information, than the matching of observed with expected frequencies employed in the original *all-in-one* SVM method. Finally the comparison of the results of experiments 3 and 4 suggests that the number of different classes might not a strong influence on the relative standing of alternative estimation methods. Naturally, additional studies are necessary to verify if these results hold for other data conditions.

4.2. Benchmark datasets

We will now compare the different estimators considered regarding their performance in four benchmark data sets, previously analyzed

Table 7
Main characteristics of benchmark data sets.

	k	Total Sz.	Train. Smp. Sz.	Tun. Smp. Sz.	Valid. Smp. Sz.
Wine	3	178	105	8×15	58
Ecoli	3	336	197–199	$8 \times (27 - 29)$	110
Yeast5	5	1484	867–872	$8 \times (121 - 126)$	491
Abalone10	10	3739	1253	1245	1241

in the literature. Since the true class probabilities are unknown in real-world data, the comparisons will be based on validation data log-likelihoods and error rates.

The data sets considered here will be denoted by *Wine*, *Ecoli3*, *Yeast5* and *Abalone10*. The *Wine* data is available online at the UCI Machine Learning repository, was collected by Forina et al. (1988), and was previously analyzed by Wu et al. (2010). The Ecoli and Yeast data were collected by Horton and Nakai (1996), and analyzed by Wang et al. (2019), while the *Abalone* data was collected by Nash et al. (1994), and analyzed by Xu and Wang (2013).

The *Wine* data is divided into three classes, while the original *Ecoli* and *Yeast* data were initially divided into, respectively eight and ten classes. However, since these classes are highly imbalanced and some of them have too few examples, here we will consider aggregated versions of these data sets, leading to versions with, respectively, three (*Ecoli3*) and five (*Yeast5*) classes. The Abalone data set was collected by Nash et al. (1994) and analyzed previously by Xu and Wang (2013). In the original data, 4176 abalones were divided into 19 age groups, but, given the imbalance of those groups, here, we will consider one subsample (*Abalone10*) that includes ten age classes.

In the *Wine*, *Ecoli3* and *Yeast5* data sets, we randomly selected approximately one third of the observations in each class for a validation set, and divided the remaining data randomly, again stratifying by class, into eight different folds. Then, the value of the λ hyper-parameter was chosen by the cross-validation procedure described in Section 3.2. In the larger *Abalone10* data set, we randomly divided the original data (stratifying by class) into three roughly equally sized data sets, and used the first one for training, the second one for tuning the value of λ, and the third one for validation.

Additional characteristics of these data sets are presented below and summarized in Table 7.

Wine

The *Wine* data set consists of 178 examples divided into three classes of wine types, with 59 examples in the first class, 71 in the second class, and 48 in the third class. The characteristics of each wine are described by 13 continuous attributes. Since these attributes are measured on widely different scales, before any analysis were performed they were standardized in order to have a null mean and an unit standard deviation.

Ecoli3

The original Ecoli data set consists of 336 examples of Ecoli proteins divided into 8 classes according to its location site: inner membrane lipoproteins (*imL*), outer membrane lipoproteins (*omL*), inner membrane proteins with cleavable signal sequence (*imS*), other outer membrane proteins (*om*), periplasmic proteins (*pp*), inner brane proteins with an uncleavable signal sequence(*imU*), inner membrane proteins without a signal sequence (*im*), and cytoplasmic proteins (*cp*). Seven attributes based on amino acid sequences were used to describe the examples. Since the number of examples in each class varies widely from only 2 in the *imL* and *imS* classes to 143 in the *cp* class, we aggregated them into three different classes (*im* with 77 examples, *cp* with 143 examples and other locations with 166 examples) according to the classification tree developed by Horton and Nakai (1996), and shown on the left panel of Fig. 7.

Yeast5

The original Yeast data set consists of 1484 examples of Yeast proteins divided into 10 classes: cytoplasmic, including cytoskeletal (*CYT*); nuclear (*NUC*); vacuolar (*VAC*); mitochondrial (*MIT*); isomal (*POX*); extracellular, including those localized to the cell wall (*EXC*); proteins localized to the lumen of the endoplasmic reticulum (*ERL*); membrane proteins with a cleaved signal (*ME1*); membrane proteins with an uncleared signal (*ME2*); and membrane proteins with no N-terminal signM (*ME3*). Eight attributes based on amino acid sequences were used to describe the examples. Since the number of examples in each class is highly unbalanced, we aggregated them into five different classes with respectively, 114, 51, 244, 646 and 429 examples, according to the classification tree developed by Horton and Nakai (1996), and shown on the right panel of Fig. 7.

