A Majorization-Minimization approach for multiclass classification in a big data scenario

Filippo Camellini[1] Emilie Chouzenoux[2] Giorgia Franchini[3]
Jean–Christophe Pesquet[4] Federica Porta[5]

January 8, 2026

[1]Department of Physics, Informatics and Mathematics, University of Modena and Reggio Emilia, Via Campi 213/B, 41125 Modena, Italy. filippo.camellini@unimore.it

[2]CVN, Inria, CentraleSupélec, University Paris-Saclay, 9 rue Joliot Curie, 91190 Gif-sur-Yvette, France. emilie.chouzenoux@inria.fr

[3]Corresponding author. Department of Physics, Informatics and Mathematics, University of Modena and Reggio Emilia, Modena, Italy. giorgia.franchini@unimore.it

[4]CVN, Inria, CentraleSupélec, University Paris-Saclay, France. jean-christophe@pesquet.eu

[5]Department of Physics, Informatics and Mathematics, University of Modena and Reggio Emilia, Modena, Italy. federica.porta@unimore.it

Abstract

This work presents a novel optimization approach for training linear classifiers in multiclass classification tasks, when focusing on a regularized and smooth Weston-Watkins support vector machine (SVM) model. We propose a Majorization-Minimization (MM) algorithm to solve the resulting, Lipschitz-differentiable, optimization problem. To enhance scalability of the algorithm when tackling large datasets, we introduce an incremental MM strategy that suitably integrates the second-order information from the MM strategy within a low-complexity incremental gradient scheme. We establish convergence guarantees of the algorithm for both convex and non-convex settings and demonstrate its effectiveness through various numerical experiments. In particular, by incorporating kernel principal component analysis and foundation models at preprocessing time, we demonstrate that optimizing a linear multiclass SVM using the proposed incremental MM scheme achieves results comparable to state-of-the-art deep learning methods on benchmark tasks.

Keywords

Majorization-minimization Incremental Minimization Machine Learning Multiclass Classification Support Vector Machine Foundation Models

0.1 Introduction

This work addresses the problem of training a linear classifier for multiclass classification tasks. Linear models offer several advantages compared to more complex models, including lower training costs and a reduced number of parameters to store during the inference phase. Moreover, the linear relationship between model parameters and problem variables enhances interpretability, making it well suited for feature selection and dimensionality reduction by identifying the most significant variables. However, the performance of linear predictors is generally lower than that of most nonlinear models, particularly when the dataset is not linearly separable. Despite this, linear classifiers remain a valuable option when combined with preprocessing techniques that map the input data into a more linearly separable space. Examples of such techniques include Kernel Principal Component Analysis [46], and foundation models [35, 41]. These methods remap the original dataset's features into a more meaningful representation, enabling linear classifiers to achieve classification performance comparable to deep learning-based approaches while requiring significantly fewer trainable parameters.

Among the various linear classifiers for multiclass classification, one of the most significant is the Support Vector Machine (SVM), which extends the margin-based optimization framework from binary classification to handle multiple classes effectively.

Motivated by these considerations, the main aim of this work is to develop a Majorization-Minimization (MM) method to solve the primal optimization problem arising in linear SVM for multiclass classification. In particular, we focus here on the variant of multiclass SVMs proposed by Weston and Watkins [54], which directly learns a linear classifier without reducing the problem to multiple binary classifications (such as One-vs-All or One-vs-One). Doğan et al. [23] conducted an empirical study comparing nine well-known multiclass SVM variants and found that the Weston-Watkins SVM outperformed the others in both efficiency and accuracy. Typically, the Weston-Watkins linear SVM problem is addressed by considering its dual counterpart and devising algorithms for its minimization [26, 30, 52, 53, 54]. On the other hand, in this work we optimize a smooth version of the Weston-Watkins loss function directly, avoiding the dual formulation, which is particularly advantageous in big data scenarios. Primal approaches are also appealing as they guarantee a continuous decrease in the primal objective function, and can benefit from sound convergence guarantees even in the challenging non-convex setting.

Contributions. (i) We provide a matrix-based formulation of the primal smoothed Weston-Watkins problem, enabling an efficient computation of both the loss and its gradient from a practical perspective, while avoiding costly array indexing operations. *(ii)* Building on this matrix formulation, we first derive a batch MM algorithm for solving the regularized smoothed Weston-Watkins optimization problem, exploiting the Lipschitz-differentiability of the loss function to provide an appropriate tangent majorant inspired by half-quadratic schemes [2]. *(iii)* When the training set is large, computing the gradient of the smoothed Weston-Watkins loss becomes infeasible, making the direct application of the previous MM scheme impractical. To overcome this difficulty, we propose an incremental version of our MM method that combines computable gradient approximations with the benefits of second-order acceleration from the MM technique. In more detail, the incremental MM algorithm we propose is inspired by a classical incremental gradient scheme [9] where the descent direction is scaled by a symmetric and positive definite matrix based on the half-quadratic tangent majorant. The inverse of the scaling matrix results in the sum of a constant term that can be computed in a preprocessing phase -suitable for parallelization- and a diagonal term that remains fixed in each epoch. The product between the scaling matrix and the approximation of the gradient in the incremental iteration can be efficiently performed by Cholesky factorization. We establish convergence results for the proposed incremental MM algorithm under both convex and non-convex loss function assumptions. *(iv)* Several numerical experiments on multiclass classification problems show the effectiveness of the suggested approach in minimizing the smoothed Weston-Watkins loss function, when compared to both classical incremental and stochastic gradient methods. The second-order information leveraged by the half-quadratic majorant and the efficient computation of the scaled direction allow for faster convergence than applying first-order approaches. Furthermore, we show that, by applying kernel principal component analysis and foundation models as part of the preprocessing pipeline, the proposed incremental MM scheme for optimizing a linear multiclass SVM achieves performance comparable to that of state-of-the-art deep learning methods on benchmark tasks.

Related works. Primal optimizations of SVMs have already been studied in the case of binary classification by several authors [13, 29, 31, 34, 47]. To the best of our knowledge, the primal optimization of multiclass SVM has been investigated only in [59], where a logistic regression based loss is considered and optimized using a nonlinear conjugate gradient scheme. In this work, we design instead a highly scalable second-order method based on an MM strategy, specifically tailored to address a smooth Weston-Watkins problem, allowing for fast minimization of differentiable losses, and efficiently handling large training sets.

MM approaches have been successfully applied to several problems arising in machine learning (see, for example, [21, 24, 49, 55] and references therein), offering convergence guarantees even in the presence of non-convex objective functions [38, 42]. Moreover, MM methods have been previously applied to binary SVMs, as explored in [8]. In particular, [8] employs an MM approach with a quadratic surrogate function to train a binary SVM-based linear model on small training sets. This work extends that approach to the multiclass setting and applies it to minimizing a Weston-Watkins-based loss function, also considering a big data framework.

MM algorithms combined with gradient approximation to deal with big data scenarios have been investigated in the literature. Specifically, the extension of MM methodology to the stochastic context has been studied recently in [16, 18, 32, 33, 40]. In [16] an MM-based scheme is proposed, which can be traced back to a preconditioned stochastic gradient algorithm, with a stochastic scaling matrix as well. A similar approach is also used in [18], but applied to a more restricted setting related to the minimization of a fidelity function based on a least squares criterion. The MISO algorithm presented in [32, 33] is based on variance reduction technique by keeping a memory of past gradient estimates for each data point. In a big data context, this approach can be highly costly or even impractical. In [40], stochastic variance-reduced MM algorithms were introduced, combining the general MM principle with the variance-reduction techniques used in SAGA [22], SVRG [28], and SARAH [36] stochastic gradient methods. However, it is well-known that the gradient estimators underlying SAGA, SVRG, and SARAH can be difficult to apply in machine learning and deep learning frameworks due to the high computational cost associated with handling large-scale datasets.
The MM approach presented in this paper has the advantage of being based on an incremental (i.e., deterministic) process rather than a stochastic one. Instead of generating mini-batches at each iteration by randomly selecting individual components of the objective function, a fixed partition of the dataset is created at the beginning of the process and remains unchanged throughout the entire optimization. This approach can lead to greater computational efficiency and reduced computational cost, by minimizing the overhead of I/O operations and memory access.

The incremental MM scheme proposed in this paper is motivated by the need to address a large-scale optimization problem arising from multiclass SVM. Over the past decade, a wide range of algorithms has been developed to handle large-scale optimization problems (see, for example, [7, 12, 15, 25, 39, 44, 45] and references therein). While our approach is specifically tailored to multiclass SVMs, it can, in principle, be applied to other minimization problems whose objective functions can be formulated as a regularized finite-sum.

The paper is organized as follows. In Section 0.2, we present the objective function used to train the SVM-based multiclass linear classifier. Section 0.3 introduces the batch MM approach. Building on an incremental gradient scheme, Section 0.4 introduces our main contribution, that is a highly scalable incremental version of the MM method described in Section 0.3, and convergence analysis of it. Numerical experiments highlighting the advantages of this algorithm are provided in Section 0.5. Section 0.6 focuses on dataset remapping techniques aimed at improving the performance of linear models. Finally, the conclusions are presented in Section 0.7.

0.2 Problem formulation

The problem addressed in this work is supervised multiclass classification. Let a training dataset $\mathcal{D} = \{(\mathbf{x}_k, \mathbf{y}_k)\}_{k\in\mathbb{K}}$ with $\mathbb{K} = \{1, \dots, K\}$ consist of pairs of feature vectors $\mathbf{x}_k \in \mathbb{R}^n$, and their corresponding labels $\mathbf{y}_k \in \mathbb{R}^Q$, with Q the number of classes. Assuming one-hot encoding for the Q classes, each $\mathbf{y}_k$ is an element of the canonical basis of $\mathbb{R}^Q$, namely $\mathbf{y}_k = \mathbf{e}_{q_k^*}$, where $q_k^* \in \{1, \dots, Q\}$ corresponds to the class associated with $\mathbf{x}_k$. Given the labeled dataset, the goal of multiclass classification is to assign a new observation $\mathbf{x} \in \mathbb{R}^n$ to the correct class among $\{1, \dots, Q\}$. To achieve this task, we formulate a penalized minimization problem whose objective function is the sum of a loss term and a regularization term. More details are provided in Sections 0.2.1 and 0.2.2.

0.2.1 A smooth Weston-Watkins loss function

A linear classifier for categorizing an observation $\mathbf{x}$ into Q classes can be defined as

$$\begin{aligned} M : \mathbb{R}^n &\to \mathbb{R}^Q \\ \mathbf{x} &\to M(\mathbf{x}) = \mathbf{e}_{\hat{q}} \end{aligned}$$

where

$$\hat{q} = \underset{q\in\{1,\dots,Q\}}{\operatorname{argmax}}\ (\mathbf{w}_q^\top \mathbf{x} + b_q), \tag{1}$$

and, for every $q \in \{1, \dots, Q\}$, $\mathbf{w}_q \in \mathbb{R}^n$ and $b_q \in \mathbb{R}$ define the separating hyperplane for class q. The value $\mathbf{w}_q^\top \mathbf{x} + b_q$ is called the *similarity score* for the q-th class. Hence, the predicted label $M(\mathbf{x})$ is the vector $\mathbf{e}_{\hat{q}}$ corresponding to the index $\hat{q}$ attaining the highest similarity score with $\mathbf{x}$. The classifier parameters $\mathbf{w}_q$ and b_q with $q \in \{1, \dots, Q\}$ in (1) can be estimated from the training dataset $\{(\mathbf{x}_k, \mathbf{y}_k)\}_{k\in\mathbb{K}}$ as described hereafter. The goal is to find $\mathbf{w}_q$ and b_q with $q \in \{1, \dots, Q\}$ such that the correct class is maximally separated from the others, while still minimizing the classification error on the training set. Given a sample $(\mathbf{x}_k, \mathbf{y}_k) \in \mathcal{D}$, an ideal loss function measuring classification correctness is the *misclassification error*, which can be defined as

$$\ell\left(M(\mathbf{x}_k), \mathbf{y}_k\right) = \sigma\left(\mathbf{w}_{q_k^*}^\top \mathbf{x}_k + b_{q_k^*} - \left(\max_{\substack{1\le q\le Q\\ q\neq q_k^*}} \mathbf{w}_q^\top \mathbf{x}_k + b_q\right)\right), \tag{2}$$

where, for every $v \in \mathbb{R}$, $\sigma(v) = \dfrac{1-\operatorname{sign}(v)}{2}$. We remark that the loss function in (2) is related to the definition of *margin* for a given sample $\mathbf{x}_k$, provided below.

Definition 0.1. *The margin for a given sample $\mathbf{x}_k$ is the difference between the similarity score of the correct class and the maximum similarity score of any incorrect class. More formally, the margin for a sample $\mathbf{x}_k$ is*

$$m(\mathbf{x}_k) = \mathbf{w}_{q_k^*}^\top \mathbf{x}_k + b_{q_k^*} - \left(\max_{\substack{1\le q\le Q\\ q\neq q_k^*}} \mathbf{w}_q^\top \mathbf{x}_k + b_q\right).$$

Obviously, if the margin $m(\mathbf{x}_k)$ is negative then the predicted label $M(\mathbf{x}_k)$ for $\mathbf{x}_k$ is wrong. On the other hand, a larger positive margin means the correct class is more distinct from the incorrect ones and the related misclassification error is zero. Matrix $\mathbf{W} = [\mathbf{w}_1, \dots, \mathbf{w}_Q]^\top \in \mathbb{R}^{Q\times n}$ and vector $\mathbf{b} = (b_q)_{1\le q\le Q}$ must be trained so as to minimize the misclassification error over all the K samples, hence solve the following optimization problem:

$$\underset{\mathbf{W}\in\mathbb{R}^{Q\times n},\, \mathbf{b}\in\mathbb{R}^Q}{\operatorname{minimize}} \quad \sum_{k=1}^{K} \sigma\left(\mathbf{w}_{q_k^*}^\top \mathbf{x}_k + b_{q_k^*} - \left(\max_{\substack{1\le q\le Q\\ q\neq q_k^*}} \mathbf{w}_q^\top \mathbf{x}_k + b_q\right)\right). \tag{3}$$

Unfortunately, the objective function in (3) is non-differentiable and non-convex. To partially overcome this issue, a popular choice consists in substituting σ with the hinge loss function $(\forall v \in \mathbb{R})\ \rho_{\text{hinge}}(v) = \max\{1 - v, 0\}$, which provides the minimal convex upper bound on the function σ. A common variant of the previous optimization problem considers the Weston-Watkins loss [54], which introduces a margin-based formulation by summing over all incorrect classes:

$$\underset{\mathbf{W}\in\mathbb{R}^{Q\times n},\, \mathbf{b}\in\mathbb{R}^Q}{\operatorname{minimize}} \quad \sum_{k=1}^{K} \sum_{\substack{q=1\\ q\neq q_k^*}}^{Q} \rho_{\text{hinge}}\left((\mathbf{w}_{q_k^*} - \mathbf{w}_q)^\top \mathbf{x}_k + b_{q_k^*} - b_q\right). \tag{4}$$

The hinge loss is convex but not differentiable, which can cause numerical issues around $v = 1$. The solution of (4) can be achieved using primal-dual methods [14], but these approaches are often computationally expensive and not very flexible (e.g., not straightforwardly accounting for non-convex penalties). To mitigate this, we adopt here a *smoothed* version of the Weston-Watkins problem, as follows

$$\underset{\mathbf{W}\in\mathbb{R}^{Q\times n},\,\mathbf{b}\in\mathbb{R}^{Q}}{\text{minimize}} \quad \sum_{k=1}^{K}\sum_{\substack{q=1\\ q\neq q_k^*}}^{Q} \rho\left((\mathbf{w}_{q_k^*}-\mathbf{w}_q)^\top \mathbf{x}_k + b_{q_k^*} - b_q\right), \tag{5}$$

where $\rho:\mathbb{R}\to\mathbb{R}$ is differentiable with a β-Lipschitz continuous derivative and penalizes negative arguments. Examples of such functions include the smooth hinge loss, squared hinge loss, sigmoid loss and logistic regression loss. Figure 1 shows possible smooth loss functions that can be used as approximations of the ρ_{hinge} loss. Finally, we remark that employing smoothed variants of the hinge loss enables the design of fast optimization algorithms based on second-order information, such as the MM approach proposed in this paper.

