Majoration-Minimization for Sparse SVMs

Alessandro Benfenati, Emilie Chouzenoux, Giorgia Franchini, Salla Latva-Äijö, Dominik Narnhofer, Jean-Christophe Pesquet, Sebastian J. Scott, Mahsa Yousefi

Majorization-Minimization for sparse SVMs

Alessandro Benfenati, Emilie Chouzenoux, Giorgia Franchini, Salla Latva–Äijö, Dominik Narnhofer, Jean–Christophe Pesquet, Sebastian J. Scott, Mahsa Yousefi

Abstract Several decades ago, Support Vector Machines (SVMs) were introduced for performing binary classification tasks, under a supervised framework. Nowadays, they often outperform other supervised methods and remain one of the most popular approaches in the machine learning arena. In this work, we investigate the training of SVMs through a smooth sparse-promoting-regularized squared hinge loss minimization. This choice paves the way to the application of quick training methods built on majorization-minimization approaches, benefiting from the Lipschitz differentiabililty of the loss function. Moreover, the proposed approach allows us to handle sparsity-preserving regularizers promoting the selection of the most significant features, so enhancing the performance. Numerical tests and comparisons conducted on

Alessandro Benfenati (e-mail: alessandro.benfenati@unimi.it)
Environmental and Science Department, Via Celoria 2, 20133, Milano, Italy.

Emilie Chouzenoux (e-mail: emilie.chouzenoux@inria.fr)
CVN, Inria, CentraleSupélec, University Paris Saclay, 9 rue Joliot Curie, 91190, Gif-sur-Yvette, France

Giorgia Franchini (e-mail: giorgia.franchini@unimore.it)
Department of Physics, Informatics and Mathematics, Via Campi 213/B, 41125, Modena, Italy.

Salla Latva-Äijö (e-mail: salla.latva-aijo@helsinki.fi)
Department of Mathematics and Statistics, Pietari Kalmin katu 5, 00014 Helsinki, Finland

Dominik Narnhofer (e-mail: dominik.narnhofer@icg.tugraz.at)
Institute for Computer Graphic and Vision, Graz University of Technology, Inffeldgasse 16, 8010 Graz, Austria

Jean–Christophe Pesquet (e-mail: jean-christophe@pesquet.eu)
CVN, Inria, CentraleSupélec, University Paris Saclay, 9 rue Joliot Curie, 91190, Gif-sur-Yvette, France

Sebastian J. Scott (e-mail: ss2767@bath.ac.uk)
Department of Mathematical Sciences, University of Bath, BA2 7AY, United Kingdom

Mahsa Yousefi (e-mail: mahsa.yousefi@phd.units.it)
Department of Mathematics and Geosciences, University of Trieste, via Valerio 12/1, 34127 TS, Italy.

three different datasets demonstrate the good performance of the proposed methodology in terms of qualitative metrics (accuracy, precision, recall, and F_1 score) as well as computational cost.

1 Introduction

Support Vector Machines (SVMs) are well-tailored for regression and classification applications. They were introduced in the seminal work [15] for supervised learning. In addition to being grounded on sound optimization techniques [28, 34], various extensions of them can be performed. They remain one of the most widely used methods in classification tasks despite the increasing role played by neural networks. As linear classifiers, SVMs have been shown to outperform many supervised methods [23, 9, 26]. Real-world applications include image classification [25], face detection [33, 37], hand-written character recognition [21], melanoma classification [1, 2], text categorization [27, 12]. The interested reader can find a complete review in [10].

The supervised learning problem in the SVM framework consists in minimizing a suitable function measuring the distance between the predicted and the true labels corresponding to a dataset sample. This minimization is carried out with respect to the SVM model parameters. The SVM training problem may be formulated as a quadratic programming one [36]. It may involve least squares loss (hard–margin SVM) under a suitable constraint [32], or the hinge loss [12].

For SVM training, the optimization problem can be solved via Lagrangian duality approaches [15, 5], naturally leading to some clever strategies such as kernel tricks [29], splitting the problem into simpler subproblems [5], cutting plane procedures [14]. In several applications, it is common to promote sparsity on the SVM parameters. This is equivalent to implicitly enforcing *feature selection*, meaning that only the features that are essential to a task are kept, and those that are useless or even result in noisy solutions are dropped. A classification task with a limited or severely unbalanced dataset, facing the so-called overfitting problem, is a standard scenario in which this sparsity condition is required. In such a case, standard (nonsparse) SVMs-based models, which are particularly tailored for specific datasets, might not be able in generalization to adapt properly to unseen data. Introducing regularization to achieve this property in the training loss is the most popular method for inducing sparsity on SVM parameters.

The most specifically designed functionals that impose sparsity on a solution are the so-called ℓ_0-norm[1] [35], and its well-known convex relaxation, i.e., ℓ_1-norm [8]. Other functionals, including $\ell_{1,p}$ norms, or elastic-net functionals, have also been shown to effectively promote sparse regularization, see e.g.[31, 30]. To effectively accommodate the additional penalty term, the modification of a training algorithm must be considered. For example, the SVM-based objective function involving the

[1] Recall that actually this function is not a proper norm.

hinge loss and a $\ell_{1,p}$-norm can be efficiently minimized by primal-dual methods [12].

The presented work focuses on training SVMs when we employ the squared hinge loss as a data fidelity function, coupled with a smooth sparsity-promoting regularization functional. In this way, as we will show hereafter, the loss function is Lipschitz differentiable. This paves the way to the application of fast training techniques. This work investigates first-order methods such as the gradient descent algorithm and discusses its acceleration via Majorization–Minimization (MM) techniques.

Contribution. The focus of this work is on training an SVMs-based model using a smooth regularization functional that promotes sparsity and the squared hinge loss as a data fidelity function. This makes the loss function Lipschitz differentiable, as we demonstrate, and allows fast training techniques. Moreover, this work explores and analyzes how Majorization-Minimization (MM) strategies can speed up first-order methods.

Outline. This paper is organized as follows. Together with the regularization functionals taken into consideration in this work, the problem is formulated in Section 2. Section 3 presents the theoretical foundations of MM methods and Section 4 depicts the MM-based algorithms as well as their practical implementation in our context. The goal of Section 5 is to assess the performance of the suggested procedures. Extensive comparisons are conducted with respect to state-of-the-art training algorithms. Finally, Section 6 draws conclusions.

Notation. Bold letters denote vectors, while bold capital letters denote matrices. Greek and italic Latin letters refer to scalars. $\mathbb{R}^n$ is the real Euclidean space of dimension n, $\mathbb{R}^{m\times n}$ denotes the real space of matrices with m rows and n columns. For $\mathbf{x} \in \mathbb{R}^n$ $\|\mathbf{x}\|$ denotes the classical Euclidean (or ℓ_2) norm of the vector. For a matrix $\mathbf{A}$, $\|\mathbf{A}\|$ is the spectral norm of $\mathbf{A}$. The function 1_Ω denotes the binary indicator of the set Ω, $1_\Omega(x) = 1$ if $x \in \Omega$, 0 otherwise.

2 Problem Formulation

The problem addressed in this work is binary classification. Given a new observation $\mathbf{x} \in \mathbb{R}^n$, which contains the describing n scalar features, the aim is to categorize $\mathbf{x}$ into one of two classes. In this section, we present the two main components of the mathematical model we propose to solve this task: the SVM data fidelity term, namely here, the squared hinge loss, and the regularization functionals that promote sparsity.