Table 8
Number of examples per class (Abalone10).

Age class (years)	5	6	7	8	9	10	11	12	13	14
Class size	115	259	391	568	689	634	487	267	203	126

Abalone10

The original Abalone data set consists of 4176 abalones divided into 19 age classes, and measured on 8 attributes of physical characteristics. However, since the younger and older classes have two few examples, here we consider the subsample of Abalones older that 4 and younger than 15 years of age, which leads to a problem with 3739 examples and 10 classes, with the number of examples per class ranging from 115 (5 years old abalones) to 689 (9 years old abalones), as shown in Table 8.

Table 9 presents the validation sample log-likelihood and error rates in the four benchmark data sets for all estimation methods compared in this paper. From these figures we can see that none of the methods is uniformly better than the alternatives, and the methods rankings in terms of validation log-likelihood do not always agree with their rankings in terms of validation error rates. In particular, according to the validation log-likelihoods, the best performing methods were the MLR in the Wine data set, MLDA in the Ecoli data, PVM in the Yeast5 data, and XW in the Abalone10 data. On the other hand, in terms of error rates, the best results were obtained the MLR and WZW methods (Wine data), MLDA (Ecoli data), MGAM (Yeast5 data) and PVM (Abalone10 data). Unlike what happens in the other data sets, the partition for the Abalone10 is intrinsically ordinal, which may help to explain the relative good performance of the XW method for this data. Furthermore, we note that as the number of classes increases so does the difficulty of the problems, which partly explains the relativity high error rates for all methods in the ten-class Abalone10 data. Finally, it is important to remark that for most methods the differences in performance were relatively small so that, apart for confirming the poor performance of the TREE and RF methods for probability estimation and the typical superiority of WZW over XW in true nominal problems, no general raking of different methods could be firmly established. Indeed, these results suggest that for probability estimation in true nominal classification problems, any of the MLDA, MLR, MGAM, PVM and WZW methods can be competitive, depending of the particular characteristics of the data to analyzed.

5. Conclusions and further research

Kernel based Support Vector Machines (SVMs) are known to be the amongst the most accurate machine learning algorithms for supervised classification problems. However, classical classification SVMs focus on pure predictions and do not pay enough attention to the related, and critical, problem of providing reliable confidence measures for these predictions. Nevertheless, taking advantage of information provided by sequences of weighted SVMs, reliable estimates of class memberships can be found. Unfortunately, up to now this approach was not computationally feasible for a moderate (or large) number of different classes, unless suboptimal pairwise *one-against-the-rest* decomposition strategies were employed.

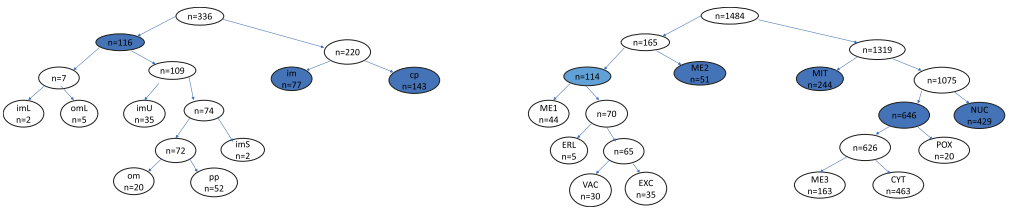


Fig. 7. Original Ecoli and Yeast class trees.

Table 9
Validation results for benchmark data sets.

	MLDA	MLR	MGAM	PVM	WZW	XW	TREE	RF
Log-Likelihoods								
Wine	−4.6	0.0	−14.9	−2.3	−2.5	−14.9	−∞	−6.2
Ecoli	−34.0	−35.2	−44.6	−41.8	−42.7	−58.2	−∞	−41.3
Yeast5	−529.8	−520.9	−501.2	−488.0	−497.5	−∞	−∞	−493.7
Abalone10	−2187.3	−2130.3	−2100.4	−2212.3	−2126.8	−2093.9	−2256.4	−∞
Error rates								
Wine	1	0	2	1	0	5	2	0
	(1.7%)	(0.0%)	(3.4%)	(1.7%)	(0.0%)	(8.6%)	(3.4%)	(0.0%)
Ecoli	10	12	14	15	17	20	14	13
	(9.1%)	(10.9%)	(12.7%)	(13.6%)	(15.5%)	(18.2%)	(12.7%)	(11.8%)
Yeast5	201	201	188	199	186	226	191	189
	(40.9%)	(40.9%)	(38.3%)	(40.5%)	(37.9%)	(46.0%)	(38.9%)	(39.3%)
Abalone10	865	842	844	831	849	851	890	886
	(69.7%)	(67.8%)	(68.0%)	(67.0%)	(68.4%)	(68.6%)	(71.7%)	(71.4%)