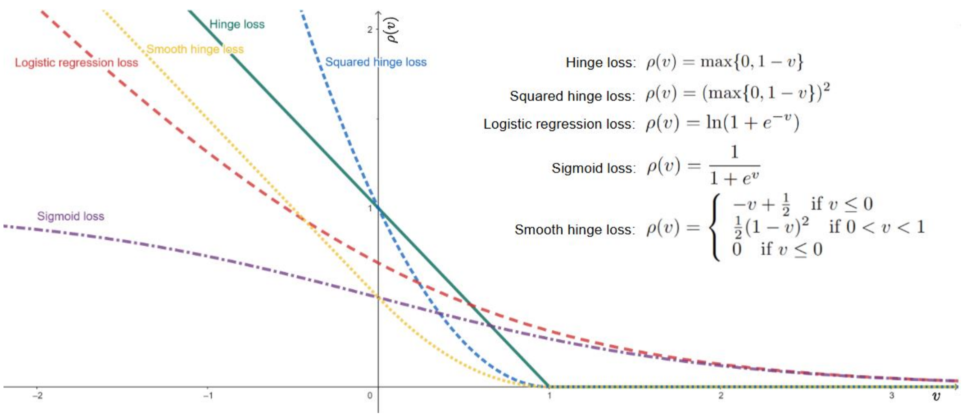


Figure 1: Smooth loss functions alternative to the hinge loss. The smooth hinge loss and squared hinge loss offer a smooth approximation around the non-differentiable point, while both the logistic regression and sigmoid losses are smooth, monotonic decreasing functions that asymptotically approach zero.

Remark 1. *The Weston-Watkins loss in* (4) *is commonly used in the multiclass extension of the SVM [26, 30, 52, 53, 54] to represent the primal problem. Unlike the approach followed in these papers, this work directly minimizes a regularized version of the smoothed Weston-Watkins loss in* (5)*, avoiding the transition to the dual problem. This offers an advantage, especially in big data scenarios, where solving the primal problem directly can lead to better scalability and computational efficiency compared to dual optimization methods, which often involve solving large-scale quadratic programs. In the case of binary SVM, some benefits of primal optimization have been highlighted in [13, 29] .*

0.2.2 Regularization

In order to mitigate overfitting, we consider a regularized version of problem (5). Specifically, in this paper, we focus on the following optimization problem:

$$\underset{\mathbf{W}\in\mathbb{R}^{Q\times n},\,\mathbf{b}\in\mathbb{R}^{Q}}{\text{minimize}} \quad \sum_{k=1}^{K}\sum_{\substack{q=1\\ q\neq q_k^*}}^{Q} \rho\left((\mathbf{w}_{q_k^*}-\mathbf{w}_q)^\top\mathbf{x}_k + b_{q_k^*} - b_q\right) + f(\mathbf{W}), \tag{6}$$

where the regularization term f is defined as

$$\left(\forall\,\mathbf{W}=(w_{i,j})_{\substack{1\le i\le Q\\ 1\le j\le n}}\in\mathbb{R}^{Q\times n}\right) \quad f(\mathbf{W}) = \lambda\sum_{i=1}^{Q}\sum_{j=1}^{n}\varphi(w_{i,j}) + \frac{\eta}{2}\|\mathbf{W}\|_{\mathrm{F}}^2. \tag{7}$$

Here $\|\cdot\|_{\mathrm{F}}$ denotes the Frobenius norm , and $\eta\ge 0$ and $\lambda \ge 0$ are regularization parameters. Function $\varphi:\mathbb{R}\longrightarrow\mathbb{R}$ is a potential function that satisfies the following conditions:

1. φ is a-Lipschitz differentiable on $\mathbb{R}$, with $a>0$;
2. φ is even;
3. $\varphi(\sqrt{\cdot})$ is concave on $]0,+\infty[$;
4. φ is increasing on $]0,+\infty[$.

These assumptions on the function φ are required to establish the convergence results for the MM method proposed in the paper. They allow for various choices of regularization terms, such as smooth approximations to the ℓ_1-norm or the ℓ_0-pseudo-norm [17, 18], that promote sparsity of the SVM model parameters. Sparsity is desirable in many practical applications, as it effectively performs implicit feature selection, retaining only the features that are most relevant to the task while discarding those that are redundant or introduce noise. Sparsity becomes particularly important in classification problems involving limited or highly unbalanced datasets, where models are prone to overfitting. In such cases, conventional (non-sparse) SVMs, often fitted closely to specific training data, may fail to generalize well to unseen samples.
For $\lambda = 0$ in (7), we recover the standard quadratic penalty commonly used in SVMs [51]. When both $\eta > 0$ and $\lambda > 0$, with φ acting as a sparse-promoting term, f can be viewed as an elastic-net penalty [51, 60].
Throughout the paper, we assume that either $\eta > 0$ or φ is coercive. As a result, the problem in (6) is well-posed, meaning that a solution exists.

0.2.3 Matrix formulation

In this section we rewrite problem (6) in a convenient matrix form which allows for efficient computaton of the objective function and its gradient. In matrix- and tensor-based languages such as MATLAB and Python, adopting a fully matrix-based reformulation is advantageous, as it eliminates both logical constructs and array indexing that could slow down computation.

We begin by rewriting the differentiable version of the Weston-Watkins loss in (5), hereafter denoted by $\mathcal{L}_{WW}$:

$$\begin{aligned}
\mathcal{L}_{WW}(\mathbf{W},\mathbf{b}) &= \sum_{k=1}^{K} \sum_{\substack{q=1\\ q\neq q_k^*}}^{Q} \rho\left((\mathbf{w}_{q_k^*}-\mathbf{w}_q)^\top \mathbf{x}_k + b_{q_k^*} - b_q\right) \\
&= \sum_{k=1}^{K} \sum_{q=1}^{Q} \rho\left((\mathbf{w}_{q_k^*}-\mathbf{w}_q)^\top \mathbf{x}_k + b_{q_k^*} - b_q\right) \\
&\quad - \sum_{k=1}^{K} \rho\left((\mathbf{w}_{q_k^*}-\mathbf{w}_{q_k^*})^\top \mathbf{x}_k + b_{q_k^*} - b_{q_k^*}\right) \\
&= \sum_{k=1}^{K} \sum_{q=1}^{Q} \rho\left((\mathbf{w}_{q_k^*}-\mathbf{w}_q)^\top \mathbf{x}_k + b_{q_k^*} - b_q\right) - K\rho(0) \\
&= \sum_{k=1}^{K} \sum_{q=1}^{Q} \rho\left((\mathbf{e}_{q_k^*}-\mathbf{e}_q)^\top (\mathbf{W}\mathbf{x}_k + \mathbf{b})\right) - K\rho(0).
\end{aligned} \tag{8}$$

Given the matrix

$$\Theta = [\mathbf{W},\mathbf{b}] = \begin{bmatrix} \mathbf{w}_1^\top & b_1 \\ \mathbf{w}_2^\top & b_2 \\ \vdots & \\ \mathbf{w}_Q^\top & b_Q \end{bmatrix} \in \mathbb{R}^{Q\times(n+1)}$$

and the vector $\tilde{\mathbf{x}}_k = \begin{bmatrix} \mathbf{x}_k \\ 1 \end{bmatrix} \in \mathbb{R}^{n+1}$, it follows from (8) that

$$\begin{aligned}
\mathcal{L}_{WW}(\Theta) &= \sum_{k=1}^{K}\sum_{q=1}^{Q} \rho\left((\mathbf{e}_{q_k^*} - \mathbf{e}_q)^\top \Theta \tilde{\mathbf{x}}_k\right) - K\rho(0) \\
&= \sum_{k=1}^{K}\sum_{q=1}^{Q} \rho\left((\mathbf{e}_{q_k^*} - \mathbf{e}_q)^\top (\tilde{\mathbf{x}}_k^\top \otimes \mathbf{I}_Q)\,\mathrm{vec}(\Theta)\right) - K\rho(0) \\
&= \sum_{k=1}^{K}\sum_{q=1}^{Q} \rho\left((\mathbf{e}_{q_k^*} - \mathbf{e}_q)^\top [(\tilde{\mathbf{x}}_k)_1\mathbf{I}_Q \;\; (\tilde{\mathbf{x}}_k)_2\mathbf{I}_Q \ldots (\tilde{\mathbf{x}}_k)_{n+1}\mathbf{I}_Q]\,\mathrm{vec}(\Theta)\right) \\
&\quad - K\rho(0) \\
&= \sum_{k=1}^{K}\sum_{q=1}^{Q} \rho\left([(\tilde{\mathbf{x}}_k)_1(\mathbf{e}_{q_k^*} - \mathbf{e}_q)^\top \;\ldots\; (\tilde{\mathbf{x}}_k)_{n+1}(\mathbf{e}_{q_k^*} - \mathbf{e}_q)^\top]\,\mathrm{vec}(\Theta)\right) \\
&\quad - K\rho(0),
\end{aligned} \tag{9}$$

where the second equality follows from a well-known property of the Kronecker product[1], vec denotes the vectorization operation by columns, $\mathbf{I}_Q$ is the identity matrix in $\mathbb{R}^{Q\times Q}$ and $(\tilde{\mathbf{x}}_k)_j$ represents the j-th component of $\tilde{\mathbf{x}}_k$. Additionally, to reformulate (9) for each sample index $k \in \mathbb{K}$, we introduce the matrix $\hat{\mathbf{Y}}_k \in \mathbb{R}^{Q\times Q}$ defined as

$$\hat{\mathbf{Y}}_k = \begin{bmatrix} \mathbf{e}_{q_k^*}^\top \\ \vdots \\ \mathbf{e}_{q_k^*}^\top \end{bmatrix} - \begin{bmatrix} \mathbf{e}_1^\top \\ \vdots \\ \mathbf{e}_Q^\top \end{bmatrix} = \begin{bmatrix} \mathbf{y}_k^\top \\ \vdots \\ \mathbf{y}_k^\top \end{bmatrix} - \mathbf{I}_Q$$

and the matrix $\mathbf{L}_k \in \mathbb{R}^{Q\times Q(n+1)}$ such that:

$$\mathbf{L}_k = \tilde{\mathbf{x}}_k^\top \otimes \hat{\mathbf{Y}}_k = [(\tilde{\mathbf{x}}_k)_1\hat{\mathbf{Y}}_k \;\; (\tilde{\mathbf{x}}_k)_2\hat{\mathbf{Y}}_k \;\ldots\; (\tilde{\mathbf{x}}_k)_{n+1}\hat{\mathbf{Y}}_k].$$

It is clear that the q-th row of $\mathbf{L}_k$ is equal to the matrix

$$[(\tilde{\mathbf{x}}_k)_1(\mathbf{e}_{q_k^*} - \mathbf{e}_q)^\top \;\ldots\; (\tilde{\mathbf{x}}_k)_{n+1}(\mathbf{e}_{q_k^*} - \mathbf{e}_q)^\top]$$

appearing in (9). In view of all these considerations and by setting $\boldsymbol{\theta} = \mathrm{vec}(\Theta) \in \mathbb{R}^{Q(n+1)}$, the Weston-Watkins loss can be written as

$$\mathcal{L}_{WW}(\boldsymbol{\theta}) = \sum_{k=1}^{K}\sum_{q=1}^{Q} \rho\left((\mathbf{L}_k)_{\{q,:\}}\boldsymbol{\theta}\right) - K\rho(0),$$

where $(\mathbf{L}_k)_{\{q,:\}}$ is the q-th row of the matrix $\mathbf{L}_k$. In order to finalize the matrix reformulation of (6), we introduce $\mathbf{L} \in \mathbb{R}^{Q\times Q(n+1)\times K}$ and $\mathbf{L}^\top \in \mathbb{R}^{Q(n+1)\times Q\times K}$,

[1]Given $L \in \mathbb{R}^{a\times b}$, $M \in \mathbb{R}^{b\times c}$, $N \in \mathbb{R}^{c\times d}$, it holds that $vec(LMN) = (N^\top \otimes L)\,\mathrm{vec}(M)$.

which are, respectively, the tensors defined as the collection of all K matrices $\mathbf{L}_k$ and $\mathbf{L}_k^\top$. It can be noticed that $\mathbf{L}\boldsymbol{\theta}$ is a matrix in $\mathbb{R}^{Q\times K}$ whose element at row q and column k, denoted by $(\mathbf{L}\boldsymbol{\theta})_{q,k}$, is given by

$$(\mathbf{L}\boldsymbol{\theta})_{q,k} = (\mathbf{L}_k)_{\{q,:\}}\boldsymbol{\theta} = \sum_{t=1}^{Q(n+1)} (\mathbf{L}_k)_{q,t}\theta_t, \quad q \in \{1,\dots,Q\},\ k \in \{1,\dots,K\}. \tag{10}$$

Hence, we get

$$\mathcal{L}_{WW}(\boldsymbol{\theta}) = \sum_{k=1}^{K}\sum_{q=1}^{Q} \rho\left((\mathbf{L}\boldsymbol{\theta})_{q,k}\right) - K\rho(0).$$

From a practical perspective, once $\mathbf{L}\boldsymbol{\theta}$ is computed and assuming that ρ operates element-wise on matrices, the objective function can be evaluated directly by applying function ρ to the entire matrix $\mathbf{L}\boldsymbol{\theta}$, without explicitly extracting its elements. The value of the Weston-Watkins loss is finally given by summing over all rows and columns of $\rho(\mathbf{L}\boldsymbol{\theta})$.
We now aim to rewrite the regularizer in (6) in terms of $\boldsymbol{\theta}$. To do so, we decompose $\boldsymbol{\theta}$ as follows:

$$\boldsymbol{\theta} = \begin{bmatrix}\boldsymbol{\theta}_w \\ \boldsymbol{\theta}_b\end{bmatrix}$$

where $\boldsymbol{\theta}_w \in \mathbb{R}^{Qn}$ contains the first Qn components of $\boldsymbol{\theta}$, and $\boldsymbol{\theta}_b \in \mathbb{R}^{Q}$ contains the remaining components, indexed from $Qn+1$ to $Q(n+1)$. With a slight abuse of notation, we can rewrite the regularization term f in (6) as

$$\left(\forall \boldsymbol{\theta} = ((\boldsymbol{\theta}_w)_i)_{1\le i\le Qn} \in \mathbb{R}^{Qn}\right) \quad f(\boldsymbol{\theta}) = \lambda\sum_{i=1}^{Qn} \varphi\left((\boldsymbol{\theta}_w)_i\right) + \frac{\eta}{2}\|\boldsymbol{\theta}_w\|_2^2. \tag{11}$$

The resulting reformulation of the problem (6) is

$$\underset{\boldsymbol{\theta}\in\mathbb{R}^{Q(n+1)}}{\text{minimize}} \quad \sum_{k=1}^{K}\sum_{q=1}^{Q} \rho\left((\mathbf{L}\boldsymbol{\theta})_{q,k}\right) - K\rho(0) + f(\boldsymbol{\theta}) \equiv \mathcal{L}_{WW}(\boldsymbol{\theta}) + f(\boldsymbol{\theta}) \equiv \Phi(\boldsymbol{\theta}). \tag{12}$$

To make this expression more concise, we can introduce the function

$$\begin{aligned} g_K : \mathbb{R}^{Q\times K} &\longrightarrow \mathbb{R} \\ \mathbf{X} &\longrightarrow g_K(\mathbf{X}) = \sum_{k=1}^{K}\sum_{q=1}^{Q} \rho\left((\mathbf{X})_{q,k}\right) \end{aligned}$$

where $(\mathbf{X})_{q,k}$ denotes the element of the q-th row and the k-th column of matrix $\mathbf{X}$. In view of the definition of g_K, problem (12) boils down to

$$\underset{\boldsymbol{\theta}\in\mathbb{R}^{Q(n+1)}}{\text{minimize}} \quad g_K(\mathbf{L}\boldsymbol{\theta}) - K\rho(0) + f(\boldsymbol{\theta}) \equiv \Phi(\boldsymbol{\theta}). \tag{13}$$