2.1 SVM loss function

The mathematical model for the categorization of the observation $\mathbf{x}$ into two classes encompasses a linear classifier

$$\begin{aligned} M : \mathbb{R}^n &\to \{-1, 1\} \\ \mathbf{x} &\to \operatorname{sign}(\mathbf{w}^\top \mathbf{x} + \beta) \end{aligned} \tag{1}$$

where $\mathbf{w} \in \mathbb{R}^n$ and $\beta \in \mathbb{R}$. The classifier in (1) defines a separating hyperplane whose purpose is to distinguish between items belonging to different classes, see e.g. Fig. 1 for a 2D example. The output of M in Eq. (1) will be then 1 or -1, and these two labels correspond to the categorization in one of the classes. The

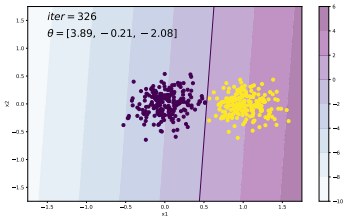


Fig. 1 Toy example of a linear classifier. The line easily separates the two classes.

parameters $\mathbf{w}$ and β in (1) must be estimated during a training phase. Given a dataset $\{(\mathbf{x}_k, y_k)\}_{k=1,\dots,K}$, $\mathbf{x}_k \in \mathbb{R}^n$, $y_k \in \{-1, 1\}$, and where $\mathbf{x}_{k,i}$ denotes the i–th feature of the k–th sample, the ideal training loss would consist in the misclassification count

$$\ell\left(M(\mathbf{x}_k), y_k\right) = \frac{1 - y_k M(\mathbf{x}_k)}{2} = \rho\left(y_k(\mathbf{w}^\top \mathbf{x}_k + \beta)\right) \tag{2}$$

where

$$(\forall \upsilon \in \mathbb{R}) \qquad \rho(\upsilon) = \frac{1 - \operatorname{sign}(\upsilon)}{2}.$$

The training would then be carried on by solving the following optimization problem:

$$\underset{\mathbf{w}\in\mathbb{R}^n,\, \beta\in\mathbb{R}}{\text{minimize}} \quad \sum_{k=1}^{K} \rho\left(y_k(\mathbf{w}^\top \mathbf{x}_k + \beta)\right). \tag{3}$$

Unfortunately, (3) reveals to be a difficult nonconvex problem. To overcome this issue, a popular choice is to subsitute the hinge loss ρ_{hinge} for ρ in (2), where

$$(\forall \upsilon \in \mathbb{R}) \qquad \rho_{\text{hinge}}(\upsilon) = \max\{1 - \upsilon, 0\}. \tag{4}$$

Function (4) provides the minimal convex upper bound of the misclassification count function, see Fig. 2 for a visual inspection.

Remark 1 When the two classes are nonempty and separable, the goal of the training is to find a separating hyperplane such that

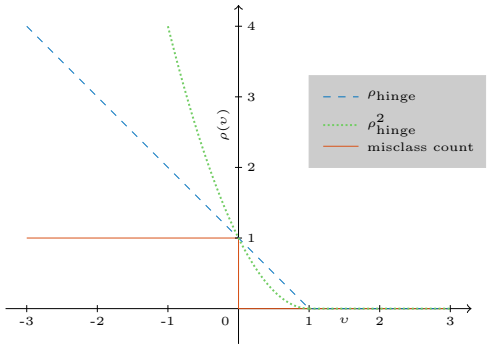


Fig. 2 Hinge loss function (cyan) versus misclassification count function (orange) for the case in which the true label is 1. The hinge loss strongly penalizes uncorrected labels less than 0, while it assumes low values for outputs in $[0, 1]$. Obviously, when the classifier provides the correct label, the loss is zero.

$$\begin{cases} \mathbf{w}^\top \mathbf{x}_k + \beta > 0 & \text{if } y_k = 1 \\ \mathbf{w}^\top \mathbf{x}_k + \beta < 0 & \text{if } y_k = -1 \end{cases}$$

In the case of nonseparable classes, one should employ a slack variable:

$$\begin{cases} \mathbf{w}^\top \mathbf{x}_k + \beta \geq 1 - \xi_k & \text{if } y_k = 1 \\ \mathbf{w}^\top \mathbf{x}_k + \beta \leq -1 + \xi_k & \text{if } y_k = -1 \end{cases}$$

which has the following interpretation:

$$\ell(M(\mathbf{x}_k), y_k) = \min_{\xi_k \in [0,+\infty]} \xi_k \quad \text{s.t.} \quad y_k(\mathbf{w}^\top \mathbf{x}_k + \beta) \geq 1 - \xi_k.$$

The hinge loss in (4) is convex but not differentiable and it may cause numerical issues around $v = 1$. The training can be performed by using primal–dual methods for solving the related optimization problem [12], which are usually costly and might lack flexibility. To overcome this issue, we focus instead on the squared hinge loss

$$(\forall v \in \mathbb{R}) \qquad \rho^2_{\text{hinge}}(v) = \max\{(1 - v)^2, 0\}. \tag{5}$$

Function (5) is convex and differentiable on the entire domain. Moreover, it has a 2–Lipschitz gradient, which is a useful property when dealing with optimization problems. A possible drawback is that the squared hinge loss might be more sensitive than the hinge function with respect to larger errors (see Figure 2 for comparison).

2.2 Regularization

Our work focuses on the regularized version of the squared-hinge loss problem:

$$\underset{\mathbf{w}\in\mathbb{R}^N,\,\beta\in\mathbb{R}}{\text{minimize}} \quad \sum_{k=1}^{K} \rho_{\text{hinge}}^2\big(y_k(\mathbf{w}^\top \mathbf{x}_k + \beta)\big) + f(\mathbf{w}). \tag{6}$$

Various choices can be made for function f, to favor the sparsity of the solution. Below, we list some examples covered by our approach. Namely, we consider

$$\Big(\forall\, \mathbf{w} = (w_i)_{1\leq i\leq N} \in \mathbb{R}^N\Big) \quad f(\mathbf{w}) = \sum_{i=1}^{N} \varphi(w_i) + \frac{\eta}{2}\|\mathbf{w}\|^2, \tag{7}$$

where $\varphi : \mathbb{R} \to \mathbb{R}$ is a potential function and $\eta \geq 0$, for which we introduce the following requirements:

1. φ is even ;
2. φ is differentiable on $\mathbb{R}$;
3. $\varphi\left(\sqrt{\cdot}\right)$ is concave on $[0, +\infty[$.

This framework is rather versatile, as it allows us to consider several interesting choices, such as smooth approximations for the ℓ_1 norm or for the ℓ_0 pseudo–norm. For $\varphi \equiv 0$, we retrieve the standard quadratic penalty often used in SVMs. When $\eta \neq 0$ and φ is a sparse promoting term, f can be viewed as an elastic-net penalty. See Fig. 3 for a visual inspection. Typically, we can use the so-called hyperbolic potential defined, for $\lambda \geq 0$, by

$$(\forall w \in \mathbb{R}) \quad \varphi(w) = \lambda\sqrt{w^2 + \delta^2},\ \delta > 0. \tag{8}$$

Function (8) is a convex function approximating $w \mapsto \lambda|w|$. Another choice is the Welsh potential

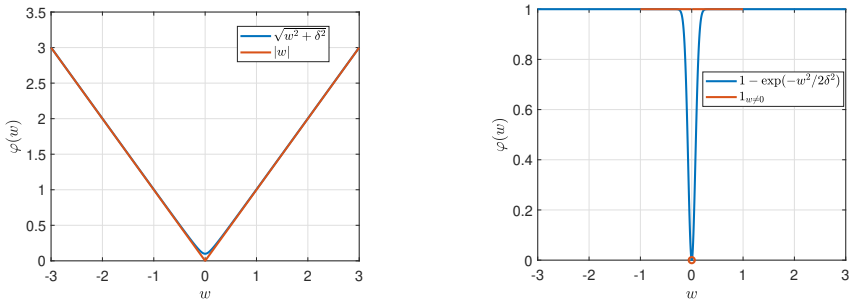
$$(\forall w \in \mathbb{R}) \quad \varphi(w) = \lambda\left(1 - \exp\left(-\frac{w^2}{2\delta^2}\right)\right),\ \delta > 0, \tag{9}$$

Function (9) is nonconvex and approximates the binary indicator function

$$w \mapsto \lambda 1_{w\neq 0}.$$

2.3 General formulation

Problem (6) can be reformulated as in the following way:



(a) (b)

Fig. 3 (a) Absolute value and its smooth approximation with hyperbolic potential. (b) Binary indicator function and its smooth approximation with Welsh potential.

$$\underset{\boldsymbol{\theta}\in\mathbb{R}^{N+1}}{\text{minimize}} \quad (g(\mathbf{L}\boldsymbol{\theta}) + \tilde{f}(\boldsymbol{\theta}) \equiv \Phi(\boldsymbol{\theta})) \tag{10}$$

where

- $\boldsymbol{\theta} = [\mathbf{w}^\top \ \ \beta]^\top \in \mathbb{R}^{N+1}$
- $\mathbf{L} = \mathrm{Diag}(y_1, \cdots, y_K) \begin{bmatrix} \mathbf{x}_1^\top & 1 \\ \vdots & \vdots \\ \mathbf{x}_K^\top & 1 \end{bmatrix} \in \mathbb{R}^{K\times(N+1)}$
- $(\forall\, \mathbf{v} = (v_k)_{1\leq k\leq K})\ g(\mathbf{v}) = \displaystyle\sum_{k=1}^{K} \rho^2_{\text{hinge}}(v_k)$
- $\tilde{f}(\boldsymbol{\theta}) = f(\mathbf{w})$.