In this paper we have filled this gap, and developed a computationally efficient *all-in-one* estimation method based on sequences of weighted SVMs that consider all classes simultaneously. Furthermore, we have provided statistical evidence that SVM based probability estimators are amongst the most reliable distribution free estimators for class probabilities. Our results also suggest that no particular method of extending two-class to general k-class SVM methodologies dominates the alternatives, and different data conditions may favor different approaches.

The research presented in this paper can be extended in a number of different directions.

Firstly, while here we have focused on SVMs with universal kernels, many successful standard SVM applications rely on non-universal kernels. In particular, in applications with a large number of attributes, linear SVMs have shown to be very effective. It has been argued that, in such cases, the attribute expansion implied by kernel transformations may not be particularly useful because the original attribute space is already rich enough (Chauhan et al., 2019). Furthermore, there are faster specialized training algorithms for linear SVMs with large sparse data (*e.g.* (Keerthi et al., 2008)) that are common in several application domains, such as document classification. We intend to adapt the approach developed here to weighted linear SVMs using variants these algorithms, and study their properties in the relevant domains.

Secondly, we also intend to specialize this approach to the problem of multiclass probability estimation in ordinal classification problems.

Lastly, while here we have focused on weighted Fisher consistent SVMs, the literature on standard classification SVMs suggests that sometimes SVMs without consistent properties can match, or even outperform, competing SVMs with apparently sounder theoretical properties. In particular, extensive simulation studies by Dogan et al. (2011) show that SVMs based on the non Fisher consistent loss proposed by Weston and Watkins (1999) (WW) have similar, or even better, prediction accuracy than the LLW SVM, for most of the data conditions considered. We conducted some preliminary simulations with weighted WW SVMs for probability estimation, and found some equally promising results. We intend to further study if this intriguing finding holds out in more general conditions, and take a deeper look at the relevance, and limitations, of weighted SVM theoretical properties in several application domains

Acknowledgments

This work is financed by National Funds through the Portuguese funding agency, FCT - Fundação para a Ciência e a Tecnologia, Portugal, within project UID/GES/00731/2020.

Appendix A. Dual of probability estimation problem

Introducing slack variables, problem (9)–(10) can be reformulated as

$$min_{\mathbf{d}^+,\mathbf{d}^i \in \Re^{G\times k},\ \mathbf{p}(\mathbf{x})\in\Re^k} \sum_{g=1}^{G} \sum_{c\in\mathcal{Y}\setminus\{\hat{y}_g\}} \eta\ \mathbf{d}_{gc}^+ - \mathbf{d}_{gc}^- \tag{A.1}$$

subject to

$$\boldsymbol{\pi}_c^g\, \mathbf{p}_c(\mathbf{x}) - \boldsymbol{\pi}_{\hat{y}_g}^g\, \mathbf{p}_{\hat{y}_g}(\mathbf{x}) - \mathbf{d}_{gc}^+ + \mathbf{d}_{gc}^- = 0 \qquad g=1,\ldots,G;\ c\in\mathcal{Y}\setminus\{\hat{y}_g\} \tag{A.2}$$

$$\sum_{c\in\mathcal{Y}} \mathbf{p}_c(\mathbf{x}) = 1 \tag{A.3}$$

$$\mathbf{p}_c(\mathbf{x}) \geq\ \epsilon \qquad \forall\ c\in\mathcal{Y} \tag{A.4}$$

$$\mathbf{d}_{gc}^+, \mathbf{d}_{gc}^- \geq 0 \qquad g=1,\ldots,G;\ c\in\mathcal{Y}\setminus\{\hat{y}_g\} \tag{A.5}$$