We conclude this section by deriving the expression of the gradient of Φ. First we clarify the definition of $\nabla\mathcal{L}_{WW}(\boldsymbol{\theta}) \in \mathbb{R}^{Q(n+1)}$. The j-th element of $\nabla\mathcal{L}_{WW}(\boldsymbol{\theta})$ with $j \in \{1,\ldots,Q(n+1)\}$ is

$$\begin{aligned}(\nabla\mathcal{L}_{WW}(\boldsymbol{\theta}))_j = \frac{\partial\mathcal{L}_{WW}(\boldsymbol{\theta})}{\partial\theta_j} &= \sum_{k=1}^{K}\sum_{q=1}^{Q}\rho'\left((\mathbf{L}\boldsymbol{\theta})_{q,k}\right)(\mathbf{L}_k)_{q,j}\\ &= \sum_{k=1}^{K}\left(\mathbf{L}_k^\top\right)_{\{j,:\}}\begin{bmatrix}\rho'\left((\mathbf{L}\boldsymbol{\theta})_{1,k}\right)\\ \vdots\\ \rho'\left((\mathbf{L}\boldsymbol{\theta})_{Q,k}\right)\end{bmatrix},\end{aligned} \tag{14}$$

where $\rho' : \mathbb{R} \longrightarrow \mathbb{R}$ is the derivative of ρ. In order to simplify the expression of the gradient of the function $\mathcal{L}_{WW}$, will make use of function g_K. Its gradient $\nabla g_K(\mathbf{X})$ at $\mathbf{X}$ is a matrix in $\mathbb{R}^{Q\times K}$ whose (q,k)-th element is

$$(\nabla g_K(\mathbf{X}))_{q,k} = \frac{\partial g_K(\mathbf{X})}{\partial(\mathbf{X})_{q,k}} = \rho'\left((\mathbf{X})_{q,k}\right), \quad q \in \{1,\ldots,Q\},\ k \in \{1,\ldots,K\}. \tag{15}$$

Consequently, (14) can be reformulated as

$$(\nabla\mathcal{L}_{WW}(\boldsymbol{\theta}))_j = \sum_{k=1}^{K}(\mathbf{L}_k^\top)_{\{j,:\}}(\nabla g_K(\mathbf{L}\boldsymbol{\theta}))_{\{:,k\}}$$

where $(\nabla g_K(\mathbf{L}\boldsymbol{\theta}))_{\{:,k\}}$ denotes the k-th column of $\nabla g_K(\mathbf{L}\boldsymbol{\theta})$. We remark that all components of $\nabla\mathcal{L}_{WW}$ can be computed simultaneously by properly applying a *tensor product* function either in MATLAB or Python to $\mathbf{L}^\top$ and $\nabla g_K(\mathbf{L}\boldsymbol{\theta})$, eliminating the need to extract specific rows or columns from either matrix. In particular, we have

$$\nabla\mathcal{L}_{WW}(\boldsymbol{\theta}) = \mathbf{L}^\top\nabla g_K(\mathbf{L}\boldsymbol{\theta}) := \begin{bmatrix}\displaystyle\sum_{k=1}^{K}(\mathbf{L}_k^\top)_{\{1,:\}}(\nabla g_K(\mathbf{L}\boldsymbol{\theta}))_{\{:,k\}}\\ \displaystyle\sum_{k=1}^{K}(\mathbf{L}_k^\top)_{\{2,:\}}(\nabla g_K(\mathbf{L}\boldsymbol{\theta}))_{\{:,k\}}\\ \vdots\\ \displaystyle\sum_{k=1}^{K}(\mathbf{L}_k^\top)_{\{Q(n+1),:\}}(\nabla g_K(\mathbf{L}\boldsymbol{\theta}))_{\{:,k\}}\end{bmatrix}. \tag{16}$$

On the other hand, the gradient $\nabla f(\boldsymbol{\theta})$ of the regularization term is a vector in

$\mathbb{R}^{Q(n+1)}$, expressed as

$$\nabla f(\boldsymbol{\theta}) = \begin{pmatrix} \lambda\varphi'(\theta_1) + \eta\theta_1 \\ \vdots \\ \lambda\varphi'(\theta_Q) + \eta\theta_Q \\ \vdots \\ \lambda\varphi'(\theta_{Q\cdot n-n+1}) + \eta\theta_{Q\cdot n-n+1} \\ \vdots \\ \lambda\varphi'(\theta_{Q\cdot n}) + \eta\theta_{Q\cdot n} \\ 0 \\ \vdots \\ 0 \end{pmatrix},$$

where φ' is the first derivative of the potential function φ involved in the construction of the regularization term f.

In summary, the expression of the gradient of Φ is given by

$$\left(\forall \boldsymbol{\theta} \in \mathbb{R}^{Q(n+1)}\right) \quad \nabla\Phi(\boldsymbol{\theta}) = \mathbf{L}^\top \nabla g_K(\mathbf{L}\boldsymbol{\theta}) + \nabla f(\boldsymbol{\theta}). \tag{17}$$

0.3 Batch MM approach

The MM approach is an iterative optimization technique that aims at transforming a challenging optimization problem into a sequence of easier subproblems by iteratively constructing and minimizing a surrogate function that majorizes the original objective function. At each iteration, the surrogate function, also referred to as the *tangent majorant*, is designed to be a convenient approximation to the objective function and exhibits good properties such as convexity. Under suitable assumptions, the MM algorithm converges to a local minimizer of the original function while often improving computational efficiency compared to classical gradient methods. For a comprehensive introduction to MM schemes, the reader is referred to [50, 58]. This section is devoted to providing the key ingredients needed to apply an MM method to the resolution of (12) by processing the training set in a single full batch.

Definition 0.2. *Given problem* (12)*, a tangent majorant* $h(\cdot;\boldsymbol{\theta}') : \mathbb{R}^{Q(n+1)} \to \mathbb{R}$ *of* Φ *at* $\boldsymbol{\theta}' \in \mathbb{R}^{Q(n+1)}$ *is a function such that*

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

Building on the previous definition of tangent majorant, the general MM iterative scheme for (12) reads as

$$(\forall t \in \mathbb{N}) \quad \boldsymbol{\theta}^{(t+1)} = \underset{\boldsymbol{\theta}\in\mathbb{R}^{Q(n+1)}}{\operatorname{argmin}} \; h(\boldsymbol{\theta};\boldsymbol{\theta}^{(t)}), \tag{18}$$

given $\boldsymbol{\theta}^{(0)} \in \mathbb{R}^{Q(n+1)}$. Under suitable hypotheses on the objective function in (12) and its majorizing approximations, the iterative scheme (18) generates a sequence that converges to a solution to (12) [27]. We now discuss the construction of reliable majorizing approximations for the function Φ. First, we introduce the so called *descent lemma majorant*, which is based on the Lipschitz smoothness property of Φ.

Proposition 0.1. *Consider the function* Φ *involved in* (12). Φ *is* μ*-Lipschitz differentiable on* $\mathbb{R}^{Q(n+1)}$ *with*

$$\mu = \beta \|\mathbf{L}\|^2 + a\lambda + \eta, \tag{19}$$

where $\|\mathbf{L}\|$ *is the spectral norm of* $\mathbf{L}$*. As a consequence, for every* $\boldsymbol{\theta}' \in \mathbb{R}^{Q(n+1)}$*, the following function is a tangent majorant of* Φ *at* $\boldsymbol{\theta}'$*,*

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

While the previous majorizing approximation is useful, its accuracy may be limited since its curvature remains independent of the tangency point $\boldsymbol{\theta}'$. To address this limitation, we introduce a more refined approximation inspired by the half-quadratic techniques commonly used in image processing algorithms [2]. Before stating the proposition, we recall the definition of the tensor product $\mathbf{L}^\top\mathbf{L} \in \mathbb{R}^{Q(n+1)\times Q(n+1)}$. The (i, j) entry of $\mathbf{L}^\top\mathbf{L}$ is given by

$$\left(\mathbf{L}^\top\mathbf{L}\right)_{i,j} = \sum_{k=1}^{K}\sum_{q=1}^{Q} \left(\mathbf{L}_k^\top\right)_{i,q} (\mathbf{L}_k)_{q,j} = \sum_{k=1}^{K} \left(\mathbf{L}_k^\top\right)_{\{i,:\}} (\mathbf{L}_k)_{\{:,j\}} \tag{20}$$

where $(i, j) \in \{1, \dots, Q(n+1)\}^2$. Here, we recall that $\left(\mathbf{L}_k^\top\right)_{\{i,:\}}$ and $(\mathbf{L}_k)_{\{:,j\}}$ denote the i-th row of $\mathbf{L}_k^\top$ and the j-th column of $\mathbf{L}_k$, respectively. Their product in the last equality is understood as the standard row-by-column multiplication.

Proposition 0.2. *For every* $\boldsymbol{\theta}' \in \mathbb{R}^{Q(n+1)}$*, the following function is a tangent majorant of function* Φ *in* (12)*:*

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

where, for every $\boldsymbol{\theta} \in \mathbb{R}^{Q(n+1)}$,

$$\mathbf{A}(\boldsymbol{\theta}) = \beta \mathbf{L}^\top \mathbf{L} + \mathbf{R}(\boldsymbol{\theta}) = \beta \mathbf{L}^\top \mathbf{L} + \mathrm{Diag}\left(\begin{bmatrix} \lambda\psi(\theta_1) + \eta \\ \vdots \\ \lambda\psi(\theta_n) + \eta \\ \vdots \\ \lambda\psi(\theta_{(Q-1)\cdot n+1}) + \eta \\ \vdots \\ \lambda\psi(\theta_{Q\cdot n}) + \eta \\ \varepsilon \\ \vdots \\ \varepsilon \end{bmatrix} \right). \tag{21}$$

Here, $\psi : v \mapsto \varphi'(v)/v$ *(with* $\psi(0) := 0$*),* $\varepsilon > 0$ *and* β *denotes the Lipschitz constant of* ρ'.

The proof of Proposition 0.2 relies on standard results from the MM approach. For completeness, it is included in Appendix .1.

Based on Proposition 0.2, the MM method in (18) can be specialized as

$$\begin{aligned} (\forall t \in \mathbb{N}) \quad \boldsymbol{\theta}^{(t+1)} &= \underset{\boldsymbol{\theta}\in\mathbb{R}^{Q(n+1)}}{\operatorname{argmin}} \left(\nabla\Phi(\boldsymbol{\theta}^{(t)})^\top (\boldsymbol{\theta} - \boldsymbol{\theta}^{(t)}) + \frac{1}{2}(\boldsymbol{\theta} - \boldsymbol{\theta}^{(t)})^\top \mathbf{A}(\boldsymbol{\theta}^{(t)})(\boldsymbol{\theta} - \boldsymbol{\theta}^{(t)}) \right) \\ &= \boldsymbol{\theta}^{(t)} - (\mathbf{A}(\boldsymbol{\theta}^{(t)}))^{-1} \nabla\Phi(\boldsymbol{\theta}^{(t)}). \end{aligned} \tag{22}$$

The MM approach (22) relies on the positive definiteness of the matrix $\mathbf{A}(\boldsymbol{\theta})$ defined in Proposition 0.2. Consequently, we assume that $\eta > 0$ hereafter. The iterative scheme (22) can be interpreted both as a scaled gradient algorithm with a unit step-size, and as a special case of the subspace acceleration method introduced in [17], for which a convergence proof was provided. Indeed, the following theoretical convergence result for the MM method (22) can be established assuming that the objective function satisfies Kurdyka-Lojasiewicz inequality [5].

Theorem 0.1. *Assume that* Φ *in* (12) *satisfies the Kurdyka-Lojasiewicz inequality. Let* $(\boldsymbol{\theta}^{(t)})_{t\in\mathbb{N}}$ *be generated by* (22)*. Then,* $(\boldsymbol{\theta}^{(t)})_{t\in\mathbb{N}}$ *converges to a stationary point of* Φ *in* (12)*. Moreover, if* Φ *is strongly convex,* $(\boldsymbol{\theta}^{(t)})_{t\in\mathbb{N}}$ *converges to the unique minimizer of* Φ.

Proof. In view of Proposition 0.1, function Φ in (12) is Lipschitz differentiable. As a consequence, the theorem directly follows from [17, Theorem 3], using Proposition 0.2, and taking into account that (22) is a particular case of the MM quadratic subspace algorithm proposed in [17] with one inner iteration, and full space mapping. □

Examples of Kurdyka-Lojasiewicz functions are the indicator functions of semi-algebraic sets, real polynomials, p-norms and, in general, semi-algebraic functions or real analytic functions [4, 10, 56]. If the regularization function φ in (7) is chosen as the hyperbolic potential or the Welsh potential, as done in our numerical experiments, then the objective function Φ satisfies the Kurdyka-Lojasiewicz property, hence Theorem 0.1 holds.

0.4 Incremental MM approach

Despite its simplicity and guaranteed convergence properties, the MM algorithm introduced in (22) for solving problem (12) may not be feasible when the number of examples K is very large. Indeed, if number of data is too large, tensor $\mathbf{L} \in \mathbb{R}^{Q\times Q(n+1)\times K}$ may become too large to be stored. As a consequence, in view of (16), the full gradient of Φ cannot be computed and the MM scheme (22) cannot be directly applied in this form. In this section we propose an *incremental* version of the MM scheme defined in (22), which can be successfully employed to solve problem (12) when K is large.

To design this incremental MM approach, we first rewrite the target function Φ in (6) as a sum of m functions, where m is a predefined number of blocks into which the training dataset $\mathcal{D}$ is partitioned. Specifically, we assume that

$$\mathcal{D} = \bigcup_{i=1}^{m} \mathcal{D}_i, \tag{23}$$

where, without loss of generality, each $\mathcal{D}_i$ with $i \in \{1, \dots, m\}$ contains $N = K/m$ examples, with N being a positive integer. Each block indexed by $i \in \{1, \dots, m\}$ can be interpreted as a minibatch defined as follows:

$$\mathcal{D}_i = \{(\mathbf{x}_k, \mathbf{y}_k)\}_{k\in\mathbb{K}_i} \tag{24}$$

with $\mathbb{K}_i = \{(i-1)\,N + 1, \dots, i\,N\}$. It follows from (6) that

$$(\forall \boldsymbol{\theta} \in \mathbb{R}^{Q(n+1)}) \quad \Phi(\boldsymbol{\theta}) = \sum_{i=1}^{m} \Phi_i(\boldsymbol{\theta}), \tag{25}$$

where, for every $i \in \{1, \dots, m\}$,

$$(\forall \boldsymbol{\theta} \in \mathbb{R}^{Q(n+1)}) \quad \Phi_i(\boldsymbol{\theta}) = \sum_{k\in\mathbb{K}_i} \sum_{\substack{q=1\\ q\neq q_k^*}}^{Q} \rho\big((\mathbf{w}_{q_k^*} - \mathbf{w}_q)^\top \mathbf{x}_k + b_{q_k^*} - b_q\big) + \frac{f(\boldsymbol{\theta})}{m}. \tag{26}$$

We now aim to rewrite functions $(\Phi_i)_{1\le i\le m}$ similarly to the form of Φ introduced in Section 0.2.3. First we define the sub-tensors of $\mathbf{L}$ corresponding to the blocks $(\mathcal{D}_i)_{1\le i\le m}$.