Note that the regularization term only affects the variable $\mathbf{w}$ and not the bias β. Function Φ involved in (6) is differentiable on $\mathbb{R}^{N+1}$. Its gradient reads

$$\left(\forall\, \boldsymbol{\theta} \in \mathbb{R}^{N+1}\right) \quad \nabla\Phi(\boldsymbol{\theta}) = \mathbf{L}^\top \nabla g(\mathbf{L}\boldsymbol{\theta}) + \nabla\tilde{f}(\boldsymbol{\theta}). \tag{11}$$

The derivative of the squared hinge loss is

$$(\forall \mathbf{v} \in \mathbb{R}^K) \quad \nabla g(\mathbf{v}) = (\max(2(v_k - 1), 0))_{1\leq k\leq K}\,. \tag{12}$$

Moreover,

$$(\forall \boldsymbol{\theta} \in \mathbb{R}^{N+1}) \quad \nabla\tilde{f}(\boldsymbol{\theta}) = \begin{pmatrix} \varphi'(w_1) + \eta w_1 \\ \vdots \\ \varphi'(w_N) + \eta w_N \\ 0 \end{pmatrix}, \tag{13}$$

with φ' as the derivative of the potential function φ involved in the construction of the regularization term f. In particular, for the hyperbolic potential function (8),

$$(\forall w \in \mathbb{R}) \quad \varphi'(w) = \lambda \frac{w}{\sqrt{w^2 + \delta^2}}. \tag{14}$$

while, for the Welsh potential (9),

$$(\forall w \in \mathbb{R}) \quad \varphi'(w) = \lambda \frac{w}{\delta^2} \exp\left(-\frac{w^2}{2\delta^2}\right) \tag{15}$$

In Section 3, we give some important additional properties of function Φ. Then, in Section 4, we provide a family of algorithms based on the MM principle to solve Problem (10).

3 Majorization Properties

This section is devoted to presenting a key tool as the core of the training algorithms proposed in this work, namely the MM technique and the underlying concept of majorizing approximation. The MM technique consists of alternating between two steps to solve an initial complex optimization problem. The first step involves computing the tangent majorant of the objective function, and the second is to minimize that majorant in order to progressively converge to a reliable minimizer of the original function. The definition of a majorant function for the function Φ in (10) is given in the following.

Definition 1 A tangent majorant $h(\cdot; \boldsymbol{\theta}') : \mathbb{R}^{N+1} \to \mathbb{R}$ of Φ at $\boldsymbol{\theta}' \in \mathbb{R}^{N+1}$ is a function such that

$$\begin{aligned} &h(\boldsymbol{\theta}; \boldsymbol{\theta}') \geq \Phi(\boldsymbol{\theta}) \quad (\forall\, \boldsymbol{\theta} \in \mathbb{R}^{N+1}) \\ &h(\boldsymbol{\theta}'; \boldsymbol{\theta}') = \Phi(\boldsymbol{\theta}') \end{aligned}$$

The general MM iterative scheme then reads:

$$(\forall n \in \mathbb{N}) \quad \boldsymbol{\theta}^{(n+1)} = \operatorname{argmin}_{\boldsymbol{\theta} \in \mathbb{R}^{N+1}} h(\boldsymbol{\theta}; \boldsymbol{\theta}^{(n)}), \tag{16}$$

with some initialization $\boldsymbol{\theta}^{(0)} \in \mathbb{R}^{N+1}$. Under suitable hypotheses on the loss function in (10) and its majorizing approximations, the iterative scheme leads to a sequence converging to a solution [20].

Let us now discuss the construction of reliable majorizing approximations for the considered function Φ.

3.1 Descent lemma majorant

Proposition 1 *Assume that φ is a-Lipschitz differentiable on $\mathbb{R}$, with $a > 0$. Then, function Φ involved in* (10) *is μ-Lipschitz differentiable on $\mathbb{R}^{N+1}$ with*

$$\mu = 2\|\mathbf{L}\|^2 + a + \eta. \tag{17}$$

As a consequence, for every $\boldsymbol{\theta}' \in \mathbb{R}^{N+1}$*, the following function is a tangent majorant of* Φ *at* $\boldsymbol{\theta}'$*,*

$$(\forall \boldsymbol{\theta} \in \mathbb{R}^{N+1}) \quad h(\boldsymbol{\theta}, \boldsymbol{\theta}') = \Phi(\boldsymbol{\theta}') + \nabla\Phi(\boldsymbol{\theta}')^\top(\boldsymbol{\theta} - \boldsymbol{\theta}') + \frac{\mu}{2}\|\boldsymbol{\theta} - \boldsymbol{\theta}'\|^2. \tag{18}$$

Note that functions (8) and (9) are a-Lipschitz differentiable with $a = \frac{\lambda}{\delta}$ and $a = \frac{\lambda}{\delta^2}$, respectively.

3.2 Half-quadratic majorant

The previous majorizing approximation is interesting but might lack accuracy, as its curvature does not depend on the tangency point $\boldsymbol{\theta}'$. Hereafter, we propose a more sophisticated approximation, reminiscent of the constructions in half-quadratic algorithms for image processing [3].

Proposition 2 *For every* $\boldsymbol{\theta}' \in \mathbb{R}^{N+1}$*, the following function is a tangent majorant of function* Φ *involved in Problem* (10)*:*

$$(\forall \boldsymbol{\theta} \in \mathbb{R}^{N+1}) \quad h(\boldsymbol{\theta}; \boldsymbol{\theta}') = \Phi(\boldsymbol{\theta}') + \nabla\Phi(\boldsymbol{\theta}')^\top(\boldsymbol{\theta} - \boldsymbol{\theta}') + \frac{1}{2}(\boldsymbol{\theta} - \boldsymbol{\theta}')^\top \mathbf{A}(\boldsymbol{\theta}')(\boldsymbol{\theta} - \boldsymbol{\theta}') \tag{19}$$

with,

$$(\forall \boldsymbol{\theta} \in \mathbb{R}^{N+1}) \quad \mathbf{A}(\boldsymbol{\theta}) = 2\mathbf{L}^\top\mathbf{L} + \mathrm{Diag}\left(\begin{bmatrix} \psi(\theta_1) + \eta \\ \vdots \\ \psi(\theta_N) + \eta \\ \varepsilon \end{bmatrix}\right), \tag{20}$$

with: $\psi : w \mapsto \varphi'(w)/w$ *and* $\varepsilon > 0$*.*

For the potential (8), we have

$$(\forall w \in \mathbb{R}) \quad \psi(w) = \lambda \frac{1}{\sqrt{w^2 + \delta^2}}, \tag{21}$$

while, for (9),

$$(\forall w \in \mathbb{R}) \quad \psi(w) = \frac{\lambda}{\delta^2} \exp\left(-\frac{w^2}{2\delta^2}\right). \tag{22}$$

4 Training SVMs

In this section, we present a set of MM-based strategies to solve optimization (10). First, using the descent lemma majorant, we describe a basic gradient descent algo-

rithm with constant stepsize. Then, using a more sophisticated majorant construction, we derive an MM quadratic approach and provide a skillful strategy for the inversion of the majorant curvature. We also present a subspace acceleration of the aforementioned MM method. Finally, we discuss the stochastic implementation of the training methods and propose a set of hybrid methods with fast convergence in both warm-up and asymptotic regimes.

4.1 Gradient Descent Approach

The iterative procedure reads as

$$(\forall n \in \mathbb{N}) \quad \boldsymbol{\theta}^{(n+1)} = \boldsymbol{\theta}^{(n)} - \alpha \nabla \Phi(\boldsymbol{\theta}^{n}), \tag{23}$$

with $\boldsymbol{\theta}^{(0)} \in \mathbb{R}^N$. The iterates produced by (23) are guaranteed to converge to a stationary point of (10) for $\alpha \in]0, 2/\mu[$, where μ is defined in Proposition 1.