While typically (A.1)–(A.5) is a linear programming model with a large ($= G(k-1)+1$) number of explicit of constraints (i.e. excluding simple variable bounds), after defining the auxiliary functions,

$$h_g(c) = \begin{cases} c, & c < \hat{y}_g \\ c-1, & c > \hat{y}_g \end{cases}$$

its dual given can be formulated as

$$max_{\boldsymbol{\mu}\in\Re^{G\times(k-1)},\,\xi\in\Re^k,\,\nu,\in\Re} \left(\nu\ +\ \epsilon \sum_{c\in\mathcal{Y}} \xi_c\right) \tag{A.6}$$

subject to

$$\sum_{g=1:\hat{y}_g\neq c}^{G} \boldsymbol{\pi}_c^g \boldsymbol{\mu}_{gh_g(c)} - \sum_{g=1:\hat{y}_g=c}^{G} \sum_{c'\in\mathcal{Y}\setminus\{\hat{y}_g\}} \boldsymbol{\pi}_{\hat{y}_g}^g\ \boldsymbol{\mu}_{gh_g(c')} + \nu + \xi_c \leq 0 \quad \forall c\in\mathcal{Y} \tag{A.7}$$

$$1 \leq \boldsymbol{\mu}_{gh_g(c)} \leq \eta \qquad g=1,\ldots,G; c\in\mathcal{Y}\setminus\{\hat{y}_g\} \tag{A.8}$$

$$\xi_c \geq 0 \qquad \forall c \in \mathcal{Y} \quad \text{(A.9)}$$

which is a problem with a much smaller ($=k$) number of explicit constraints. After optimizing (A.6)–(A.9), the values of $\mathbf{p}_c(\mathbf{x})$ can be recovered from the dual variables corresponding to the (A.7) constraints.

Appendix B. Dual of base weighted SVM problem

Here, we will show that the dual of optimization problem (3)–(4) with loss (6) is given by problem (15)–(17). First notice that, using matrix notation and introducing slack variables, the primal problem can be expressed as

$$min_{\boldsymbol{\theta},\boldsymbol{\epsilon}\in\mathbb{R}^{n\times k}} \sum_{c\in\mathcal{Y}} \left[\lambda\, \boldsymbol{\theta}_{.c}^T\, \mathbf{K}\, \boldsymbol{\theta}_{.c} \;+\; \mathbf{C}_{.c}^T\, \boldsymbol{\epsilon}_{.c}\right] \quad \text{(B.1)}$$

$$subject\ to \qquad \mathbf{K}\,\boldsymbol{\theta} \;\leq\; -(k-1)^{-1}\,\mathbf{1}_{n\times k} + \boldsymbol{\epsilon} \quad \text{(B.2)}$$

$$\sum_{c\in\mathcal{Y}} \mathbf{K}\,\boldsymbol{\theta}_{.c} = \mathbf{0}_n \quad \text{(B.3)}$$

$$\boldsymbol{\epsilon} \;\geq\; \mathbf{0}_{n\times k} \quad \text{(B.4)}$$

where, with a slight abuse of notation, the matrix $\boldsymbol{\theta}$ is defined as the matrix with columns $\boldsymbol{\theta}_{.c}$ equal to the $\boldsymbol{\theta}^c$ vectors of the $\mathbf{f}_c$ expansion given by the representer theorem, $\mathbf{K} \in \mathbb{R}^{n\times n}$ is the Gram matrix with generic element $\mathbf{K}_{ii'} = K(\mathbf{x}_i, \mathbf{x}_{i'})$, and the cost matrix $\mathbf{C} \in \mathbb{R}^{n\times k}$ has generic elements $\mathbf{C}_{ic} = n^{-1}\,\pi_{y_i}$ if $c \neq y_i$, and $\mathbf{C}_{iy_i} = 0$. Furthermore, we denote the transpose of a generic matrix $\mathbf{M}$ (or vector $\mathbf{v}$) by $\mathbf{M}^T$ ($\mathbf{v}^T$), its c-column by $\mathbf{M}_{.c}$ and $\mathbf{1}_{n\times k}$, $\mathbf{0}_{n\times k}$, $\mathbf{0}_n$ are $n \times k$ matrices and an n-dimensional vector with all elements equal, respectively, to 1 and 0.