Definition 0.3. *For every $i \in \{1, \dots, m\}$, the tensor $\mathbf{L}^{(i)} \in \mathbb{R}^{Q\times Q(n+1)\times N}$, related to $\mathcal{D}_i$, is defined as the collection of matrices $(\mathbf{L}_k)_{k\in\mathbb{K}_i}$.*

Following the same arguments as in Section 0.2.3, each functions Φ_i with $i \in \{1, \dots, m\}$, and its gradient are expressed as

$$(\forall \boldsymbol{\theta} \in \mathbb{R}^{Q(n+1)}) \quad \Phi_i(\boldsymbol{\theta}) = g_N(\mathbf{L}^{(i)}\boldsymbol{\theta}) - N\rho(0) + \frac{f(\boldsymbol{\theta})}{m} \tag{27}$$

and

$$(\forall \boldsymbol{\theta} \in \mathbb{R}^{Q(n+1)}) \quad \nabla\Phi_i(\boldsymbol{\theta}) = \mathbf{L}^{(i)^\top}\nabla g_N(\mathbf{L}^{(i)}\boldsymbol{\theta}) + \frac{\nabla f(\boldsymbol{\theta})}{m}, \tag{28}$$

where $\mathbf{L}^{(i)}\boldsymbol{\theta}$ is defined as in (10), and the gradient $\nabla g_N(\mathbf{X})$ at $\mathbf{X}$ of function g_N is a matrix in $\mathbb{R}^{Q\times N}$ such that its (q, ν)-th element is defined as

$$(\nabla g_N(\mathbf{X}))_{q,\nu} = \rho'\left((\mathbf{X})_{q,\nu}\right), \quad q \in \{1, \dots, Q\},\ \nu \in \{1, \dots, N\}.$$

Formulation (25) allows us to process each block incrementally to solve problem (12). In particular, we propose to combine a standard incremental technique [9] with the half-quadratic MM approach, which led to the scheme (22). The resulting algorithm is an incremental gradient method where the gradient direction is properly scaled using matrix $\mathbf{A}(\cdot)$ defined in (21).

Remark 2. *According to* (21)*, matrix* $\mathbf{A}(\cdot)$ *also depends on the tensor* $\mathbf{L}$*. However, as seen from the definition in* (20)*, the computation of* $\mathbf{L}^\top\mathbf{L}$ *can be performed efficiently by accumulating each product* $\mathbf{L}_k^\top\mathbf{L}_k$*, eliminating the need to store the entire tensor* $\mathbf{L}$*. Moreover, since the term* $\mathbf{L}^\top\mathbf{L}$ *remains constant throughout the optimization process, it can be computed as a pre-processing step, as detailed in Section 0.4.1. Finally, we note that computing* $\mathbf{L}^\top\mathbf{L}$ *via* (20) *is both parallelizable and adaptable to potential additions to the training dataset, as arises in continual learning.*

The main steps of the proposed approach are detailed in Algorithm 1. This algorithm differs from a standard incremental gradient method applied to problem (12) through the scaling matrix $\mathbf{A}_t = \mathbf{A}(\boldsymbol{\theta}^{(t)}) = \beta\mathbf{L}^\top\mathbf{L} + \mathbf{R}(\boldsymbol{\theta}^{(t)})$, which is given by (21). This matrix is computed at the beginning of the t-th epoch and it is employed for the update of the successive m inner iterations. In particular, matrix $\mathbf{A}_t$, as well as the step-size γ_t, do not depend on the blocks $(\mathcal{D}_i)_{1\le i\le m}$ or the iterates $(\boldsymbol{\omega}^{(t,i-1)})_{1\le i\le m+1}$. We remark that the conditions on the sequence $(\gamma_t)_{t\in\mathbb{N}}$, stated at the beginning of Algorithm 1, are quite standard for incremental gradient methods and are needed for the convergence results, as detailed in Section 0.4.2.

Remark 3. *The update of* $\boldsymbol{\omega}^{(t,i)}$ *in* (29) *would, in principle, require the inversion of the full matrix* $\mathbf{A}_t$*. In practice, we avoid explicitly inverting* $\mathbf{A}_t$ *by computing* $\mathbf{A}_t^{-1}\nabla\Phi_i(\boldsymbol{\omega}^{(t,i-1)})$ *through the solution of two linear systems. Particularly, we exploit the fact that* $\mathbf{A}_t$ *is symmetric and positive definite. Based on this property, at the beginning of each epoch, we compute an upper triangular matrix* $\mathbf{P}_t \in \mathbb{R}^{Q(n+1)\times Q(n+1)}$ *via the Cholesky decomposition such that*

Algorithm 1 Incremental MM algorithm

```
Set $\boldsymbol{\theta}^{(0)} \in \mathbb{R}^{Q(n+1)}$, $\{\gamma_t\}_{t\in\mathbb{N}} \subset \mathbb{R}_+$ such that $\sum_{t=0}^{\infty} \gamma_t = +\infty$ and $\sum_{t=0}^{\infty} \gamma_t^2 < +\infty$,
$m > 0$, $\beta > 0$, and compute $\mathbf{L}^\top\mathbf{L}$.
for $t = 0, 1, \dots$ do
Set $\boldsymbol{\omega}^{(t,0)} = \boldsymbol{\theta}^{(t)}$
Compute $\mathbf{A}_t = \mathbf{A}(\boldsymbol{\theta}^{(t)}) = \beta\mathbf{L}^\top\mathbf{L} + \mathbf{R}(\boldsymbol{\theta}^{(t)})$ as defined in (21)
    for $i = 1, \dots, m$ do
        $\boldsymbol{\omega}^{(t,i)} = \boldsymbol{\omega}^{(t,i-1)} - \gamma_t \mathbf{A}_t^{-1} \nabla\Phi_i(\boldsymbol{\omega}^{(t,i-1)})$    (29)
    end for
Set
        $\boldsymbol{\theta}^{(t+1)} = \boldsymbol{\omega}^{(t,m)}$    (30)
end for
```

$\mathbf{A}_t = \mathbf{P}_t^\top\mathbf{P}_t$. *Subsequently, in the incremental update step* (29)*, the descent direction* $\mathbf{d}^{(t,i)} = \mathbf{A}_t^{-1}\nabla\Phi_i(\boldsymbol{\omega}^{(t,i-1)})$ *is obtained by solving the following two linear systems:*

$$\begin{cases} \mathbf{P}_t^\top \mathbf{z} = \nabla\Phi_i(\boldsymbol{\omega}^{(t,i-1)}) \\ \mathbf{P}_t \mathbf{d}^{(t,i)} = \mathbf{z} \end{cases}$$

with $\mathbf{z} \in \mathbb{R}^{Q(n+1)}$*. The solution of these linear systems is highly efficient because of the structure of matrix* $\mathbf{P}_t$*.*

We conclude the presentation of Algorithm 1 by noting that it can be regarded as an incremental version of the MM method detailed in (22). An epoch of the incremental MM gradient method through the components Φ_i differs from the standard gradient iteration (22) in that the evaluation of $\nabla\Phi_i$ is performed at the corresponding current estimate $\boldsymbol{\omega}^{(t,i-1)}$ rather than at the estimate $\boldsymbol{\theta}^{(t)}$ available at the start of the epoch. This approach enables solving problem (12) efficiently, even when K is very large.

0.4.1 Warm-up phase

As already mentioned, Algorithm 1 relies on an efficient computation of $\mathbf{L}^\top\mathbf{L}$, whose definition depends on the available training data. As highlighted in Remark 2, the computation of this matrix can be carried out using (20), which allows for an incremental approach by accumulating the products $\mathbf{L}_k^\top\mathbf{L}_k$. The warm-up phase is outlined in Algorithm 2. According to (20), the output $\mathbf{H}_m$ of Algorithm 2 coincides with $\mathbf{L}^\top\mathbf{L}$.

Algorithm 2 Warm-up phase

```
Set $m > 0$ and $\mathbf{H}_0$ as the null matrix in $\mathbb{R}^{Q(n+1)\times Q(n+1)}$.
Split the training dataset into blocks $(\mathcal{D}_i)_{1\le i\le m}$, as defined in (23)-(24).
for $i = 1,\dots,m$ do
Compute the tensor $\mathbf{L}^{(i)}$ corresponding to the block $\mathcal{D}_i$ from Definition 0.3.
Compute $\mathbf{H}_i = (\mathbf{L}^{(i)})^\top \mathbf{L}^{(i)} + \mathbf{H}_{i-1}$
end for
```

0.4.2 Convergence results

In this section we provide a proof of convergence of the incremental MM method defined in Algorithm 1. To ensure greater generality, the convergence results are established for a broad class of incremental scaled gradient schemes, of which Algorithm 1 is a specific instance. Specifically, we consider the general differentiable finite-sum optimization problem of the form:

$$\underset{\boldsymbol{\theta}\in\mathbb{R}^d}{\text{minimize}} \;\; \Phi(\boldsymbol{\theta}) \equiv \sum_{i=1}^{m} \Phi_i(\boldsymbol{\theta}). \tag{31}$$

The theoretical results presented in this section pertain to Algorithm 3. Further assumptions regarding the objective function Φ and the sequence of scaling matrices $(\mathbf{A}_t)_{t\in\mathbb{N}}$ will be introduced in the relevant theorems.

Algorithm 3 General incremental scaled gradient algorithm

```
Set $\boldsymbol{\theta}^{(0)} \in \mathbb{R}^d$, $\{\gamma_t\}_{t\in\mathbb{N}} \subset \mathbb{R}_+$ such that $\sum_{t=0}^{\infty}\gamma_t = +\infty$ and $\sum_{t=0}^{\infty}\gamma_t^2 < +\infty$, $m > 0$.

for $t = 0, 1, \dots$ do
Set $\boldsymbol{\omega}^{(t,0)} = \boldsymbol{\theta}^{(t)}$
Fix a symmetric and positive definite matrix $\mathbf{A}_t$.
    for $i = 1,\dots,m$ do
        $\boldsymbol{\omega}^{(t,i)} = \boldsymbol{\omega}^{(t,i-1)} - \gamma_t \mathbf{A}_t^{-1}\nabla\Phi_i(\boldsymbol{\omega}^{(t,i-1)})$
    end for
Set $\boldsymbol{\theta}^{(t+1)} = \boldsymbol{\omega}^{(t,m)}$
end for
```

Theorem 0.2 extends the results in the seminal work [9], by accounting for the presence of a scaling matrix in the incremental gradient method update. Without any convexity assumption, we prove that the gradient sequence $(\nabla\Phi(\boldsymbol{\theta}^{(t)}))_{t\in\mathbb{N}}$

tends to zero. If, in addition, we assume that the function is strictly convex and that the iterates are bounded, it is possible to prove the convergence of the iterates $(\boldsymbol{\theta}^{(t)})_{t\in\mathbb{N}}$ through Theorem 0.3.

We firstly recall [9, Proposition 1].

Proposition 0.3. *Let μ be some positive constant and let $\Phi : \mathbb{R}^d \to \mathbb{R}$ be a μ-Lipschitz differentiable function. Let $(\gamma_t)_{t\in\mathbb{N}}$ be positive step-sizes and let $(\boldsymbol{\theta}^{(t)})_{t\in\mathbb{N}}$ be a sequence generated by the iterates:*

$$(\forall t \in \mathbb{N}) \quad \boldsymbol{\theta}^{(t+1)} = \boldsymbol{\theta}^{(t)} + \gamma_t(\mathbf{s}_t + \mathbf{w}_t), \tag{32}$$

where $(\mathbf{s}_t)_{t\in\mathbb{N}}$ are descent directions satisfying for some positive constants c_1, c_2,

$$(\forall t \in \mathbb{N}) \quad c_1\|\nabla\Phi(\boldsymbol{\theta}^{(t)})\|^2 \leq -\langle\nabla\Phi(\boldsymbol{\theta}^{(t)}), \mathbf{s}_t\rangle, \quad \|\mathbf{s}_t\| \leq c_2(1 + \|\nabla\Phi(\boldsymbol{\theta}^{(t)})\|), \tag{H1}$$

and $(\mathbf{w}_t)_{t\in\mathbb{N}}$ is an vector error sequence satisfying for some positive constants p, q,

$$(\forall t \in \mathbb{N}) \quad \|\mathbf{w}_t\| \leq \gamma_t(p + q\|\nabla\Phi(\boldsymbol{\theta}^{(t)})\|). \tag{H2}$$

In addition, we assume that $(\gamma_t)_{t\in\mathbb{N}}$ is such that

$$\sum_{t=0}^{\infty}\gamma_t = +\infty, \quad \sum_{t=0}^{\infty}\gamma_t^2 < +\infty, \tag{H3}$$

Then, either $\Phi(\boldsymbol{\theta}^{(t)}) \to -\infty$ or $\Phi(\boldsymbol{\theta}^{(t)})$ converges to a finite value. In the latter case, $\lim_{t\to\infty}\nabla\Phi(\boldsymbol{\theta}_t) = 0$. Furthermore, every cluster point of $(\boldsymbol{\theta}^{(t)})_{t\in\mathbb{N}}$ is a stationary point of Φ.

Now, we prove that, under suitable assumptions, the incremental scaled gradient Algorithm 3 belongs to the class of algorithms defined in Proposition 0.3. We then discuss the validity of these assumptions for the objective function in (25)-(26) and the incremental MM algorithm (Algorithm 1) under consideration (see Remark 4).

Our first convergence result for the general incremental scaled gradient algorithm is stated in Theorem 0.2. This results extends [9, Proposition 2] to Algorithm 3. In [9], the analysis is restricted to a constant scaling matrix.

Theorem 0.2. *Given the optimization problem* (31)*, let $(\boldsymbol{\theta}^{(t)})_{t\in\mathbb{N}}$ be a sequence generated by Algorithm 3. Let us make the following assumptions.*

(i) For all $i \in \{1, \ldots, m\}$, there exists $\mu_i > 0$ such that Φ_i is μ_i-Lipschitz differentiable.

(ii) There exist positive constants C and D, such that

$$(\forall t \in \mathbb{N})(\forall i \in \{1, \ldots, m\}) \quad \|\nabla\Phi_i(\omega^{(t,i)})\| \leq C + D\|\nabla\Phi(\omega^{(t,i)})\|.$$

(iii) There exists $\chi > 0$, such that, for every $t \in \mathbb{N}$, the eigenvalues of $\mathbf{A}_t$ belong to the interval $[\frac{1}{\chi}, \chi]$.

(iv) $(\gamma_t)_{t\in\mathbb{N}}$ is a sequence of positive step-sizes satisfying (H3).

Then, either $\Phi(\boldsymbol{\theta}^{(t)}) \to -\infty$ or $\Phi(\boldsymbol{\theta}^{(t)})$ converges to a finite value. In the latter case, $\lim_{t\to\infty} \nabla\Phi(\boldsymbol{\theta}^{(t)}) = 0$. Furthermore, every cluster point of $(\boldsymbol{\theta}^{(t)})_{t\in\mathbb{N}}$ is a stationary point of Φ.