4.2 MM Quadratic Approach

The gradient descent method is often characterized by a slow convergence. Improved performance can be obtained by the MM quadratic scheme based on Proposition 2:

$$\begin{aligned}(\forall n \in \mathbb{N}) \quad \boldsymbol{\theta}^{(n+1)} &= \operatorname{argmin}_{\boldsymbol{\theta}} \left(\nabla \Phi(\boldsymbol{\theta}^{(n)})^\top (\boldsymbol{\theta} - \boldsymbol{\theta}^{(n)}) + \frac{1}{2} (\boldsymbol{\theta} - \boldsymbol{\theta}^{(n)})^\top \mathbf{A}(\boldsymbol{\theta}^{(n)}) (\boldsymbol{\theta} - \boldsymbol{\theta}^{(n)}) \right) \\ &= \boldsymbol{\theta}^{(n)} - (\mathbf{A}(\boldsymbol{\theta}^{(n)}))^{-1} \nabla \Phi(\boldsymbol{\theta}^{n}). \end{aligned} \tag{24}$$

The iterative scheme (24), related to half-quadratic techniques popular in imaging [3], can be viewed as a preconditioned gradient algorithm. The practical implementation and acceleration of this scheme are discussed below.

4.2.1 Numerically inverting the majorant curvature

The computation of the inverse of $\mathbf{A}(\boldsymbol{\theta}^{(n)})$ at each iteration, in (24), might be time-consuming. We propose an approach for computing the product $(\mathbf{A}(\boldsymbol{\theta}^{(n)}))^{-1} \nabla \Phi(\boldsymbol{\theta}^{(n)})$, without explicitly constructing the inverse of the matrix. Referring to (20), we majorize the curvature matrix as follows

$$(\forall \boldsymbol{\theta} \in \mathbb{R}^{N+1}) \quad \mathbf{A}(\boldsymbol{\theta}) \preceq \overline{\mathbf{A}}(\boldsymbol{\theta}) = 2\mathbf{L}^\top \mathbf{L} + \sigma_{\max}(\boldsymbol{\theta}) \mathbf{I}_{\mathrm{d}}, \tag{25}$$

where

$$\sigma_{\max}(\boldsymbol{\theta}) = \max \left\{ \psi(\theta_1) + \eta, \ldots, \psi(\theta_N) + \eta, \varepsilon \right\}. \tag{26}$$

Suppose $\mathbf{L}^\top = \mathbf{QR}$ is the QR factorization of $\mathbf{L}^\top \in \mathbb{R}^{(N+1)\times K}$, where $\mathbf{Q}$ is an orthogonal matrix of order $N+1$ and $\mathbf{R}$ is a $(N+1)\times K$ trapezoidal matrix (and hence $\mathbf{RR}^\top$ is a symmetric matrix of order $N+1$). Then

$$(\forall \boldsymbol{\theta} \in \mathbb{R}^{N+1}) \quad 2\mathbf{L}^\top\mathbf{L} + \sigma_{\max}(\boldsymbol{\theta})\mathbf{I}_{\mathrm{d}} = 2\mathbf{QRR}^\top\mathbf{Q}^\top + \sigma_{\max}(\boldsymbol{\theta})\mathbf{I}_{\mathrm{d}}. \tag{27}$$

Let $\mathbf{U}\boldsymbol{\Lambda}\mathbf{U}^\top$ be the spectral decomposition of $\mathbf{RR}^\top$ where $\mathbf{U}$ is a matrix whose columns are the eigenvectors of $\mathbf{RR}^\top$, and $\boldsymbol{\Lambda} = \mathrm{Diag}(\lambda_1, \ldots, \lambda_{N+1})$ is diagonal whose elements are the associated eigenvalues. Substituting into (27), we obtain

$$(\forall \boldsymbol{\theta} \in \mathbb{R}^{N+1}) \quad 2\mathbf{L}^\top\mathbf{L} + \sigma_{\max}(\boldsymbol{\theta})\mathbf{I}_{\mathrm{d}} = 2\mathbf{QU}\boldsymbol{\Lambda}\mathbf{U}^\top\mathbf{Q}^\top + \sigma_{\max}(\boldsymbol{\theta})\mathbf{I}_{\mathrm{d}}. \tag{28}$$

Since $\mathbf{Q}$ and $\mathbf{U}$ have orthogonal columns, considering orthogonal matrix $\mathbf{P} = \mathbf{QU}$ yields

$$(\forall \boldsymbol{\theta} \in \mathbb{R}^{N+1}) \quad \overline{\mathbf{A}}(\boldsymbol{\theta}) = \mathbf{P}\hat{\boldsymbol{\Lambda}}(\boldsymbol{\theta})\mathbf{P}^\top, \tag{29}$$

where $\hat{\boldsymbol{\Lambda}}(\theta) = 2\boldsymbol{\Lambda} + \sigma_{\max}(\boldsymbol{\theta})\mathbf{I}_{\mathrm{d}}$. Consequently, constructing $\overline{\mathbf{A}}(\boldsymbol{\theta})$ as defined in (29) allows us to compute its inversion efficiently as follows

$$(\forall \boldsymbol{\theta} \in \mathbb{R}^{N+1}) \quad \left(\overline{\mathbf{A}}(\boldsymbol{\theta})\right)^{-1} = \mathbf{P}\hat{\boldsymbol{\Lambda}}(\boldsymbol{\theta})^{-1}\mathbf{P}^\top. \tag{30}$$

Remark 2 The proposed approach for computing the inverse of curvature matrix approximation might be even more efficient if $K < N$. Indeed, the "thin" QR factorization leads to quickly computing the spectral decomposition $\mathbf{U}\boldsymbol{\Lambda}\mathbf{U}^\top$ of the smaller $K \times K$ matrix $\mathbf{RR}^\top$.

4.2.2 Subspace acceleration

Another approach for reducing the complexity of (24), without jeopardizing its convergence properties, is to resort to a subspace acceleration technique. The method then reads

$$(\forall n \in \mathbb{N}) \quad \boldsymbol{\theta}^{(n+1)} = \boldsymbol{\theta}^{(n)} - \mathbf{D}^{(n)}((\mathbf{D}^{(n)})^\top\mathbf{A}(\boldsymbol{\theta}^{(n)})\mathbf{D}^{(n)})^\dagger(\mathbf{D}^{(n)})^\top\nabla\Phi(\boldsymbol{\theta}^n). \tag{31}$$

Hereabove, † stands for the pseudo-inversion, and $\mathbf{D}^{(n)} \in \mathbb{R}^{(N+1)\times M_n}$ with $M_n \geq 1$ (typically small), is the so-called subspace matrix. A standard choice is

$$(\forall n \in \mathbb{N}) \quad \mathbf{D}^{(n)} = \left[-\nabla\Phi(\boldsymbol{\theta}^{(n)}) \mid \boldsymbol{\theta}^{(n)} - \boldsymbol{\theta}^{(n-1)}\right], \tag{32}$$

(i.e., $M_n = 2$), with the convention $\boldsymbol{\theta}^{(0)} = \mathbf{0}$, which leads to the 3MG (MM Memory Gradient) algorithm. Another simplest possibility is

$$(\forall n \in \mathbb{N}) \quad \mathbf{D}^{(n)} = -\nabla\Phi(\boldsymbol{\theta}^{(n)}), \tag{33}$$

(i.e., $M_n = 1$) which results in a gradient descent technique with varying stepsize

$$(\forall n \in \mathbb{N}) \quad \boldsymbol{\theta}^{(n+1)} = \boldsymbol{\theta}^{(n)} - \frac{\nabla\Phi(\boldsymbol{\theta}^{n)})^\top \nabla\Phi(\boldsymbol{\theta}^{n)})}{\nabla\Phi(\boldsymbol{\theta}^{n)})^\top \mathbf{A}(\boldsymbol{\theta}^{(n)}) \nabla\Phi(\boldsymbol{\theta}^{n)})} \nabla\Phi(\boldsymbol{\theta}^{n)}). \tag{34}$$

Convergence of the iterates produced by (31) to a stationary point of Φ is shown in [13] under mild assumptions. Convergence to the (unique) solution to (10) is obtained when we additionally assume that the potential function φ is convex and $\eta > 0$. Interestingly, the previously introduced schemes (23) and (24) can both be viewed as special cases of (31), and thus inherit the same convergence properties.

4.2.3 Convergence result

Let us now state the theoretical convergence guaranties for the MM quadratic method (24) and its variants.