Introducing Lagrange multipliers for constraints (B.2), (B.3) and (B.4), the Wolfe dual problem can be written as

$$max_{\boldsymbol{\alpha},\boldsymbol{\gamma}\in\mathbb{R}^{n\times k},\boldsymbol{\beta}\in\mathbb{R}^n} \inf_{\boldsymbol{\theta},\boldsymbol{\epsilon}\in\mathbb{R}^{n\times k}} L_D =$$

$$= \sum_{c\in\mathcal{Y}} \left[\lambda\, \boldsymbol{\theta}_{.c}^T\, \mathbf{K}\, \boldsymbol{\theta}_{.c} + \mathbf{C}_{.c}^T\, \boldsymbol{\epsilon}_{.c} + \boldsymbol{\alpha}_{.c}^T \left(\mathbf{K}\,\boldsymbol{\theta}_{.c} + \frac{1}{k-1}\,\mathbf{1}_n - \boldsymbol{\epsilon}_{.c}\right) - \boldsymbol{\gamma}_{.c}^T\, \boldsymbol{\epsilon}_{.c}\right] +$$

$$+ \boldsymbol{\beta}^T \left[\sum_{c\in\mathcal{Y}} \mathbf{K}\,\boldsymbol{\theta}_{.c}\right] \quad \text{(B.5)}$$

$$subject\ to \qquad \boldsymbol{\alpha},\ \boldsymbol{\gamma} \;\geq\; \mathbf{0}_{n\times k} \quad \text{(B.6)}$$

Minimization in order to the $\boldsymbol{\theta}$ and $\boldsymbol{\epsilon}$ primal variables leads to

$$\frac{\partial L_D}{\partial \boldsymbol{\theta}_{.c}} = \mathbf{0_n} \;\Leftrightarrow\; 2\lambda\, \boldsymbol{\theta}_{.c}^T\, \mathbf{K} + (\boldsymbol{\alpha}_{.c}^T + \boldsymbol{\beta}^T)\, \mathbf{K} = \mathbf{0_n} \;\Rightarrow$$

$$\Rightarrow\; \boldsymbol{\theta}_{.c} = -(2\lambda)^{-1}\,(\boldsymbol{\alpha}_{.c} + \boldsymbol{\beta}) \quad \text{(B.7)}$$

$$\frac{\partial L_D}{\partial \boldsymbol{\epsilon}_{.c}} = \mathbf{0_n} \;\Leftrightarrow\; \mathbf{C}_{.c} - (\boldsymbol{\alpha}_{.c} + \boldsymbol{\gamma}) = \mathbf{0}_n \quad \text{(B.8)}$$

where the second equality in (B.7) follows from the fact that $\mathbf{K}$, being a positive definite matrix, is always non-singular.

From (B.6) and (B.8), it follows that

$$\mathbf{0_{n\times k}} \;\leq\; \boldsymbol{\alpha} \;\leq\; \mathbf{C} \quad \text{(B.9)}$$

while replacing (B.7) into (B.3) leads to

$$\sum_{c\in\mathcal{Y}} \mathbf{K}\,(\boldsymbol{\alpha}_{.c} + \boldsymbol{\beta}) = \mathbf{0_n} \;\Rightarrow\; \boldsymbol{\beta} \;=\; -k^{-1} \sum_{c\in\mathcal{Y}} \boldsymbol{\alpha}_{.c} \quad \text{(B.10)}$$

Replacing (B.7) and (B.10) into (B.5), and simplifying making use of (B.8), leads to

$$L_D = \sum_{c\in\mathcal{Y}} [\,(k-1)^{-1}\,\boldsymbol{\alpha}_{.c}^T\,\mathbf{1_n} \;-\; (2\lambda)^{-1}\;\boldsymbol{\alpha}_{.c}^T\,\mathbf{K}\,(\boldsymbol{\alpha}_{.c} \;-\; k^{-1} \sum_{c'\in\mathcal{Y}} \boldsymbol{\alpha}_{.c'})\,] \quad \text{(B.11)}$$

so that the dual in matrix form is given by

$$max_{\boldsymbol{\alpha}\in\mathbb{R}^{n\times k}} \sum_{c\in\mathcal{Y}} [\,(k-1)^{-1}\,\boldsymbol{\alpha}_{.c}^T\,\mathbf{1_n} \;-\; (2\lambda)^{-1}\;\boldsymbol{\alpha}_{.c}^T\,\mathbf{K}\,(\boldsymbol{\alpha}_{.c} \;-\; k^{-1} \sum_{c'\in\mathcal{Y}} \boldsymbol{\alpha}_{.c'})\,] \quad \text{(B.12)}$$

$$subject\ to \qquad \mathbf{0_n} \times k \;\leq\; \boldsymbol{\alpha} \;\leq\; \mathbf{C} \quad \text{(B.13)}$$

which is equivalent to (15)–(17).