Proof. Let $t \in \mathbb{N}$. First of all, we reduce the scheme (30) to the form (32):

$$\begin{aligned}\boldsymbol{\theta}^{(t+1)} &= \boldsymbol{\theta}^{(t)} - \gamma_t \mathbf{A}_t^{-1} \sum_{i=1}^m \nabla\Phi_i(\boldsymbol{\omega}^{(t,i-1)}) \\ &= \boldsymbol{\theta}^{(t)} + \gamma_t \Big[\underbrace{-\mathbf{A}_t^{-1}\nabla\Phi(\boldsymbol{\theta}^{(t)})}_{\mathbf{s}_t} + \underbrace{\mathbf{A}_t^{-1} \sum_{i=1}^m \big(\nabla\Phi_i(\boldsymbol{\theta}^{(t)}) - \nabla\Phi_i(\boldsymbol{\omega}^{(t,i-1)})\big)}_{\mathbf{w}_t} \Big].\end{aligned}$$

This yields

$$\begin{cases} \mathbf{s}_t = -\mathbf{A}_t^{-1}\nabla\Phi(\boldsymbol{\theta}^{(t)}) \\ \mathbf{w}_t = \mathbf{A}_t^{-1} \sum_{i=1}^m \big(\nabla\Phi_i(\boldsymbol{\theta}^{(t)}) - \nabla\Phi_i(\boldsymbol{\omega}^{(t,i-1)})\big) \end{cases} \tag{33}$$

The vector $\mathbf{s}_t$ in (33) satisfies the first inequality in $(H1)$ since $\mathbf{A}_t^{-1}$ is a symmetric positive definite matrix and

$$\begin{aligned}\langle \nabla\Phi(\boldsymbol{\theta}^{(t)}), \mathbf{s}_t \rangle &= \langle \nabla\Phi(\boldsymbol{\theta}^{(t)}), -\mathbf{A}_t^{-1}\nabla\Phi(\boldsymbol{\theta}^{(t)}) \rangle = -\|\nabla\Phi(\boldsymbol{\theta}^{(t)})\|^2_{\mathbf{A}_t^{-1}} \\ &\le -\lambda_{\min}(\mathbf{A}_t^{-1})\|\nabla\Phi(\boldsymbol{\theta}^{(t)})\|^2 \le -\frac{1}{\chi}\|\nabla\Phi(\boldsymbol{\theta}^{(t)})\|^2,\end{aligned}$$

where $\lambda_{\min}(\cdot)$ (resp. $\lambda_{\max}(\cdot)$) denotes the smallest (resp. largest) eigenvalue of the symmetric matrix in argument. Thus the first inequality in $(H1)$ is simply verified for $c_1 = \frac{1}{\chi}$.
Moreover, $\mathbf{s}_t$ in (33) also satisfies the second inequality in $(H1)$ for $c_2 = \chi$:

$$\begin{aligned}\|\mathbf{s}_t\| &= \|-\mathbf{A}_t^{-1}\nabla\Phi(\boldsymbol{\theta}^{(t)})\| \le \|\mathbf{A}_t^{-1}\| \|\nabla\Phi(\boldsymbol{\theta}^{(t)})\| \\ &= \lambda_{\max}(\mathbf{A}_t^{-1})\|\nabla\Phi(\boldsymbol{\theta}^{(t)})\| \le \chi\|\nabla\Phi(\boldsymbol{\theta}^{(t)})\| \\ &\le \chi\big(1 + \|\nabla\Phi(\boldsymbol{\theta}^{(t)})\|\big).\end{aligned}$$

Let us now focus on $\mathbf{w}_t$ defined in (33). We have

$$\begin{aligned}\|\mathbf{w}_t\| &= \|\mathbf{A}_t^{-1} \sum_{i=1}^m \big(\nabla\Phi_i(\boldsymbol{\theta}^{(t)}) - \nabla\Phi_i(\boldsymbol{\omega}^{(t,i-1)})\big)\| \\ &\le \chi \sum_{i=1}^m \|\nabla\Phi_i(\boldsymbol{\theta}^{(t)}) - \nabla\Phi_i(\boldsymbol{\omega}^{(t,i-1)})\| \\ &\le \chi \sum_{i=1}^m \mu_i \|\boldsymbol{\theta}^{(t)} - \boldsymbol{\omega}^{(t,i-1)}\|,\end{aligned}$$

where the first inequality follows from the assumed boundedness of the eigenvalues of $\mathbf{A}_t$ and the second one follows from the Lipschitz regularity of the gradients of functions $(\Phi_i)_{1\le i\le m}$.
If, for each $i \in \{1,\dots,m\}$, there exist proper positive constants p_{i-1} and q_{i-1} such that

$$\|\boldsymbol{\theta}^{(t)} - \boldsymbol{\omega}^{(t,i-1)}\| \le \gamma_t(p_{i-1} + q_{i-1}\|\nabla\Phi(\boldsymbol{\theta}^{(t)})\|) \tag{34}$$

then (H2) follows with

$$\begin{cases} p = \chi \sum_{i=1}^m \mu_i p_{i-1} \\ q = \chi \sum_{i=1}^m \mu_i q_{i-1}. \end{cases}$$

Let us proceed by induction on $i \in \{1,\dots,m\}$ to show that (34) holds. First, we observe that for $i = 1$, the inequality obsviously holds with $p_0 = q_0 = 0$. Now we suppose that (34) holds and we prove that the same property is satisfied for the value i. Particularly we observe that

$$\begin{aligned}
\|\boldsymbol{\theta}^{(t)} - \boldsymbol{\omega}^{(t,i)}\| &\le \|\boldsymbol{\theta}^{(t)} - \boldsymbol{\omega}^{(t,i-1)}\| + \|\boldsymbol{\omega}^{(t,i-1)} - \boldsymbol{\omega}^{(t,i)}\| \\
&\le \gamma_t(p_{i-1} + q_{i-1}\|\nabla\Phi(\boldsymbol{\theta}^{(t)})\|) + \gamma_t\chi\|\nabla\Phi_i(\boldsymbol{\omega}^{(t,i-1)})\| \\
&\le \gamma_t(p_{i-1} + q_{i-1}\|\nabla\Phi(\boldsymbol{\theta}^{(t)})\|) + \gamma_t\chi\big(C + D\|\nabla\Phi(\boldsymbol{\omega}^{(t,i-1)})\|\big) \\
&\le \gamma_t(p_{i-1} + q_{i-1}\|\nabla\Phi(\boldsymbol{\theta}^{(t)})\|) \\
&\quad + \gamma_t\chi\big(C + D\|\nabla\Phi(\boldsymbol{\omega}^{(t,i-1)}) - \nabla\Phi(\boldsymbol{\theta}^{(t)})\| + D\|\nabla\Phi(\boldsymbol{\theta}^{(t)})\|\big) \\
&\le \gamma_t(p_{i-1} + q_{i-1}\|\nabla\Phi(\boldsymbol{\theta}^{(t)})\|) \\
&\quad + \gamma_t\chi\big(C + D\mu\|\boldsymbol{\omega}^{(t,i-1)} - \boldsymbol{\theta}^{(t)}\| + D\|\nabla\Phi(\boldsymbol{\theta}^{(t)})\|\big) \\
&\le \gamma_t(p_{i-1} + q_{i-1}\|\nabla\Phi(\boldsymbol{\theta}^{(t)})\|) \\
&\quad + \gamma_t\chi\big(C + D\mu\gamma_t(p_{i-1} + q_{i-1}\|\nabla\Phi(\boldsymbol{\theta}^{(t)})\|)\big) \\
&\quad + \gamma_t\chi\big(D\|\nabla\Phi(\boldsymbol{\theta}^{(t)})\|\big) \\
&\le \gamma_t(p_{i-1} + q_{i-1}\|\nabla\Phi(\boldsymbol{\theta}^{(t)})\|) \\
&\quad + \gamma_t\chi\big(C + D\mu\gamma_{\max}(p_{i-1} + q_{i-1}\|\nabla\Phi(\boldsymbol{\theta}^{(t)})\|)\big) \\
&\quad + \gamma_t\chi\big(D\|\nabla\Phi(\boldsymbol{\theta}^{(t)})\|\big) \\
&= \gamma_t(p_{i-1} + \chi C + \chi D\mu\gamma_{\max}p_{i-1}) \\
&\quad + \gamma_t(q_{i-1} + \chi D\mu\gamma_{\max}q_{i-1} + \chi D)\|\nabla\Phi(\boldsymbol{\theta}^{(t)})\|,
\end{aligned}$$

where $\gamma_{\max} = \sup_{t\in\mathbb{N}}\gamma_t$ is finite because of assumption (iv) on the step-size. Hereabove, the second inequality follows from the inductive assumption and the definition of the incremental method, the third inequality follows from assumption (ii), the fifth inequality follows from the μ-smoothness of $\nabla\Phi$ with $\mu = \sum_{i=1}^m \mu_i$, and the sixth inequality follows again from the inductive assumption. Thus we have shown that

$$\|\boldsymbol{\theta}^{(t)} - \boldsymbol{\omega}^{(t,i)}\| \le \gamma_t(p_i + q_i\|\nabla\Phi(\boldsymbol{\theta}^{(t)})\|)$$

where

$$\begin{cases} p_i = p_{i-1} + \chi(C + D\mu p_{i-1}\gamma_{\max}) \\ q_i = q_{i-1} + \chi(D + D\mu q_{i-1}\gamma_{\max}). \end{cases}$$

Then, applying Proposition 0.3 concludes the proof. □

Remark 4. *Let us assess the applicability of the assumptions made in Theorem 0.2 to the objective function in* (25)-(26) *and the incremental MM method reported in Algorithm 1.*

- *We firstly recall that, for the objective function in* (25)-(26)*,* $d = Q(n+1)$ *is the dimension of the input argument* $\boldsymbol{\theta}$*.*

- *In view of Proposition 0.1, Assumption (i) easily holds for the objective function in* (25)-(26) *with* $\mu_i = \beta\|\mathbf{L}^{(i)}\| + \frac{a\lambda+\eta}{m}$*.*

- *As for Assumption (ii), in view of the definitions* (17) *and* (28)*, we note that, for every* $\boldsymbol{\theta} \in \mathbb{R}^d$*, it holds that*

$$\begin{aligned}
\|\nabla\Phi_i(\boldsymbol{\theta})\| &= \left\|\mathbf{L}^{(i)\top}\nabla g_N(\mathbf{L}^{(i)}\boldsymbol{\theta}) + \frac{\nabla f(\boldsymbol{\theta})}{m}\right\| = \frac{1}{m}\left\|m\mathbf{L}^{(i)\top}\nabla g_N(\mathbf{L}^{(i)}\boldsymbol{\theta}) + \nabla f(\boldsymbol{\theta})\right\| \\
&= \frac{1}{m}\left\|m\mathbf{L}^{(i)\top}\nabla g_N(\mathbf{L}^{(i)}\boldsymbol{\theta}) + \nabla\Phi(\boldsymbol{\theta}) - \mathbf{L}^\top\nabla g_K(\mathbf{L}\boldsymbol{\theta})\right\| \\
&\le \left\|\mathbf{L}^{(i)\top}\nabla g_N(\mathbf{L}^{(i)}\boldsymbol{\theta})\right\| + \frac{1}{m}\left\|\mathbf{L}^\top\nabla g_K(\mathbf{L}\boldsymbol{\theta})\right\| + \frac{1}{m}\|\nabla\Phi(\boldsymbol{\theta})\| \\
&\le \left\|\mathbf{L}^{(i)\top}\right\|\left\|\nabla g_N(\mathbf{L}^{(i)}\boldsymbol{\theta})\right\| + \frac{1}{m}\left\|\mathbf{L}^\top\right\|\|\nabla g_K(\mathbf{L}\boldsymbol{\theta})\| + \frac{1}{m}\|\nabla\Phi(\boldsymbol{\theta})\| \\
&= \left\|\mathbf{L}^{(i)}\right\|\sqrt{\sum_{q=1}^{Q}\sum_{\nu=1}^{N}\left(\rho'\left((\mathbf{L}^{(i)}\boldsymbol{\theta})_{q,\nu}\right)\right)^2} + \\
&\quad + \frac{\|\mathbf{L}\|}{m}\sqrt{\sum_{q=1}^{Q}\sum_{k=1}^{K}\left(\rho'\left((\mathbf{L}\boldsymbol{\theta})_{q,k}\right)\right)^2} + \frac{1}{m}\|\nabla\Phi(\boldsymbol{\theta})\|.
\end{aligned}$$

If the function ρ' *is bounded — as in the case when* ρ *is either the logistic loss or the sigmoid loss — then Assumption (ii) is satisfied.*
On the other hand, if ρ' *is unbounded, as with the squared hinge loss, the required condition for the proof of Theorem 0.2 can still be ensured by assuming that the iterates remain bounded. This is because* ρ *is assumed to have a Lipschitz continuous derivative. It is worth noting that the assumption of bounded iterates is quite standard in the literature and, in particular, it is satisfied if* f *is coercive.*

- *If the regularization function* φ *in* (7) *is chosen such that* ψ *is upper bounded by a nonnegative constant* $\bar{\psi}$*, then the matrix* $\mathbf{A}_t$ *used in Algorithm 1 and defined in* (21)*, fulfills Assumption (iii). Indeed, denoting by* $\underline{a_t}$ *and* $\overline{a_t}$ *the* *minimum and maximum eigenvalues of* $\mathbf{A}_t$*, respectively, it follows from* (21) *that*

$$\beta\underline{\ell} + \underline{r_t} \le \underline{a_t} \le \overline{a_t} \le \beta\bar{\ell} + \overline{r_t},$$

where $0 \le \underline{\ell} \le \bar{\ell}$ *and* $0 < \underline{r_t} \le \overline{r_t}$ *are the minimum and maximum eigenvalues of* $\mathbf{L}^\top\mathbf{L}$ *and* $\mathbf{R}(\boldsymbol{\theta}^{(t)})$*, respectively. Since* $\min(\eta, \epsilon) \le \underline{r_t} \le \overline{r_t} \le \max(\lambda\bar{\psi} + \eta, \varepsilon)$*, there exists* $\chi > 0$ *such that* $\frac{1}{\chi} \le \underline{a_t} \le \overline{a_t} \le \chi$*. Specifically, if* φ *corresponds to the hyperbolic (resp. Welsh) potential, as done in our numerical experiments, then*

$$\min(\eta, \varepsilon) \le \underline{r_t} \le \overline{r_t} \le \max\left(\frac{\lambda}{\delta} + \eta, \varepsilon\right)$$

$$\left(\textit{resp.}\ \ \min(\eta, \varepsilon) \le \underline{r_t} \le \overline{r_t} \le \max\left(\frac{\lambda}{\delta^2} + \eta, \varepsilon\right)\right).$$

More details on the definition of the function ψ *defining* $\mathbf{R}(\boldsymbol{\theta})$ *when* φ *is either the hyperbolic or Welsh potential are given in Subsection 0.5.1.*

- *A choice for stepsizes satisfying Assumption (iv) is the following:*

$$(\forall t \in \mathbb{N}) \quad \gamma_t = \frac{\alpha_1}{\alpha_2 + t}, \quad \alpha_1 > 0, \alpha_2 > 0. \tag{35}$$

With the additional assumption of strict convexity, we can prove a further result for Algorithm 3. Before doing so, we first recall a key property of strictly convex functions.

Property 0.1. *[43, Corollary 27.2.2] Given a strictly convex function* $f : \mathbb{R}^p \longrightarrow \mathbb{R}$*, for any sequence* $(t^{(k)})_{k\in\mathbb{N}} \in \mathbb{R}^p$*, if* $\lim_{k\to+\infty} f(t^{(k)}) = f(x^*)$*, then* $\lim_{k\to+\infty} t^{(k)} = x^*$*.*

Theorem 0.3. *Let* $(\boldsymbol{\theta}^{(t)})_{t\in\mathbb{N}}$ *be a sequence generated by Algorithm 3. Under the same assumptions as in Theorem 0.2, let us further suppose that*

(i) there exists a compact set Ω *that contains* $(\boldsymbol{\theta}^{(t)})_{t\in\mathbb{N}}$*;*

(ii) Φ *is bounded from below;*

(iii) Φ *is strictly convex.*

Then $(\boldsymbol{\theta}^{(t)})_{t\in\mathbb{N}}$ *converges to the unique mininimizer* $\hat{\boldsymbol{\theta}}$ *of* Φ*.*

Proof. From assumption *(i)*, $(\boldsymbol{\theta}^{(t)})_{t\in\mathbb{N}}$ admits a convergent subsequence, namely there exist $(\boldsymbol{\theta}^{(t_\ell)})_{\ell\in\mathbb{N}}$ and $\hat{\boldsymbol{\theta}} \in \mathbb{R}^d$ such that $\lim_{\ell\to+\infty} \boldsymbol{\theta}^{(t_\ell)} = \hat{\boldsymbol{\theta}}$. From the continuity of $\nabla\Phi$, we deduce that $\lim_{\ell\to+\infty} \nabla\Phi(\boldsymbol{\theta}^{(t_\ell)}) = \nabla\Phi(\hat{\boldsymbol{\theta}})$. Moreover, in view of Theorem 0.2, $\hat{\boldsymbol{\theta}}$ is a stationary point of Φ, hence $\nabla\Phi(\hat{\boldsymbol{\theta}}) = 0$. The strict convexity of Φ guarantees that $\hat{\boldsymbol{\theta}}$ is its unique minimizer.