Theorem 1 *Let φ given by* (8) *or* (9)*. Let $(\boldsymbol{\theta}^{(n)})_{n\in\mathbb{N}}$ be generated either by* (24)*, or* (31)-(32)*, or* (34)*. Then, $(\boldsymbol{\theta}^{(n)})_{n\in\mathbb{N}}$ converges to a stationary point of Φ in* (10)*. Moreover, if $\eta > 0$ and φ is given by* (8)*, Φ is strongly convex, and $(\boldsymbol{\theta}^{(n)})_{n\in\mathbb{N}}$ converges to its unique minimizer.*

Proof Function Φ in (10) is Lipschitz differentiable, by Proposition 1. Moreover, it satisfies Kurdika-Lojasewicz inequality [4] for φ given by (8) or (9). The proof results directly from [13, Theo.3], using Proposition 2, and noticing than (24), (31)-(32), and (34), are all particular cases of an MM quadratic subspace algorithm with one inner iteration. □

4.3 Stochastic minimization approaches

When we face minimization with a particularly large dataset, it is often necessary to use a stochastic technique based on minibatches [7]. The same issue arises in the context of online learning [6] when the entire dataset is not completely available at the beginning of the learning process. Employing a stochastic method may also be convenient for the speed of convergence, especially in a warm-up phase (i.e., first iterations). The stochastic gradient descent updates the current iterate by a gradient calculated on a single sample $(\mathbf{x}_k, y_k)$ with randomly chosen $k \in \{1, \dots, K\}$ in order to lighten the computational cost. For every $k \in \{1, \dots, K\}$, let us denote

$$(\forall \boldsymbol{\theta} = [\mathbf{w}^\top \ \beta]^\top \in \mathbb{R}^{N+1}) \quad \Phi_k(\boldsymbol{\theta}) = \rho^2_{\text{hinge}}(y_k(\mathbf{w}^\top \mathbf{x}_k + \beta)) + f(\mathbf{w}), \tag{35}$$

$$= \rho^2_{\text{hinge}}(\mathbf{L}_k^\top \boldsymbol{\theta}) + \widetilde{f}(\boldsymbol{\theta}), \tag{36}$$

with $\mathbf{L}_k \in \mathbb{R}^{N+1}$ as the k-th row of $\mathbf{L}$. We deduce the gradient for the k-th sample

$$(\forall \boldsymbol{\theta} \in \mathbb{R}^{N+1}) \quad \nabla\Phi_k(\boldsymbol{\theta}) = \mathbf{L}_k \max(2(\mathbf{L}_k^\top \boldsymbol{\theta} - 1), 0) + \nabla\widetilde{f}(\boldsymbol{\theta}).$$

We present in Algorithm 1 a basic constant stepsize implementation of the stochastic gradient descent method.

Algorithm 1 Stochastic Gradient (SG) Method

1: Choose an initial iterate $\boldsymbol{\theta}_0$, the stepsize $\alpha > 0$ and the maximum number of iterates $maxit$.
2: **for** $n \in \{0, \ldots, maxit\}$ **do**
3: Draw at random an index $\kappa^{(n)} \in \{1, \ldots, K\}$.
4: Compute the stochastic descent direction $\nabla \Phi_{\kappa^{(n)}}(\boldsymbol{\theta}^{(n)})$.
5: Set the new iterate as $\theta^{(n+1)} \leftarrow \boldsymbol{\theta}^{(n)} - \alpha \nabla \Phi_{\kappa^{(n)}}(\boldsymbol{\theta}^{(n)})$
6: **end for**

In the same fashion, we can also adopt Momentum [24] and AdaM [22] methods, described in Algorithm 2 and Algorithm 3, respectively.

Algorithm 2 Momentum

1: Choose an initial iterate $\boldsymbol{\theta}^{(0)}$, the stepsize $\alpha > 0$, the maximum number of iterates $maxit$ and $\beta \in [0, 1)$
2: Initialize $\mathbf{m}_0 \leftarrow 0$
3: **for** $n \in \{1, \ldots, maxit\}$ **do**
4: Draw at random an index $\kappa^{(n)} \in \{1, \ldots, K\}$.
5: Compute the stochastic descent direction $\nabla \Phi_{\kappa^{(n)}}(\boldsymbol{\theta}^{(n)})$.
6: $\mathbf{m}^{(n+1)} \leftarrow \beta \mathbf{m}^{(n)} + \nabla \Phi_{\kappa^{(n)}}(\boldsymbol{\theta}^{(n)})$
7: $\boldsymbol{\theta}^{(n)} \leftarrow \boldsymbol{\theta}^{(n-1)} - \alpha \mathbf{m}^{(n+1)}$
8: **end for**

Algorithm 3 Adam

1: Choose an initial iterate $\boldsymbol{\theta}_0$, the stepsize $\alpha > 0$, the maximum number of iterates $maxit$, $\hat{\epsilon}$, β_1 and $\beta_2 \in [0, 1)$;
2: Initialize $\mathbf{m}^{(0)} \leftarrow 0$, $\mathbf{v}^{(0)} \leftarrow 0$
3: **for** $n \in \{1, \ldots, maxit\}$ **do**
4: Draw at random an index $\kappa^{(n)} \in \{1, \ldots, K\}$.
5: Compute the stochastic descent direction $\nabla \Phi_{\kappa^{(n)}}(\boldsymbol{\theta}^{(n)})$.
6: $\mathbf{m}^{(n)} \leftarrow \beta_1 m^{(n-1)} + (1 - \beta_1) \nabla \Phi_{\kappa^{(n)}}(\boldsymbol{\theta}^{(n)})$
7: $\mathbf{v}^{(n)} \leftarrow \beta_2 \mathbf{v}^{(n-1)} + (1 - \beta_2) \nabla \Phi_{\kappa^{(n)}}(\boldsymbol{\theta}^{(n)}) \otimes \nabla \Phi_{\kappa^{(n)}}(\boldsymbol{\theta}^{(n)})$
8: $\alpha^{(n)} = \alpha \dfrac{\sqrt{1 - \beta_2^n}}{(1 - \beta_1^n)}$
9: $\boldsymbol{\theta}^{(n)} \leftarrow \boldsymbol{\theta}^{(n+1)} - \alpha^{(n)} \mathbf{m}^{(n+1)} \oslash (\sqrt{\mathbf{v}^{(n+1)}} + \hat{\epsilon})$
10: **end for**

In Algorithm 3, $\otimes$ in step 7 denotes the element-wise product, and $\oslash$ in step 9 is the element-wise division.

Remark 3 For simplicity, only one index $\kappa^{(n)}$ is selected at each iteration of the above schemes. One may employ the idea of *minibatch*, where a set $\mathcal{B}$ of B indexes

is randomly chosen, and the descent direction is given by the weighted sum of the gradients. For example, step 5 in Algorithm 1 becomes

$$\boldsymbol{\theta}^{(n+1)} \leftarrow \boldsymbol{\theta}^{(n)} - \alpha \frac{1}{B} \sum_{i \in \mathcal{B}} \nabla \Phi_i(\boldsymbol{\theta}^{(n)})$$

Algorithms 1, 2 and 3 are effective when the hyperparameters are fine-tuned. Several papers in the literature investigate how to develop reliable stepsize selection strategies in stochastic methods, mainly in an adaptive manner [19, 18, 17]. Details on the hyperparameter choice will be given in Section 5.

4.3.1 Hybrid approach

Considering the practical benefits of stochastic methods, and while keeping in mind their convergence-related constraints, we propose to introduce a hybrid strategy. It consists in using stochastic methods to minimize the objective function for a preset $\iota \in \mathbb{N}^*$ number of iterations, and, thus, taking the advantage of their initial learning speed characteristic. After this phase (the *warm-up*), the iterate $\boldsymbol{\theta}^{(\iota)}$ obtained from the stochastic methodology is used as an initial point of the deterministic method, benefiting from more stable convergence. Special attention should be paid to the choice of ι, which must result from a trade-off between the benefit offered by the initial speedup of stochastic methods and the convergence properties of deterministic methods. The choice of ι will also be discussed in Section 5.

Remark 4 From the perspective of convergence guarantees, this warm-up phase has no impact, since it is equivalent to choosing a specific initial point $\boldsymbol{\theta}^{(0)}$.

5 Numerical Experiments

This section is devoted to numerically assessing the performance of the proposed methods. In particular, we consider three different datasets summarized in Table 1. We split each dataset into training and testing sets, and we use 80% of the elements for the training set and the remaining 20% for the testing set.

dataset	$N+1$	K_{training}	K_{testing}
a1a	120	1284	321
cina0	133	12827	3206
w8a	301	39800	9949

Table 1 Data set characteristics.