Data availability

Data will be made available on request.

References

Bazaraa, M.S., Jarvis, J.J., Sherali, H.D., 2011. Linear Programming and Network Flows. John Wiley and Sons.

Bertero, M., 2006. Regularization methods for linear inverse problems. In: Inverse Problems: Lectures Given At the 1st 1986 Session of the Centro Internazionale Matematico Estivo (CIME) Held At Montecatini Terme, Italy, May 28–June 5 1986. Springer, pp. 55–112.

Breiman, L., 2001. Random forests. Mach. Learn. 45 (1), 5–32.

Breiman, L., Friedman, R., Olshen, J., Stone, C., 1984. Classification and Regression Trees. Wadsworth Publishing Company.

Caputo, B., Sim, F., Furesjo, K., Smola, A., 2002. Appearance-based object recognition using svms: which kernel should i use? In: Proc of NIPS Workshop on Statistical Methods for Computational Experiments in Visual Processing and Computer Vision. Whistler.

Casado, L., Salmeon, E., BG-Toth, J., Garcia, I., 2016. On regular refinement of unit simplex by just visiting grid points. In: XIII Global Optimization Workshop GOW'16 4-8 2016, vol. 16, pp. 33–36.

Chauhan, V.K., Dahiya, K., Sharma, A., 2019. Problem formulations and solvers in linear svm: a review. Artif. Intell. Rev. 52 (2), 803–855.

Crammer, C., Singer, Y., 2001. On the algorithmic implementation of multiclass kernel-based vector machines. J. Mach. Learn. Res. 2, 265–292.

Cristianini, N., Shawe-Taylor, J., 2000. An Introduction To Support Vector Machines. Cambridge University Press.

Dogan, U., Glasmachers, T., Igel, C., 2011. Fast Training of Multi-Class Support Vector Machines. Technical report, Department of Computer Science, University of Copenhagen..

Dogan, U., Glasmachers, T., Igel, C., 2016. A unified view on multi-class support vector classification. J. Mach. Learn. Res. 17 (45), 1–32.

Duarte Silva, A.P., 2017. Optimization approaches to supervised classification. European J. Oper. Res. 261 (2), 772–788.

Evgeniou, T., Pontil, M., Poggio, M., 2000. Regularization networks and support vector machines. Adv. Comput. Math. 13, 1–50.

Forina, M., Lanteri, S., Armanino, C., et al., 1988. Parvus-an Extendible Package for Data Exploration, Classification and Correlation. Technical report, Institute of Pharmaceutical and Food Analysis and Technologies, via brigata salerno, 16147 Genoa, Italy.

Girosi, F., 1986. An equivalence between sparse approximation and support vector machines. Neural Comput. 10 (6), 1455–1480.

Guermeur, Y., 2017. Lp-norm sauer–shelah lemma for margin multi-category classifiers. J. Comput. System Sci. 89, 450–473.

Guermeur, Y., 2020. Rademacher complexity of margin multi-category classifiers. Neural Comput. Appl. 32 (24), 17995–18008.

Horton, P., Nakai, K., 1996. A probabilistic classification system for predicting the cellular localization sites of proteins. In: Proceedings of the International Conference on Intelligent Systems for Molecular Biology, vol. 4, pp. 109–115.

Igel, C., Heidrich-Meisner, T., Glasmachers, V., 2008. Shark. J. Mach. Learn. Res. 9 (6), 993–996.

Keerthi, S., Sundararajan, K., Chang, C., Hsieh, S., Lin, C., 2008. A sequential dual method for large scale multi-class linear svms. In: Li, Y., Liu, B., Sarawagi, S. (Eds.), Proceedings of the 14th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining. ACM, pp. 408–416.

Kimeldorf, G., Wahba, G., 1971. Somes results on tchebycheffian spline functions. J. Math. Anal. Appl. 33, 82–95.

Lee, Y., Lin, Y., Wahba, G., 2004. Multicategory support vector machines: Theory and application to the classification of microarray data and satellite radiance data. J. Amer. Statist. Assoc. 99 (465), 67–81.