Theorem 0.2 ensures that there exists $l \in \mathbb{R}$ such that $\lim_{t\to+\infty} \Phi(\boldsymbol{\theta}^{(t)}) = l$. As a consequence, from the continuity of Φ, we get:

$$\lim_{\ell\to+\infty} \Phi(\boldsymbol{\theta}^{(t_\ell)}) = \Phi(\hat{\boldsymbol{\theta}}) = l,$$

and, hence,

$$\lim_{t\to+\infty} \Phi(\boldsymbol{\theta}^{(t)}) = \Phi(\hat{\boldsymbol{\theta}}).$$

Property 0.1 guarantees that

$$\lim_{t\to+\infty} \boldsymbol{\theta}^{(t)} = \hat{\boldsymbol{\theta}}.$$

□

Remark 5. *Some comments on the assumptions made in Theorem 0.3, in relation to the objective function* (25)-(26) *and Algorithm 1, are in order. Assumption (i) corresponds to the bounded iterates assumption, which was already discussed in 4. Moreover, if both the loss function ρ and the regularization function φ in* (25)-(26) *are chosen to be strictly convex and non-negative, then:*

- *the objective function Φ is non-negative and thus satisfies Assumption (ii);*
- *the objective function Φ is strictly convex.*

For instance, by selecting the hyperbolic potential as the regularization function and the logistic regression function as the loss, the assumptions in Theorem 0.3 are satisfied.

0.5 Experimental results

In this section, we present various numerical experiments carried out on several datasets and target functions demonstrating the good performance of Algorithm 1. The features of the training datasets and objective functions used are listed in Subsection 0.5.1. In Subsection 0.5.2, we compare the performance of Algorithm 1 when initialized with the warm-up phase detailed in Algorithm 2 versus a random starting point. Subsection 0.5.3 discusses the selection of hyperparameters associated with Algorithm 1.

Next, we present two different experiments. In the first experiment, detailed in Subsection 0.5.4, we examine a scenario where the standard MM algorithm (22) is applicable due to a sufficiently small value of K. This allows for a direct performance comparison with the incremental MM method we propose. In the second experiment, described in Subsection 0.5.5, Algorithm 1 is applied to a setting with a large number of examples, where using the standard MM method (22) may become infeasible.

These tests were performed on a machine with an Intel Core i5-6198DU processor and 12 GB of RAM.

0.5.1 Datasets and target functions

The description of the datasets used in the experiments is provided in Table 1. In

Dataset	$n+1$	K	**Test set examples**	Q
`protein`	358	14213	3553	3
`MNIST`	785	60000	10000	10
`covtype`	55	464810	116202	7

Table 1: Datasets features.

Tables 2 and 3, we list the considered loss and regularization functions, respectively. Both the loss and the regularization functions are Lipschitz differentiable, with the Lipschitz constant given in Tables 2 and 3. Additionally, the regularization functions φ satisfy assumptions 2, 3 and 4 reported in Subsection 0.2.2. In Table 3, δ is referred to as the smoothing parameter for the hyperbolic and the Welsh potential, and it controls the degree of approximation of the ℓ_1 norm and ℓ_0 pseudo-norm, respectively. Note that the former potential is convex, while the second is nonconvex. In Table 3 we also report the expression of the function ψ and the corresponding upper bound $\bar{\psi}$ which is here equal to the Lipschitz constant a of the function φ.

Function	**Analytical expression**	**Lipschitz constant β**
Squared hinge (SH)	$\rho^2_{hinge}(v) = (\max\{1-v;0\})^2$	2
Sigmoid (SGM)	$\rho_{sigm}(v) = (1+\exp(v))^{-1}$	$\frac{1}{6\sqrt{3}}$
Logistic regression (LR)	$\rho_{log}(v) = \ln(1+\exp(-v))$	$\frac{1}{4}$

Table 2: Fidelity functions ρ.

Function	$\varphi(v)$	$\psi(v)$	$a = \bar{\psi}$
Hyperbolic potential (HP)	$\sqrt{v^2+\delta^2},\ \delta > 0$	$\dfrac{1}{\sqrt{v^2+\delta^2}}$	$\dfrac{1}{\delta}$
Welsh potential (WP)	$1-\exp\left(-\dfrac{v^2}{2\delta^2}\right),\ \delta > 0$	$\dfrac{1}{\delta^2}\exp\left(-\dfrac{v^2}{2\delta^2}\right)$	$\dfrac{1}{\delta^2}$

Table 3: Regularization functions φ.

For all the experiments, we set the regularization parameters in (7) to $\lambda = 10^{-3}$ and $\eta = 1$, based on empirical tuning. On the other hand, the parameter δ for the Welsh potential and the hyperbolic potential was set to 10^{-1} and 10^{-4}, respectively. These values were chosen to ensure the numerical stability of the inversion of matrix $\mathbf{A}_t$. Indeed, it follows from Equation 21 that, if the matrix $\mathbf{L}^\top\mathbf{L}$ is ill-conditioned, the diagonal matrix $\mathbf{R}(\boldsymbol{\theta})$ derived from the regularization terms plays a crucial role in ensuring the stable computation of $\mathbf{A}_t^{-1}$. In particular, each i-th diagonal element of $\mathbf{R}(\boldsymbol{\theta})$ is computed as the sum of η and $\lambda\psi(\theta_i)$, where the value of $\psi(\theta_i) = \varphi'(\theta_i)/\theta_i$ also depends on δ, as shown in the third column of Table 3. In the case of the considered experiments, the smallest eigenvalue of $\mathbf{L}^\top\mathbf{L}$ is very close or equal to 0, thus the matrix is ill-conditioned. Therefore, the parameters η, λ and δ, in addition to having a meaningful role in the regularizing effect of the model Equation 6, are essential for the numerical stability of the algorithm.

0.5.2 Efficient computation of the initial point

In this section, we describe a way to modify the warm-up phase detailed in Algorithm 2 so that it can be used not only to construct the matrix $\mathbf{L}^\top\mathbf{L}$, but also to obtain a reliable initial point for Algorithm 1.

This modified warm-up phase is detailed in Algorithm 4 and consists of a single epoch of incremental gradient descent, where the gradient direction is scaled by a varying scaling matrix at each iteration. This matrix, denoted as $\mathbf{C}_i$, is inspired by (21). As in Algorithm 2, the matrix $\mathbf{H}_i$ progressively approximates $\mathbf{L}^\top\mathbf{L}$.

Hereafter, we present a comparison of Algorithm 1 when initialized with the output $\boldsymbol{\omega}^{(m)}$ of the warm-up phase presented in Algorithm 4 versus a random starting point. Hereafter, we denote $\boldsymbol{\theta}_0^{\mathrm{OPT}} = \boldsymbol{\omega}^{(m)}$.

To verify that $\boldsymbol{\theta}_0^{\mathrm{OPT}}$ can be an effective initial point for Algorithm 1, we conducted an empirical test on the `protein` dataset. For the target function Φ in (6), we used the squared hinge loss for ρ and the hyperbolic potential for φ. For Algorithm 4, we set $\bar{\gamma}$ equal to 1, following the step-size value in the standard MM approach (22), and $m = 10$.

First, we compare the value of the objective function Φ at $\boldsymbol{\theta}_0^{\mathrm{OPT}}$ and the mean of the values of Φ on 100 random initial points, namely $\Phi_{\mathrm{rand}} = \dfrac{1}{100}\displaystyle\sum_{j=1}^{100}\Phi(\tilde{\boldsymbol{\theta}}_j)$, where $\tilde{\boldsymbol{\theta}}_j$ is drawn from a standard normal distribution, i.e., each component has zero mean and unit variance. Table 4 reports $\Phi(\boldsymbol{\theta}_0^{\mathrm{OPT}})$ and Φ_{rand}, along with the relative error with respect to the optimal value $\Phi^\star$ achieved by applying the MM method (22) for 1000 epochs. The values in Table 4 clearly illustrate that $\boldsymbol{\theta}_0^{\mathrm{OPT}}$ serves as a good initial point in terms of objective function value reduction. We emphasize that the values shown in Table 4 are reported solely for quantitative analysis. Computing the objective function Φ at different starting points in order to identify the one yielding the lowest value is not efficient. Indeed, especially for large training sets, evaluating Φ can be computationally expensive or even untractable.

Algorithm 4 Modified warm-up phase

```
    Set $\boldsymbol{\omega}^{(0)} \in \mathbb{R}^{Q(n+1)}$ as a random vector from a standard normal distribution, $\bar{\gamma} > 0$, $m > 0$, $\beta > 0$ as the Lipschitz constant defined in Table 2, $\lambda > 0$, $\eta > 0$, $\mathbf{H}_0$ as the null matrix in $\mathbb{R}^{Q(n+1)\times Q(n+1)}$.
    Split the training dataset into blocks $(\mathcal{D}_i)_{1\le i\le m}$, as defined in (23)-(24).
    for $i = 1, \dots, m$ do
    Compute the tensor $\mathbf{L}^{(i)}$ corresponding to the block $\mathcal{D}_i$.
    Compute $\mathbf{H}_i = (\mathbf{L}^{(i)})^\top \mathbf{L}^{(i)} + \mathbf{H}_{i-1}$ and
        $$\mathbf{C}_i = \beta\mathbf{H}_i + \mathrm{Diag}\left(\begin{bmatrix} \lambda\psi(\boldsymbol{\omega}_1^{(i-1)}) + \eta \\ \vdots \\ \lambda\psi(\boldsymbol{\omega}_n^{(i-1)}) + \eta \\ \vdots \\ \lambda\psi(\boldsymbol{\omega}_{(Q-1)\cdot n+1}^{(i-1)}) + \eta \\ \vdots \\ \lambda\psi(\boldsymbol{\omega}_{Q\cdot n}^{(i-1)}) + \eta \\ \varepsilon \\ \vdots \\ \varepsilon \end{bmatrix}\right)$$
    Set $\boldsymbol{\omega}^{(i)} = \boldsymbol{\omega}^{(i-1)} - \bar{\gamma}\mathbf{C}_i^{-1}\nabla\Phi_i(\boldsymbol{\omega}^{(i-1)})$
    end for
```

$\boldsymbol{\Phi_{\text{rand}}}$	$\boldsymbol{\dfrac{\lvert\Phi_{\text{rand}} - \Phi^\star\rvert}{\lvert\Phi^\star\rvert}}$	$\boldsymbol{\Phi(\theta_0^{\text{OPT}})}$	$\boldsymbol{\dfrac{\lvert\Phi(\theta_0^{\text{OPT}}) - \Phi^\star\rvert}{\lvert\Phi^\star\rvert}}$
4.0259×10^5	24.94	3.1474×10^4	1.03

Table 4: Example on the `protein` dataset of the comparison between the objective function values at both the warm-up and a random initialization of $\boldsymbol{\theta}^{(0)}$.

Figure 2 illustrates the decrease in the relative optimality gap, defined as $\dfrac{|\Phi(\boldsymbol{\theta}^{(k)}) - \Phi^\star|}{|\Phi^\star|}$, obtained by running Algorithm 1 with both $\boldsymbol{\theta}_0^{\text{OPT}}$ and a random initial point $\boldsymbol{\theta}^{(0)}$. In both cases, the other hyperparameters for Algorithm 1 have been set as $m = 10$, $\gamma_0 = 1$, $\gamma_t = \gamma_0\frac{100}{100+t}$. From Figure 2 we can conclude that initializing Algorithm 1 with $\boldsymbol{\theta}_0^{\text{OPT}}$ significantly accelerates the optimization process, particularly during the first epochs.

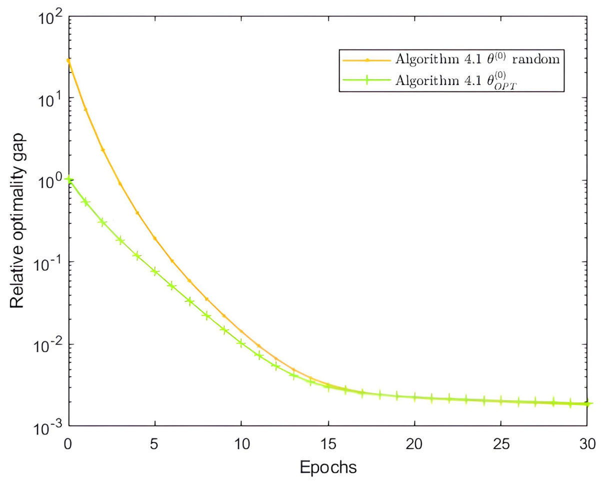


Figure 2: Relative optimality gap achieved on the `protein` dataset by Algorithm 1 initialized by the output of the warm-up phase and a random vector.

This experiment shows that leveraging the warm-up phase in Algorithm 4 to compute an initial point via a single epoch of an incremental MM-like method provides a good initialization for Algorithm 1. In all the experiments presented in Sections 0.5.3, 0.5.4 and 0.5.5, $\boldsymbol{\theta}_0^{\text{OPT}}$ is used as the initial point for Algorithm 1 and all its competitors.

0.5.3 Hyperparameters selection for Algorithm 1

The implementation of Algorithm 1 depends on the choice of the step-size sequence $(\gamma_t)_t$ and the number of blocks (i.e., mini-batches) m used to partition the training dataset. To satisfy the assumptions on the step-size required for the theoretical results, we chose it according to (35) with $\alpha_1 = \gamma_0\alpha_2$ and $\alpha_2 = 100$. Below, we discuss how γ_0 and m were determined for the numerical experiments.

Choice of m To analyze the impact of m on the performance of Algorithm 1, we considered classification problems on the small-sized `protein` dataset and the large-sized `MNIST` dataset. For the objective function in (6), we used the squared hinge loss for ρ and the hyperbolic potential for φ. For both datasets, Algorithm 1 was evaluated for $m \in \{3, 10, 100\}$. The step-size γ_0 is set to 1 and $\boldsymbol{\theta}_0^{\mathrm{OPT}}$ is used as the initial point. Figure 3 illustrates the decrease in the objective function achieved using Algorithm 1 with different values of m. Figures 3(a) and 3(c) show that the choice of m does not significantly impact the decrease in the objective function, as all curves are nearly identical. On the other hand, Figures 3(b) and 3(d) indicate that m affects the computational time: using a small number of blocks leads to handling large matrices, which slows down the optimization process. As a result, choosing $m = 10$ provides a good trade-off between block size and the number of update steps per epoch. For the numerical experiments in the following sections, this value of m was chosen.

Choice of γ_0 As for the selection of γ_0, we consider three possible strategies.

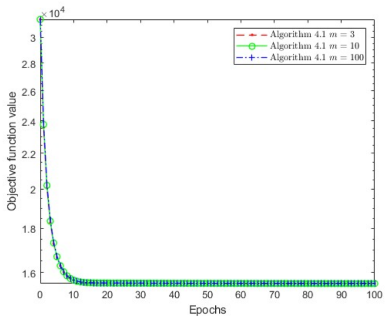
1. Inspired by the fact that in the standard MM method (22), the step-size is constant and equal to 1, a natural choice is to set $\gamma_0 = 1$. This option incurs no additional computational cost, but a larger value of γ_0 may lead to faster convergence of Algorithm 1.

2. The second approach is to select γ_0 using a *grid-search*. Specifically, for φ set as the hyperbolic potential, we run Algorithm 1 with different values of γ_0 for each combination of training dataset in Table 1 and fidelity function ρ in Table 2. We then evaluate which values yield the best performance in terms of objective function reduction.

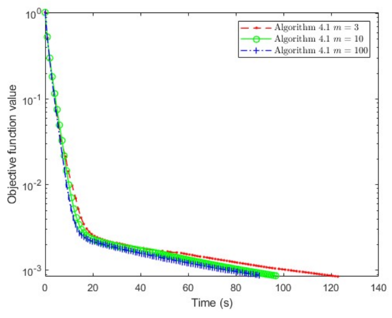
   The tested values for γ_0 are $\{1, 5, 10, 15, 20\}$. The value of m is set to 10, and $\boldsymbol{\theta}^{(0)}$ is initialized as $\boldsymbol{\theta}_0^{\mathrm{OPT}}$. We assess the performance of Algorithm 1 over the first 10 epochs, as the choice of γ_0 significantly impacts the method's behavior, particularly in the initial iterations when t is relatively small. Additionally, limiting the evaluation to 10 epochs helps reduce the computational cost associated with tuning γ_0.

   Table 5 provides the optimal γ_0 values for each dataset and fidelity function. The values reported in Table 5 were also used when φ was chosen as the Welsh potential.