The datasets `a1a` and `w8a` can be found at [11], while `cina0` is available at [16]. We minimize the loss in Eq. (10) with three different choices for the regularizer term $\tilde{f}(\boldsymbol{\theta})$, namely $\varphi = 0$ (i.e., squared l_2 norm regularization), or φ equal to the potential either (8) or (9). We emphasize that the case when a squared l_2 norm is adopted as the regularizer in addition to the fidelity term in the loss is entirely comparable to the formulation of SVM's primary problem; thus ensuring an experimental comparison with standard SVM as well.

Remark 5 (Hyperparameters setting) As detailed in Section 4, the choice of proper hyperparameters is crucial for the speed and convergence of stochastic methods. Starting with stochastic methods (SG, Momentum, AdaM) special attention was paid to the choice of the learning rate, which was manually set after an exhaustive search for the optimal one in each method. All other hyperparameters were chosen as default ones found in the literature. In general, 100 iterations in the deterministic case (or epochs in the stochastic case) appeared enough for all methods. In the case of hybrid method we consider a total of 100 epochs+iterates. In the hybrid case, we set the warm-up parameter to $\iota = 10$, considering it a good trade-off between the speed of stochastic methods and the stability of deterministic ones. In a nutshell, we perform 10 stochastic epochs for the warm-up phase, and then 90 deterministic steps (or epochs, which is the same in the deterministic case).
Moreover, we empirically set $\eta = 10^{-4}$ ans $\lambda = \delta = 0$ to experiment only ℓ_2 regularizer, while $\lambda = \delta = 10^{-4}$ and $\eta = 0$ is used for other regularizers, which lead to fair performance on all datasets. A discussion on the influence of λ on the results is provided at the end of the section.

5.1 Results

To test the effectiveness of the methods on the datasets in the Table 1, we will compare the results of various methods associated with different regularizers. In particular, we start by comparing stochastic methods to choose which one is most suitable for the warm-up phase.

We define *optimality gap* as the difference between the value of the loss function at the point and the function calculated at an estimate of its minimum. This estimate is derived by letting a deterministic method run for thousands of iterates.

In Figure 4 we compare the optimality gap of `a1a` dataset with smooth l_1 norm regularizer. As we can see, the best method at the beginning is AdaM: the behaviour on the other datasets is similar, hence we employ AdaM for the warm-up phase.

5.1.1 Optimality gap

The methods we are going to compare are as follows:

- Gradient descent approach (23) called FULL GRADIENT (FG)

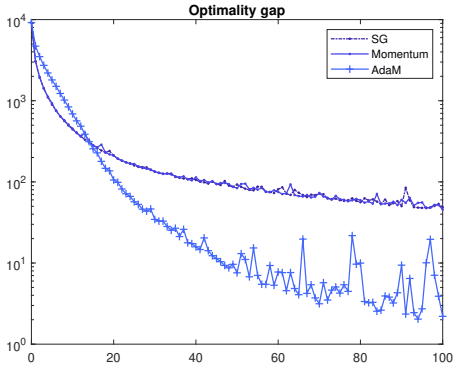


Fig. 4 Stochastic methods with `a1a` dataset and smooth l_1 norm like regularizer.

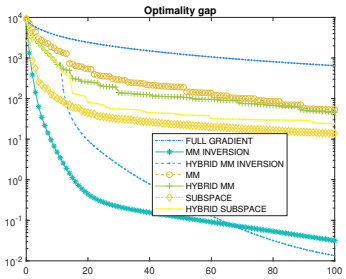
- MM quadratic approach (24) with approximated inverse curvature (30) called MM INVERSION (MM I)
- MM quadratic approach (24) with approximated inverse curvature (30) and an initial warm up of 10 AdaM iterates called HYBRID MM INVERSION (H MM I)
- MM quadratic approach (24) with exact curvature
- MM quadratic approach (24) with exact curvature and an initial warm-up of 10 AdaM iterates called HYBRID MM (H MM)
- MM quadratic with subspace acceleration method (31) called SUBSPACE (SUB)
- MM quadratic with subspace acceleration method (31) and an initial warm-up of 10 AdaM iterates called HYBRID SUBSPACE (H SUB).

In the plots of Figure 5 we consider on the x-axis the number of epochs, where an epoch in the deterministic framework means an iterate, whilst in the stochastic framework is a full vision of the dataset. In the y-axis we consider the optimality gap.

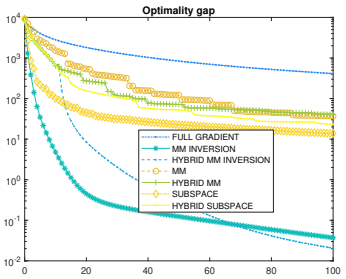
As we can observe from Figure 5, hybrid methods outperform all the deterministic methods: the warm up strategy pays off even when a low number of warm up iteration ι is set. Moreover, MM methods seems to exploit the second order information of the functional, allowing an evident boost towards the solution. Among these mixed strategies (MM plus warm up), HYBRID MM INVERSION overcomes all the other, reaching the best optimality gap among all the coupling dataset-regularizer.

5.1.2 Performance measure

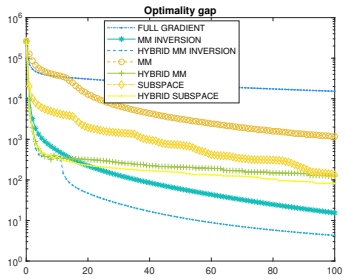
In the previous section, we presented results for the optimality gap, calculated on the training set. Here we present performance measures calculated on the test set, to emphasize the generalization ability on unseen data of the proposed method. In this regard, we considered four performance measures well known in the literature:



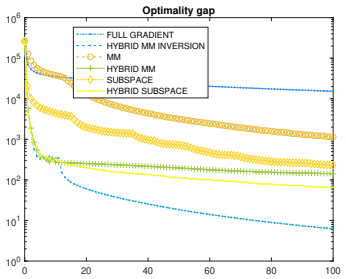
(a) `a1a` with Welsh potential.



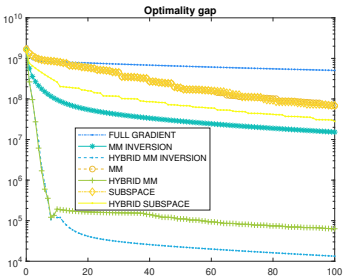
(b) `a1a` with l_2 norm only.



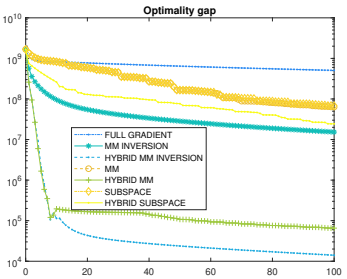
(c) `w8a` with hyperbolic potential.



(d) `w8a` with l_2 norm only.



(e) `cina0` with Welsh potential.



(f) `cina0` with l_2 norm only.

Fig. 5 Optimality gap for different dataset-regularizer combinations. In panel (d) the method MM INVERSION has been removed since it appeared instable.

accuracy, precision, recall, and F1-score. In our framework of binary classification, we denote with ”positive” and ”negative” the two classes, corresponding to the labels 1 and -1 in Eq. (1), to be consistent with standard definitions of the previous measures. True Positive (TP) denotes the number of elements of the datasets that are correctly classified as ”positive”, while True Negative (TN) is the number of elements that are correctly classified as ”negative”. These two numbers denote the correct classifications. On the other hand, False Positive (FP) and False Negative

(FN) denote the elements that are actually negative and are classified as "positive", and vice versa. This terminology stems from classification tasks in medicine and biology. The performance measures considered can be expressed as

$$\text{Accuracy} = \frac{TP+TN}{K_{\text{testing}}}, \qquad \text{Precision} = \frac{TP}{TP+FN},$$
$$\text{Recall} = \frac{TP}{TP+FN}, \qquad \text{F}_1\text{-score} = \frac{TP}{TP+\frac{FN+FP}{2}}.$$

	Reg.	FG	MMI	H MMI	MM	H MM	SUB	H SUB
Accuracy	(8)	0.7944	0.8100	0.8100	0.8193	**0.8255**	0.8193	**0.8255**
	(9)	0.7788	0.8037	0.8100	0.8224	0.8224	0.8193	0.8126
	×	0.7944	0.8100	0.8100	0.8224	0.8224	0.8193	0.8193
Recall	(8)	0.6279	0.6753	0.6753	0.6974	0.7013	0.6974	**0.7105**
	(9)	0.6000	0.6623	0.6753	0.7013	0.7013	0.6974	0.6883
	×	0.6279	0.6753	0.6753	0.6962	0.7067	0.6974	0.6976
Precision	(8)	0.6136	0.5909	0.5909	0.6023	0.6136	0.6023	0.6136
	(9)	0.5795	0.5795	0.5909	0.6136	0.6136	0.6023	0.6023
	×	0.6136	0.5909	0.5909	**0.6250**	0.6023	0.6023	0.6023
F_1	(8)	0.6207	0.6303	0.6303	0.6463	**0.6585**	0.6463	0.6585
	(9)	0.5896	0.6182	0.6303	0.6545	0.6545	0.6463	0.6424
	×	0.6207	0.6303	0.6303	0.6587	0.6503	0.6463	0.6463

Table 2 Performance measures for `a1a` dataset. (8) and (9) refers to the choice of the regularization functional, while '×' denotes the case where $\varphi = 0$ (i.e., only ℓ_2-norm regularization is used). The names of the methods refer to the list presented at the beginning of Section 5.1.1.