Lei, Y., Dogan, D.X., Zhou, U., Kloft, M., 2019. Data-dependent generalization bounds for multi-class classification. IEEE Trans. Inform. Theory 65 (5), 2995–3021.

Liaw, A., Wiener, M., 2002. Classification and regression by randomforest. R News 2 (3), 18–22.

Lin, Y., 2002. Support vector machines and the bayes rule in classification. Data Min. Knowl. Discov. 6 (3), 988–994.

Lin, Y., Lee, Y., Wahba, G., 2002. Support vector machines for classification in nonstandard situations. Mach. Learn. 46 (1), 191–202.

McLachlan, G. J., 2005. Discriminant Analysis and Statistical Pattern Recognition. John Wiley & Sons.

Nash, W.J., Sellers, S.R., Talbot, A.J., Cawthorn, T.L., Ford, W.B., 1994. The Population Biology of Abalone (Haliotis Species) in Tasmania. I. Blacklip Abalone (H. Rubra) from the North Coast and Islands of Bass Strait. Technical report, Sea Fisheries Division.

Platt, J., 1999a. Fast training of support vector machines using sequential minimal optimization. In: B. Schölkopf, J. Burges, C, Somola, A. (Eds.), Advances in Kernel Methods—Support Vector Learning. MIT Press, pp. 185–208.

Platt, J., 1999b. Probabilistic outputs for support vector machines and comparisons to regularized likelihood methods. In: Somola, A., Bartlett, P., Schölkopf, B., Schuurmans, D. (Eds.), In: Advances in large margin classifiers, vol. 10, (3), MIT Press, pp. 61—74.

Poggio, T., Mukherjee, R., Rifkin, A., Raklin, S., Verri, A., 2002. In: Winkler, B. In J., Niranjan, M. (Eds.), Uncertainty in Geometric Computations. Springer US.

R Core Team, 2025. R: A Language and Environment for Statistical Computing. R Foundation for Statistical Computing, Vienna, Austria, URL https://www.R-project.org/.

Sidiropoulos, N., Sohi, T., Pedersen, B., Porse, O., Winther, N., Rapin, S., Bagger, F., 2018. Sinaplot: an enhanced chart for simple and truthful representation of single observations over multiple classes. J. Comput. Graph. Statist. 27 (3), 673–676.

Steinwart, I., 2002. Support vector machines are universally consistent. J. Complexity 18 (3), 768–791.

Therneau, T., Atkinson, B., 2018. Rpart: Recursive partitioning and regression trees. R package version 4.1-13.

Tikhonov, A.N., 1963. On the solution of ill-posed problems and the method of regularization. In: Doklady akademii nauk, vol. 51, (3), Russian Academy of Sciences, pp. 501–504.

Vapnik, V., 1998. Statistical Learning Theory. Wiley, New York.

Venables, W.N., Ripley, B.D., 2002. Modern Applied Statistics with S, Fourth ed. Springer, New York.

Wahba, G., 1998. Support vector machines, reproducing kernel hilbert spaces, and randomized gacv. In: Schölkopf, B., Burges, C., Somola, A. (Eds.), Advances in Kernel Methods: Support Vector Learning. MIT Press, pp. 69–87.

Wang, J., Shen, X., Liu, Y., 2008. Probability estimation for large-margin classifiers. Biometrika 95 (1), 149–167.

Wang, X., Zhang, H., Wu, Y., 2019. Multiclass probability estimation with support vector machines. J. Comput. Graph. Statist. 28 (3), 586–595.

Weston, J., Watkins, C., 1999. Support vector machines for multi-class pattern recognition. In: Verleysen, M. (Ed.), Proceedings of the 7th European Symposium on Artificial Neural Networks. pp. 219–224.

Wu, Y., Zhang, H., Liu, Y., 2010. Robust model-free multiclass probavility estimation. J. Amer. Statist. Assoc. 105 (489), 424–436.

Xu, T., Wang, J., 2013. An efficient model-free estimation of multiclass conditional probability. J. Statist. Plann. Inference 143, 2079–2088.

Yee, T. W., 2010. The vgam package for categorical data analysis. J. Stat. Softw. 32, 1–34.

Yee, T.W., Wild, C., 1996. Vector generalized additive models. J. R. Stat. Soc. Ser. B Stat. Methodol. 58 (3), 481–493.