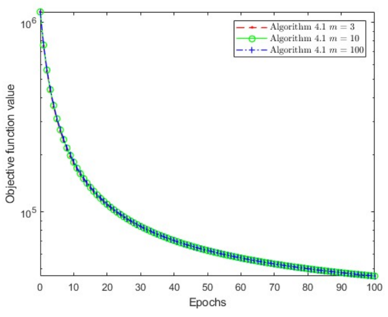
3. The last strategy to select a value for γ_0 relies on a line-search procedure aimed at finding m proper step-sizes $(\tilde{\gamma}_i)_{1\leq i\leq m}$, which aim at ensuring a



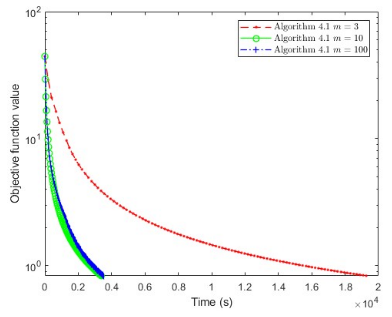
(a) Results for `protein` (epochs).



(b) Results for `protein` (time).



(c) Results for `MNIST` (epochs).



(d) Results for `MNIST` (time).

Figure 3: Decrease in the objective function achieved by Algorithm 1 for different values of m in multiclass classification problems on the `protein` and `MNIST` datasets.

sufficient decrease in functions $(\Phi_i)_{1\le i\le m}$, $i = 1, \dots, m$, while moving from $\boldsymbol{\theta}^{(0)}$ to $\boldsymbol{\theta}^{(0)} - \tilde{\gamma}_i \mathbf{A}_0^{-1} \nabla\Phi_i(\boldsymbol{\theta}^{(0)})$, where $\mathbf{A}_0 = \beta\mathbf{L}^\top\mathbf{L} + R(\boldsymbol{\theta}^{(0)})$. Then the final value of γ_0 is computed as the average of the values $\tilde{\gamma}_i$ identified by the line-search at each $i \in \{1, \dots, m\}$. The line-search strategy for the selection of γ_0 is reported in Algorithm 5. The Armijo-type line-search procedure in steps 7–11 of Algorithm 5 is used to select a suitable step-size within the interval $[1, \overline{\gamma}]$. This step-size ensures a sufficient decrease in the function Φ_i when applying a single iteration of a scaled gradient descent method starting from $\boldsymbol{\theta}^{(0)}$. Table 6 reports the values for γ_0 obtained by running Algorithm 5 with $k_{\max} = 50$, $\xi = 0.875$, $\overline{\gamma} = 25$, $m = 10$ for the different combinations of training dataset and fidelity functions ρ. The function φ is chosen as the hyperbolic potential.

Dataset	Squared hinge (SH)	Sigmoid (SGM)	Logistic regression (LR)
`protein`	1	5	1
`MNIST`	15	20	15
`covtype`	5	20	5

Table 5: Optimal γ_0 for Algorithm 1 found by means of a grid-search for the different datasets and fidelity functions.

Algorithm 5 Line-search procedure for γ_0

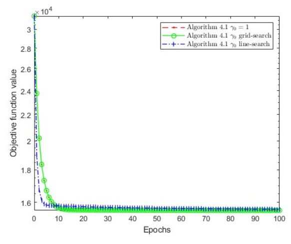
```
1: Set $\boldsymbol{\theta}^{(0)} \in \mathbb{R}^{Q(n+1)}$, $m > 0$, $\mathbf{A}_0 = \beta \mathbf{L}^\top \mathbf{L} + R(\boldsymbol{\theta}^{(0)})$, $k_{\max}$, $\xi$, $\overline{\gamma}$.
2: Set $\nu = \left(\frac{1}{\overline{\gamma}}\right)^{\frac{1}{k_{\max}}}$
3: for $i = 1, \dots, m$ do
4:     $\gamma_i = \overline{\gamma}$
5:     $\hat{\boldsymbol{\theta}} = \boldsymbol{\theta}^{(0)} - \gamma_i \mathbf{A}_0^{-1} \nabla \Phi_i(\boldsymbol{\theta}^{(0)})$
6:     $k = 1$
7:     while $\left(\Phi_i(\hat{\boldsymbol{\theta}}) > \Phi_i(\boldsymbol{\theta}^{(0)}) - \xi \gamma_i \nabla \Phi_i(\boldsymbol{\theta}^{(0)})^\top \mathbf{A}_0^{-1} \nabla \Phi_i(\boldsymbol{\theta}^{(0)})\right)$ & $(k \le k_{\max})$ do
8:         $\gamma_i = \nu \overline{\gamma}$
9:         $\hat{\boldsymbol{\theta}} = \boldsymbol{\theta}^{(0)} - \gamma_i \mathbf{A}_0^{-1} \nabla \Phi_i(\boldsymbol{\theta}^{(0)})$
10:        $k = k + 1$
11:    end while
12: end for
13: $\gamma_0 = \frac{1}{m} \sum_{i=1}^{m} \gamma_i$
```

Dataset	Squared hinge (SH)	Sigmoid (SGM)	Logistic regression (LR)
`protein`	2.54	12.03	2.98
`MNIST`	2.71	19.38	9.96
`covtype`	1.99	3.18	2.48

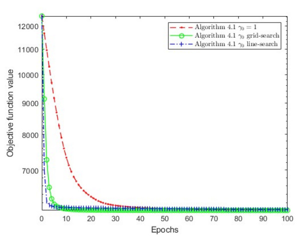
Table 6: Values for γ_0 selected via the line-search procedure in Algorithm 5

We compared the performance of Algorithm 1 using the three different strategies for selecting γ_0 described above. Moreover we considered $m = 10$, $\boldsymbol{\theta}^{(0)} = \boldsymbol{\theta}^{(0)}_{\mathrm{OPT}}$, and $(\gamma_t)_t$ as defined before. The decrease in the objective function obtained by applying Algorithm 1 with different γ_0 values is shown in Figure 4. The final objective function value after 100 epochs is reported in Table 7.

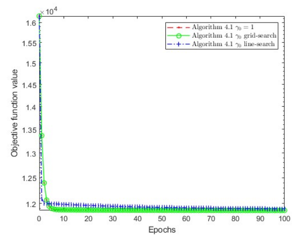
In most cases, the best strategy for selecting γ_0 is the grid-search procedure, despite its higher computational cost. However, the performance of Algorithm 1 when γ_0 is chosen via the line-search procedure is comparable to that obtained with the optimal γ_0 from grid-search. In the experiments reported in Subsections 0.5.4 and 0.5.5, we set γ_0 using the value obtained from the grid-search, as shown in Table 5. However, we emphasize that Algorithm 5 provides a suitable alternative for selecting γ_0, avoiding the use of grid-search. Finally, we note that the values in Table 5, identified for the hyperbolic potential regularization term, were also used for the Welsh potential.



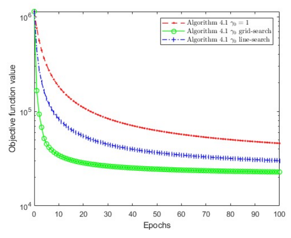
(a) `protein` + SH.



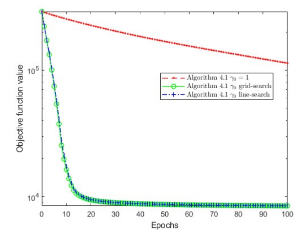
(b) `protein` + SGM.



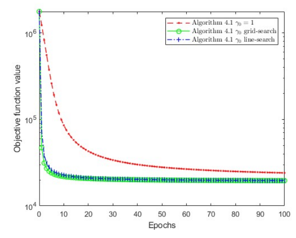
(c) `protein` + LR.



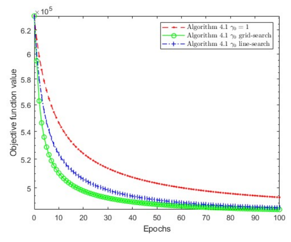
(d) `MNIST` + SH.



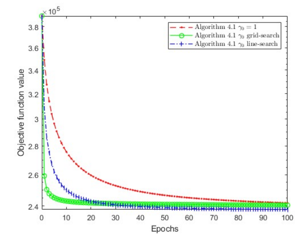
(e) `MNIST` + SGM.



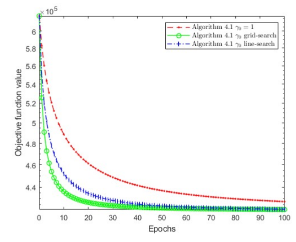
(f) `MNIST` + LR.



(g) `covtype` + SH.



(h) `covtype` + SGM.



(i) `covtype` + LR.

Figure 4: Decrease of the objective function obtained using Algorithm 1 with different strategies for selecting γ_0, evaluated across different datasets and fidelity functions.

Dataset	**Target function**	$\gamma_0 = 1$	**Grid-search**	**Line-search**
protein	SH + HP	1.5532×10^4	1.5532×10^4	1.5595×10^4
	SGM + HP	6.0308×10^3	6.0229×10^3	6.0415×10^3
	LR + HP	1.1866×10^4	1.1866×10^4	1.1895×10^4
MNIST	SH + HP	4.5935×10^4	2.2851×10^4	3.0120×10^4
	SGM + HP	1.1349×10^5	8.4493×10^3	8.4595×10^3
	LR + HP	2.4071×10^4	1.9566×10^4	1.9650×10^4
covtype	SH + HP	4.9374×10^5	4.8562×10^5	4.8677×10^5
	SGM + HP	2.4188×10^5	2.4050×10^5	2.3781×10^5
	LR + HP	4.2761×10^5	4.2092×10^5	4.2121×10^5

Table 7: Objective function value after 100 epochs achieved by Algorithm 1 for different choices of γ_0.

0.5.4 Results on small-scale dataset

The main purpose of this section is to compare Algorithm 1 with both its non-incremental version, defined in (22), and a standard first-order descent algorithm. To evaluate the performance of these three approaches, we considered the `protein` dataset, where K is not too large, making it feasible to apply the MM method in (22). More specifically, we compare the following methods:

- **Algorithm 1** denoted as I-MM, where $(\gamma_t)_t$ is chosen as explained in the previous section, $m = 10$, and the initial point $\boldsymbol{\theta}^{(0)}$ is set as $\boldsymbol{\theta}_0^{\mathrm{OPT}}$, obtained using Algorithm 4 with $\bar{\gamma} = 1$;

- the **standard MM algorithm** in (22), where $\boldsymbol{\theta}^{(0)}$ and $\mathbf{L}^\top\mathbf{L}$ are initialized in the same way as in Algorithm 1, using Algorithm 4 with $\bar{\gamma} = 1$;

- the **standard gradient descent (GD) method**, given by

$$(\forall t \in \mathbb{N}) \quad \boldsymbol{\theta}^{(t+1)} = \boldsymbol{\theta}^{(t)} - \gamma \nabla \Phi(\boldsymbol{\theta}^{(t)}),$$

  where the step-size γ is set to $1.9999/\mu$, where μ is the Lipschitz constant defined in (19).

Figure 5 illustrates the decrease in the objective function $\Phi(\boldsymbol{\theta}^{(t)})$ over 50 epochs for all the considered test problems, when varying ρ and φ. In Tables 8 and 9, we report the accuracy computed on the test set and the final value of the objective function on the training set, respectively.

Target function	**MM**	**I-MM**	**GD**
$SH + HP$	0.6811	0.6806	0.6310
$SH + WP$	0.6811	0.6806	0.6310
$SGM + HP$	0.6828	0.6811	0.4494
$SGM + WP$	0.6828	0.6811	0.4494
$LR + HP$	0.6814	0.6825	0.6200
$LR + WP$	0.6814	0.6822	0.6198

Table 8: Accuracy achieved after 50 epochs by I-MM, MM, and GD on the `protein` test set for different ρ and φ.

Target function	**MM**	**I-MM**	**GD**
$SH + HP$	1.5522×10^4	1.5542×10^4	2.7757×10^4
$SH + WP$	1.5522×10^4	1.5542×10^4	2.7757×10^4
$SGM + HP$	6.0462×10^3	6.0258×10^3	1.1614×10^4
$SGM + WP$	6.0464×10^3	6.0260×10^3	1.1615×10^4
$LR + HP$	1.1862×10^4	1.1869×10^4	1.4880×10^4
$LR + WP$	1.1863×10^4	1.1869×10^4	1.4880×10^4

Table 9: Final objective function value achieved after 50 epochs by I-MM, MM, and GD on the `protein` training set for different ρ and φ.

From these results, we can conclude that the I-MM algorithm effectively leverages second-order information from the MM method. Moreover, in terms of epoch number, we observe that

- the performance of the MM and I-MM methods, as reflected in plots and final objective function values, is comparable;
- when using the sigmoid function, I-MM exhibits faster convergence during the initial epochs: this is a typical behavior of incremental gradient schemes compared to full gradient algorithms;
- both MM and I-MM outperform the GD method in overall performance owing to the presence of the scaling matrix.

Tables 10 and 11 report the computational times needed by MM and I-MM, respectively, on the `protein` dataset to reduce the relative difference between the objective function values at two successive iterations,

$$\frac{|\Phi(\boldsymbol{\theta}^{(t+1)}) - \Phi(\boldsymbol{\theta}^{(t)})|}{|\Phi(\boldsymbol{\theta}^{(t+1)})|}$$

below specific thresholds ε. The corresponding accuracy values are also provided.

It is evident that I-MM requires more time than MM to satisfy the same stop-

	$\varepsilon = 10^{-2}$		$\varepsilon = 10^{-3}$		$\varepsilon = 10^{-4}$	
Target function	**Accuracy**	**Time**	**Accuracy**	**Time**	**Accuracy**	**Time**
$SH + HP$	0.6783	1.20	0.6797	1.62	0.6808	1.97
$SH + WP$	0.6783	0.92	0.6797	1.32	0.6808	1.62
$SGM + HP$	0.6848	2.35	0.6881	4.17	0.6836	8.05
$SGM + WP$	0.6848	2.17	0.6882	3.98	0.6834	8.31
$LR + HP$	0.6820	0.42	0.6822	0.69	0.6808	1.00
$LR + WP$	0.6820	0.43	0.6822	0.65	0.6811	1.01

Table 10: Accuracy achieved and execution times required by MM to bring the relative difference between the objective function at two successive iterates given thresholds ε on the `protein` dataset for different ρ and φ.

	$\varepsilon = 10^{-2}$		$\varepsilon = 10^{-3}$		$\varepsilon = 10^{-4}$	
Target function	**Accuracy**	**Time**	**Accuracy**	**Time**	**Accuracy**	**Time**
$SH + HP$	0.6783	7.58	0.6805	11.79	0.6805	15.98
$SH + WP$	0.6783	7.54	0.6805	11.80	0.6805	15.96
$SGM + HP$	0.6907	11.21	0.6820	16.47	0.6817	32.73
$SGM + WP$	0.6907	6.08	0.6820	11.23	0.6817	22.01
$LR + HP$	0.6822	3.38	0.6808	5.91	0.6825	8.47
$LR + WP$	0.6822	3.53	0.6808	6.15	0.6825	8.79

Table 11: Accuracy achieved and execution times required by I-MM to bring the relative difference between the objective function at two successive iterates given thresholds ε on the `protein` dataset for different ρ and φ.

ping criterion. The reason is that, for any fixed t-th epoch, MM requires a single matrix–vector product, $\mathbf{A}_t^{-1}\nabla\Phi(\boldsymbol{\theta}^{(t)})$, while I-MM requires m matrix–vector products, $\mathbf{A}_t^{-1}\nabla\Phi_i(\boldsymbol{\omega}^{(t,i-1)})$, each with the same computational complexity as the one computed by MM. However, we emphasize that our aim is not to design an algorithm that accelerates MM, but rather to develop a variant that can be applied to large-scale datasets where the main challenge is memory saturation. Indeed, the advantage of I-MM lies in its ability to handle the gradients of the individual functions Φ_i, whereas the full gradient $\nabla\Phi$ cannot be stored, rendering MM inapplicable. Finally, we note that the computational time strongly depends on the architecture of the device on which the numerical experiments are performed. A different architecture can lead to a different scenario. For instance, if the numerical experiments had been carried out on a GPGPU-based architecture, which is designed to perform tensor operations efficiently, the computational burden associated with the m matrix–vector multiplications in I-MM could have been significantly reduced.