In Tables 2-4 we report the results of the performance measures, each table referring to a different dataset. The second row identifies the type of regularizer. These tables clearly show how hybrid methods provide higher performances when a fixed number of iterations is selected: this confirms that this approach allows to start the deterministic method from a suitable initial point, reaching sooner (with respect to a deterministic method) a reliable estimation of the solution. This behaviour is clear from the plots in Figure 5. This numerical results hence show that they are able to generalize and that they are particularly effective even on data not seen in the training phase.

The choice of a suitable regularization functional induces more reliable results, considering all the performance indices: sparse-preserving functions are providing with higher scores among `a1a` and `w8a` datasets, while in `cina0` the ℓ_2 penalty seems enough to get good results.

In Table 5, we report the computational time in seconds for training the various methods. When we refer to computational time, we mean the entire training phase on

	Reg.	FG	MMI	H MMI	MM	H MM	SUB	H SUB
Accuracy	(8)	0.9607	0.9916	**0.9919**	0.9876	0.9912	0.9914	0.9916
	(9)	0.9602	0.9837	0.9918	0.9876	0.9916	0.9911	0.9913
	×	0.9607	0.6471	0.9917	0.9876	0.9912	0.9910	0.9918
Recall	(8)	0.1362	0.8000	**0.8205**	0.6299	0.7931	0.8182	0.8051
	(9)	0.1341	0.4837	0.8033	0.6299	0.7813	0.8073	0.7717
	×	0.1362	0.0187	0.8067	0.6299	0.7982	0.8173	0.8136
Precision	(8)	0.2821	0.6154	0.6154	0.5128	0.5897	0.5769	0.6090
	(9)	0.2821	0.5705	0.6282	0.5128	**0.6410**	0.5641	0.6282
	×	0.2821	0.4167	0.6154	0.5128	0.5833	0.5449	0.6154
F_1	(8)	0.1837	0.6957	0.7033	0.5654	0.6765	0.6767	0.6934
	(9)	0.1818	0.5235	**0.7050**	0.5654	0.7042	0.6642	0.6926
	×	0.1837	0.0357	0.6982	0.5654	0.6741	0.6538	0.7007

Table 3 Performance measures for `w8a` dataset. (8) and (9) refers to the choice of the regularization functional, while '×' denotes $\varphi = 0$ (i.e., only ℓ_2-norm regularization). The names of the methods refer to the list presented at the beginning of Section 5.1.1.

	Reg.	FG	MMI	H MMI	MM	H MM	SUB	H SUB
Accuracy	(8)	0.7748	0.9195	0.9245	0.8531	0.9148	0.8525	0.8668
	(9)	0.7748	0.9195	0.9245	0.8534	0.9148	0.8531	0.8696
	×	0.7748	0.9195	**0.9248**	0.8534	0.9145	0.8531	0.8677
Recall	(8)	0.5615	0.8000	0.8608	0.7138	0.8261	0.7121	0.7378
	(9)	0.5615	0.8501	0.8600	0.7178	0.8253	0.7138	0.7500
	×	0.5615	0.8501	**0.8610**	0.7121	0.8236	0.7133	0.7416
Precision	(8)	0.5845	0.6154	0.8442	0.7198	0.8490	0.7198	0.7512
	(9)	0.5845	0.8357	0.8454	0.7126	0.8502	0.7198	0.7428
	×	0.5845	0.8357	0.8454	0.7258	**0.8514**	0.7210	0.7488
F_1	(8)	0.5728	0.6957	0.8524	0.7168	0.8374	0.7159	0.7445
	(9)	0.5728	0.8429	0.8526	0.7152	0.8376	0.7168	0.7464
	×	0.5828	0.8429	**0.8531**	0.7189	0.8373	0.7171	0.7452

Table 4 Performance measures for `cina0` dataset. (8) and (9) refers to the choice of the regularization functional, while '×' denotes $\varphi = 0$ (i.e. only ℓ_2-norm regularization). The names of the methods refer to the list presented at the beginning of Section 5.1.1.

all the elements of the dataset, thus referring to the 100 epochs/iterations. All experiments were run on an Intel(R) Core(TM) i7-7700HQ CPU @ 2.80GHz 2.81 GHz. For completeness, all combinations of datasets and regularizers have been reported. The faster training is performed by FG, H MMI, H MM, and SUB approaches. The best compromise between performance and complexity is achieved by hybrid MM methods.

We evaluate in Tables 6-8 the influence of setting parameter λ, when choosing the convex regularizer (8). As expected, the sparsity (i.e., number of zero coefficients) in the retrieved coefficients increases with λ. We can also notice that the best classification metrics are obtained for an intermediary value of λ. This is in particular the

	Reg.	FG	MMI	H MMI	MM	H MM	SUB	H SUB
a1a	(8)	0.010434	0.161774	0.031648	0.161625	0.025913	0.036542	0.164651
	(9)	0.043637	0.186570	0.031504	0.160993	0.034640	0.043813	0.180505
	×	0.173047	1.769611	0.223276	1.680880	0.198365	0.211096	1.738608
cina0	(8)	0.181315	1.857992	0.225515	1.808992	0.231394	0.224193	1.888193
	(9)	0.191336	1.986022	0.230983	2.019854	0.255824	0.297928	1.975059
	×	0.173007	1.698734	0.220615	1.702079	0.214507	0.249245	2.063324
w8a	(8)	1.098484	8.006783	1.843458	8.389577	1.112776	1.288889	10.018341
	(9)	1.074437	11.936826	20.957777	13.887202	1.180779	1.133178	13.653358
	×	1.060243	7.085468	1.596996	8.402317	1.279365	1.217616	8.584693

Table 5 Time in second for all the datasets. (8) and (9) refers to the choice of the regularization functional, while '×' denotes $\varphi = 0$ (i.e., only ℓ_2-norm regularization). The names of the methods refer to the list presented at the beginning of Section 5.1.1.

λ	sparsity	accuracy	precision	recall	F1-score
10^{-1}	55/120	**0.8162**	**0.6986**	0.5795	0.6335
10^{-2}	35/120	**0.8162**	**0.68838**	**0.6023**	**0.6424**
10^{-3}	21/120	0.8100	0.6753	0.5909	0.6303
10^{-4}	18/120	0.8100	0.6753	0.5909	0.6303
10^{-5}	16/120	0.8100	0.6753	0.5909	0.6303

Table 6 Sparsity ratio and classification metrics for the a1a dataset with (8) regularization functional, for different λ choices.

λ	sparsity	accuracy	precision	recall	F1-score
10^{-1}	32/133	0.9236	0.8577	0.8394	0.8509
10^{-2}	4/133	0.9242	0.8625	0.8406	0.8514
10^{-3}	0/133	0.9242	0.8571	0.8478	0.8525
10^{-4}	1/133	**0.9261**	**0.8635**	0.8478	**0.8556**
10^{-5}	0/133	0.9258	0.8606	**0.8502**	0.8554

Table 7 Sparsity ratio and classification metrics for the cina0 dataset with (8) regularization functional, for different λ choices.

λ	sparsity	accuracy	precision	recall	F1-score
10^{-1}	249/300	0.9921	0.8469	0.6026	0.7041
10^{-2}	159/300	**0.9922**	**0.8482**	0.609	**0.709**
10^{-3}	44/300	0.9921	0.8407	0.609	0.7063
10^{-4}	33/300	0.9919	0.8205	**0.6154**	0.7033
10^{-5}	5/300	0.9919	0.8205	**0.6154**	0.7033

Table 8 Sparsity ratio and classification metrics for the w8a dataset with (8) regularization functional, for different λ choices.

case for a1a and w8a datasets. This emphasizes the important role of the introduced sparsifying penalty.

6 Conclusions

This paper revisits existing approaches for training Support Vector Machines by considering modern developments around MM strategies. The novel family of proposed optimization methods address formulations combining a square hinge loss data fidelity function with a smooth sparsity-promoting regularization functional. This combination results in a differentiable objective function, enabling the use of efficient optimization methods for training. The numerical tests performed on three datasets show that the proposed approaches provide reliable results in terms of accuracy, precision, recall, and F1 score and that a hybrid approach integrating some stochastic gradient iterations as a warm up provides an initial boost that leads to a better performance. The results demonstrate that this new approach for training SVMs can be used effectively in a variety of real-world applications, also in a big data context with the joint use of stochastic gradient methods.
A natural extension of this work would be to investigate multi-class formulations of SVMs.

References

1. S. Afifi, H. GholamHosseini, and R. Sinha. A system on chip for melanoma detection using FPGA-based SVM classifier. *Microprocessors and Microsystems*, 65:57–68, 2019.
2. S. Afifi, H. GholamHosseini, and R. Sinha. Dynamic hardware system for cascade SVM classification of melanoma. *Neural Computing and Applications*, 32(6):1777–1788, 2020.
3. M. Allain, J. Idier, and Y. Goussard. On global and local convergence of half-quadratic algorithms. *IEEE Transactions on Image Processing*, 15(5):1130–1142, 2006.
4. H. Attouch, J. Bolte, and B.F. Svaiter. Convergence of descent methods for semialgebraic and tame problems: proximal algorithms, forward backward splitting, and regularized Gauss-Seidel methods. *Mathematical Programming*, pages 1–39, 2011.
5. M. Blondel, A. Fujino, and N. Ueda. Large-scale multiclass support vector machine training via euclidean projection onto the simplex. In *2014 22nd International Conference on Pattern Recognition*, pages 1289–1294, 2014.
6. L. Bottou. Online algorithms and stochastic approximations. In D. Saad, editor, *Online Learning and Neural Networks*. Cambridge University Press, Cambridge, UK, 1998. revised, 2018.
7. L. Bottou, F. E. Curtis, and J. Nocedal. Optimization methods for large-scale machine learning. *SIAM Review*, 60(2):223–311, 2018.
8. P. S. Bradley and O. L. Mangasarian. Feature selection via concave minimization and support vector machines. In *Proceedings of the Fifteenth International Conference on Machine Learning*, volume 98, pages 82–90, 1998.
9. J. Cervantes, F Garcia-Lamont, A. López-Chau, L. Rodriguez-Mazahua, and J Sergio Ruíz. Data selection based on decision tree for svm classification on large data sets. *Applied Soft Computing*, 37:787–798, 2015.

10. J. Cervantes, F. Garcia-Lamont, L. Rodríguez-Mazahua, and A. Lopez. A comprehensive survey on support vector machine classification: Applications, challenges and trends. *Neurocomputing*, 408:189–215, 2020.
11. C.-C. Chang and C.-J. Lin. LIBSVM: A library for support vector machines. *ACM Transactions on Intelligent Systems and Technology*, 2:27:1–27:27, 2011. Software available at http://www.csie.ntu.edu.tw/ cjlin/libsvm.
12. G. Chierchia, N. Pustelnik, and J.-C. Pesquet. Random primal-dual proximal iterations for sparse multiclass svm. In *2016 IEEE 26th International Workshop on Machine Learning for Signal Processing (MLSP)*, pages 1–6, 2016.
13. E. Chouzenoux, A. Jezierska, J.-C. Pesquet, and H. Talbot. A majorize-minimize subspace approach for $\ell_2 - \ell_0$ image regularization. *SIAM Journal on Imaging Sciences*, 6(1):563–591, 2013.
14. D. Chu, C. Zhang, and Q. Tao. A faster cutting plane algorithm with accelerated line search for linear svm. *Pattern Recognition*, 67:127–138, 2017.
15. C. Cortes and V. Vapnik. Support-vector networks. *Machine Learning*, 20(3):273–297, 1995.
16. Zurich ETH. Cina (census is not adult). http://www.causality.inf.ethz.ch/data/CINA.html. Accessed: 31-01-23.
17. G. Franchini, F. Porta, V. Ruggiero, and I. Trombini. A line search based proximal stochastic gradient algorithm with dynamical variance reduction, 2022.
18. G. Franchini, V. Ruggiero, and Zanni. Ritz-like values in steplength selections for stochastic gradient methods, 2020.
19. G. Franchini, V. Ruggiero, and L. Zanni. Steplength and mini-batch size selection in stochastic gradient methods. In *Machine Learning, Optimization, and Data Science*, pages 259–263, Cham, 2020. Springer International Publishing.
20. M. W. Jacobson and J. A. Fessler. An expanded theoretical treatment of iteration-dependent Majorize-Minimize algorithms. *IEEE Transations on Image Processing*, 16(10):2411–2422, Oct. 2007.
21. S. Joseph and J. George. Handwritten character recognition of modi script using convolutional neural network based feature extraction method and support vector machine classifier. In *2020 IEEE 5th International Conference on Signal and Image Processing (ICSIP)*, pages 32–36, 2020.
22. D.P. Kingma and J. Ba. Adam: A method for stochastic optimization. *arXiv: 1412.6980 [cs.LG]*, 2017.
23. X. Liang, L. Zhu, and D.-S. Huang. Multi-task ranking svm for image cosegmentation. *Neurocomputing*, 247:126–136, 2017.
24. N. Loizou and P. Richtarik. Momentum and stochastic momentum for stochastic gradient, newton, proximal point and subspace descent methods. *arXiv:1712:09677v2*, 2018.
25. U. Maulik and D. Chakraborty. Remote sensing image classification: A survey of support-vector-machine-based advanced techniques. *IEEE Geoscience and Remote Sensing Magazine*, 5(1):33–52, 2017.
26. V. A Naik and A. A. Desai. Online handwritten gujarati character recognition using svm, mlp, and k-nn. In *2017 8th International Conference on Computing, Communication and Networking Technologies (ICCCNT)*, pages 1–6. IEEE, 2017.
27. R. H.W. Pinheiro, G. D.C. Cavalcanti, and R. Tsang. Combining binary classifiers in different dichotomy spaces for text categorization. *Applied Soft Computing*, 76:564–574, 2019.
28. John Platt. Sequential minimal optimization: A fast algorithm for training support vector machines. Technical Report MSR-TR-98-14, Microsoft, April 1998.
29. H. Qi and S. Hughes. Using the kernel trick in compressive sensing: Accurate signal recovery from fewer measurements. In *2011 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pages 3940–3943, 2011.
30. K. Qi and H. Yang. Elastic net nonparallel hyperplane support vector machine and its geometrical rationality. *IEEE Transactions on Neural Networks and Learning Systems*, pages 1–11, 2021.
31. L Rosasco, S. Villa, S. Mosci, M. Santoro, and A. Verri. Nonparametric sparsity and regularization. *Journal of Machine Learning Research*, 14:1665–1714, 2013.

32. S. Shalev-Shwartz and S. Ben-David. *Understanding machine learning: From theory to algorithms*. Cambridge university press, 2014.
33. H.-H. Tsai and Y.-C. Chang. Facial expression recognition using a combination of multiple facial features and support vector machine. *Soft Computing*, 22(13):4389–4405, 2018.
34. V. Vapnik. *The Nature of Statistical Learning Theory*. Springer science & business media, 1999.
35. J Weston, A. Elisseeff, B. Schölkopf, and M. Tipping. Use of the zero norm with linear models and kernel methods. *The Journal of Machine Learning Research*, 3:1439–1461, 2003.
36. G. Zanghirati and L. Zanni. A parallel solver for large quadratic programs in training support vector machines. *Parallel Computing*, 29(4):535–551, 2003.
37. S. Zhan, Q.-Q. Tao, and X.-H. Li. Face detection using representation learning. *Neurocomputing*, 187:19–26, 2016. Recent Developments on Deep Big Vision.