Comparison with a standard one-vs-one SVM classifier

To conclude this section, we report the results of multiclass classification on the `protein` dataset using a one-vs-one scheme implemented through the error-correcting output codes (ECOC) framework [3]. In this setting, a separate binary SVM classifier is trained for every possible pair of classes in the dataset. Specifically, the model was trained using MATLAB's built-in `fitcecoc` function. An accuracy of 0.6882 was achieved on the `protein` test set in 70.87 seconds. Based on the results reported in Tables 10 and 11, we can conclude that both MM and I-MM achieve comparable accuracy values with lower computational time.

0.5.5 Results on large-scale datasets

This section aims to evaluate the performance of Algorithm 1 on large-scale datasets where K is too large to compute the gradient of the objective function in (12). Consequently, the MM and GD schemes from the previous section cannot be applied. In this setting, we compare I-MM with both a standard incremental gradient method and a stochastic gradient method. The large-scale datasets considered are `MNIST` and `covtype`.

The methods compared in these experiments are:

- **Algorithm 1** with the same parameter setting as in the previous example;
- the standard **stochastic gradient (SG) method** [11] with a fixed mini-batch size and a decreasing step-size. Specifically, the mini-batch size is set equal to the number of samples in one block of I-MM to ensure a fair comparison. The step-size sequence follows the same form as in Algorithm 1, the only difference lying in the choice of γ_0, selected via grid-search over the set

$$\{10^{-2}, 10^{-3}, 5\times10^{-4}, 10^{-4}, 5\times10^{-5}, 10^{-5}, 5\times10^{-6}, 10^{-6}\}.$$

  Given the hyperbolic potential regularization term, the SG algorithm was run for 100 epochs for each dataset and fidelity function. The optimal γ_0 value was determined as the one that minimizes the average objective function value over the last five iterations.

The optimal γ_0 values for each dataset and fidelity function are provided in Table 12. The same values identified for the hyperbolic potential regularization term were also used for the Welsh potential;

Dataset	Squared hinge	Sigmoid	Logistic regression
`MNIST`	10^{-6}	10^{-2}	10^{-4}
`covtype`	5×10^{-6}	10^{-3}	5×10^{-5}

Table 12: Optimal value of γ_0 for the SG algorithm.

- the standard **incremental gradient (IG) method** [9], which corresponds to Algorithm 1 with $\mathbf{A}_t$ set to the identity matrix for all $t \in \mathbb{N}$. The number of blocks, m, is set to 10 to ensure a fair comparison with I-MM. The decreasing step size follows the same formulation as in I-MM and SG, with the only difference being the choice of γ_0, selected as described for SG. In Table 13, the optimal γ_0 values for each dataset and fidelity function are provided. The same values identified for the hyperbolic potential regularization term were also used for the Welsh potential. The values reported in Table 13 indicate that the optimal γ_0 for IG coincides with that of SGD in many cases.

Dataset	Squared hinge	Sigmoid	Logistic regression
`MNIST`	10^{-6}	10^{-2}	10^{-4}
`covtype`	5×10^{-6}	5×10^{-4}	5×10^{-5}

Table 13: Optimal value of γ_0 for the IG algorithm.

Figures 6, 7, 8 and 9 illustrate the decrease in the objective function provided by the compared methods over 100 epochs for all test problems considered. In Tables 15 and 17, the final test set accuracy is reported. Similarly, Tables 16 and 18 present the final objective function value computed on the training set. From the results reported in these figures and tables, we conclude that I-MM outperforms both IG and SG in terms of objective function decrease and test set accuracy. The computational time per epoch for I-MM is comparable to that of first-order methods, even for larger datasets. This confirms the effectiveness of the proposed procedure in handling the inversion of the non-sparse scaling matrix $\mathbf{A}_t$. From the figures showing the objective function decrease, we observe that I-MM enables a stable and smooth reduction of the objective function. Furthermore, it is evident that I-MM achieves very good results in the early stages of the optimization process.

This confirms that selecting γ_0 via grid-search by evaluating the objective function behavior within the first 10 epochs is an effective strategy. For first-order methods, a similar procedure cannot be used, as the optimization process is slower. Finally, we note that for the same data fidelity function ρ, the accuracy results remain very similar across different regularization terms φ.

Remark 6. *To investigate the benefits of the sparsity-inducing term, we performed an additional numerical experiment. Specifically we considered the objective function defined by the squared hinge loss combined with the hyperbolic potential. For each of the three datasets in Table 1, we evaluated the final accuracy and the number of solution components with absolute values below the threshold of* 10^{-4}*, as obtained by the I-MM algorithm after 100 epochs, for five values of* λ*:* 0, 10^{-3}, 1, 10^3*, and* 10^6*. From the results shown in Table 14, we found that, as expected, increasing the value of* λ *leads to more coefficients falling below the considered threshold. However, an excessively large value of* λ *results in a solution where most components are zero, which in turn reduces the accuracy. These results highlight that the proposed model effectively promotes sparsity in the solution while maintaining reliable accuracy, provided that the regularization parameter* λ *is properly tuned. A reduced number of nonzero parameters for the learned classifier lowers the risk of overfitting and improves model explainability by removing non-significant features.*

		`protein`	`MNIST`	`covtype`
Final accuracy	$\lambda = 0$	0.6811	0.9117	0.6015
	$\lambda = 10^{-3}$	0.6811	0.9117	0.6015
	$\lambda = 1$	0.6811	0.9126	0.6016
	$\lambda = 10^3$	0.4869	0.8853	0.5711
	$\lambda = 10^6$	0.4869	0.6430	0.4035
Number of sparse components	$\lambda = 0$	3	678	15
	$\lambda = 10^{-3}$	3	694	33
	$\lambda = 1$	316	1807	262
	$\lambda = 10^3$	1050	5962	355
	$\lambda = 10^6$	1073	7850	382

Table 14: Final accuracy and number of solution components with absolute value below 10^{-4}, obtained after 100 epochs by the I-MM algorithm. Results are reported for different values of λ, using the squared hinge loss as ρ and the hyperbolic potential as φ.

Target function	**I-MM**	**IG**	**SG**
$SH + HP$	0.9117	0.8373	0.8371
$SH + WP$	0.9117	0.8372	0.8371
$SGM + HP$	0.9251	0.9223	0.9236
$SGM + WP$	0.9251	0.9126	0.9226
$LR + HP$	0.9170	0.8916	0.8909
$LR + WP$	0.9170	0.8916	0.8909

Table 15: Final accuracy on the test set achieved after 100 epochs by I-MM, IG, and SG on the `MNIST` test set for different ρ and φ.

Target function	**I-MM**	**IG**	**SG**
$SH + HP$	2.2851×10^4	2.2143×10^5	2.2345×10^5
$SH + WP$	2.2854×10^4	2.2141×10^5	2.2343×10^5
$SGM + HP$	8.4493×10^3	9.3122×10^3	9.4292×10^3
$SGM + WP$	8.4562×10^3	1.0184×10^4	9.5285×10^3
$LR + HP$	1.9566×10^4	3.6905×10^4	3.7765×10^4
$LR + WP$	1.9569×10^4	3.6906×10^4	3.7765×10^4

Table 16: Final objective function value achieved after 100 epochs by I-MM, IG, and SG on the `MNIST` training set for different ρ and φ.

Target function	**I-MM**	**IG**	**SG**
$SH + HP$	0.6015	0.5738	0.5621
$SH + WP$	0.6015	0.5738	0.5621
$SGM + HP$	0.5504	0.5443	0.5491
$SGM + WP$	0.5504	0.5443	0.5491
$LR + HP$	0.5882	0.5571	0.5599
$LR + WP$	0.5882	0.5571	0.5599

Table 17: Final accuracy achieved after 100 epochs by I-MM, IG, and SG on the `covtype` test set for different ρ and φ.

Target function	**I-MM**	**IG**	**SG**
$SH + HP$	4.8562×10^5	6.1219×10^5	5.8890×10^5
$SH + WP$	4.8562×10^5	6.1218×10^5	5.8889×10^5
$SGM + HP$	2.4050×10^5	2.9954×10^5	2.5420×10^5
$SGM + WP$	2.4050×10^5	2.9955×10^5	2.5414×10^5
$LR + HP$	4.2092×10^5	5.4003×10^5	5.1979×10^5
$LR + WP$	4.2092×10^5	5.4003×10^5	5.1979×10^5

Table 18: Final objective function value achieved after 100 epochs by I-MM, IG, and SG on the `covtype` training set for different ρ and φ.

Comparison with a standard one-vs-one SVM classifier

Similarly to Section 0.5.4, Table 19 reports the results of multiclass classification on the large-scale datasets `MNIST` and `covtype` using the same standard SVM classifier considered in Section 0.5.4. A comparison of the results in Tables 15 and 19 shows that the accuracy obtained with the standard SVM approach is slightly higher than that of I-MM for `MNIST`. However, the one-vs-one approach requires waiting for the entire execution to generate results. In contrast, I-MM is an iterative method that can be stopped after a predefined time budget, typically yielding good results even in a short time. For example, on the `MNIST` dataset, the results in Table 15 correspond to 100 epochs, taking approximately 3000 seconds (as inferred from Figures 6 and 7). Nevertheless, comparable accuracy can be obtained in the same time required by the considered standard SVM approach; for instance, the accuracy obtained by I-MM with the SH+HP objective function after 503 seconds is 0.9155. This behavior is further confirmed by the decay of the objective function in Figures 6 and 7, where after the first 500 seconds, progress toward the solution becomes very slow. For the `covtype` dataset, I-MM typically achieves comparable accuracy in significantly lower computational time (around 1500 seconds), as shown in Table 17 and Figure 8.

0.6 Enhancement of linear model performance

As shown in Section 0.5, the achieved accuracy is lower than that of non-linear or deep learning-based models. For instance, on `MNIST`, many models reach around 99% accuracy [37]. This suboptimal performance is not a result of the optimization algorithm, since its effectiveness was well established, but rather the inherent limitations of the linear classifier. As discussed in Section 0.1, while linear models

	Accuracy	Time (s)
`MNIST`	0.9438	513.85
`covtype`	0.5858	6004.53

Table 19: Performance of a standard ECOC one-vs-one SVM multiclass classifier on the large-scale datasets. Accuracy on the test set and computational time are reported.

benefit from having fewer parameters and higher explainability, their simplicity can ultimately restrict performance

A possible solution to improve accuracy while maintaining a linear classifier is to pre-structure the training dataset using various techniques. These methods serve as advanced feature extractors, capturing abstract and informative characteristics that enhance classification performance. Additionally, they can map data into latent spaces where different classes become more distinguishable, making the classification task easier for a linear model. In particular, we consider two different approaches.

Kernel Principal Component Analysis (Kernel PCA) This algorithm, introduced in [46], firstly maps each example $\mathbf{x}_k$ into a high-dimensional space using a kernel function. Then PCA is applied in this new space to reduce the number of features in the remapped dataset. In our experiment, we used a quadratic kernel and reduced the final number of features to 500.

CLIP image embeddings Recently, foundation models have demonstrated significant advantages in various computer vision applications [6, 57]. Specifically, the CLIP model [41, 48] is a visual-language model trained on a massive dataset containing both images and text. One of its key strengths is the ability to compute embeddings of input images, producing a meaningful representation with 512 new features. These features are computed exclusively exploiting the visual part of the pre-trained model in its inference phase.

As shown in the rest of this section, using one of the techniques described above in combination with a linear classifier can be more advantageous than applying a deep learning approach or a non-linear classifier. This is because it achieves very good accuracy with fewer parameters, significantly reducing computational costs.

In the following, we present the classification results obtained using a linear classifier on the `MNIST` dataset, remapped via either Kernel PCA or CLIP image embeddings. Specifically, for the remapped `MNIST` dataset, we solve the minimization problem (6) using the incremental MM method described in Algorithm 1. Regarding the objective function, the fidelity function ρ is set to logistic regression, while the regularizer φ is chosen as the hyperbolic potential, with $\lambda = 10^{-3}$, $\eta = 10^{-2}$, and

$\delta = 10^{-4}$. For the I-MM method, the initial parameter vector $\boldsymbol{\theta}^{(0)}$ is generated from a standard normal distribution, the step size γ_0 is set to 20 using the grid-search procedure, and the number of blocks m is set to 10.

Table 20 and Figure 10 compare the accuracy obtained using the original `MNIST` dataset and the datasets remapped via Kernel PCA and CLIP image embeddings. These results indicate that these techniques significantly improve classifier accuracy. In particular, with CLIP image embeddings, performance is fully comparable to many deep learning-based algorithms [37], while the number of parameters in the linear model remains limited to 5130, corresponding to the $Q(n+1)$ dimension. This is orders of magnitude lower than the number of parameters used in deep learning architectures. Considering an example of the architectures reported in[37], the Multi Layer Perceptron used in [1] achieves an accuracy of 0.9927 and consists of 1.12 million parameters. The proposed method thus efficiently leverage the availability of existing pretrained neural networks.

Finally, in Figure 10, we show the accuracy variation during the first ten epochs. It is clear that, especially for the remapped `MNIST` dataset, the classification task is efficiently solved at the very early stage of the optimization process.

	Original dataset	**Kernel PCA**	**CLIP embeddings**
Number of features	756	500	512
Total accuracy	0.9116	0.9552	**0.9902**

Table 20: Final accuracy on the original and remapped `MNIST` dataset obtained by solving problem (6) using the I-MM method over 100 epochs.

0.7 Conclusions

In this work, we introduced a novel optimization algorithm for training linear classifiers in multiclass classification tasks, with a particular focus on the Weston-Watkins SVM. By leveraging a Majorization-Minimization (MM) algorithm, we addressed the primal optimization problem associated with a regularized and smoothed version of the Weston-Watkins SVM loss. To enhance scalability and efficiency on large datasets, we introduced an incremental MM variant that incorporates second-order information through a half-quadratic majorization approach. We provided convergence guarantees for this algorithm, in both convex and non-convex settings. Our numerical experiments demonstrated the practical effectiveness of the proposed approach, showing that it achieves competitive performance while maintaining computational efficiency. Additionally, by employing either kernel principal component analysis or a foundation model on the training set, it can be observed that the optimized linear multiclass SVM can achieve results comparable to those of deep learning methods. As for future work, we will consider extending the proposed approach to loss functions that may include non-differentiable fidelity or regularization terms, for instance by incorporating incremental techniques into the MM-based accelerated proximal gradient methods proposed in [20].

acknowledgements

This work was partially supported by the Gruppo Nazionale per il Calcolo Scientifico (GNCS-INdAM). The publication was created with the co-financing of the European Union-FSE-REACT-EU, PON Research and Innovation 2014-2020 DM1062/2021.
F. P. is partially supported by the the Italian MUR through the PRIN 2022 Project "Numerical Optimization with Adaptive Accuracy and Applications to Machine Learning", project code: 2022N3ZNAX (CUP E53D23007700006), under the National Recovery and Resilience Plan (PNRR), Italy, Mission 04 Component 2 Investment 1.1 funded by the European Commission - NextGeneration EU programme.
F. P. is partially supported by the the Italian MUR through the PRIN 2022 PNRR Project "Advanced optimization METhods for automated central veIn Sign detection in multiple sclerosis from magneTic resonAnce imaging (AMETISTA)", project code: P2022J9SNP (CUP E53D23017980001), under the National Recovery and Resilience Plan (PNRR), Italy, Mission 04 Component 2 Investment 1.1 funded by the European Commission - NextGeneration EU programme.
E.C., F.C., and G.F. acknowledge support from the European Research Council Starting Grant MAJORIS ERC-2019-STG850925.

Declarations

Funding

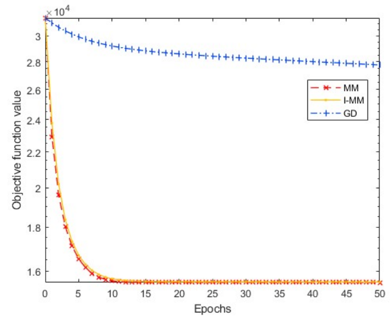
The authors did not receive support from any organization for the submitted work.

Data availabilty

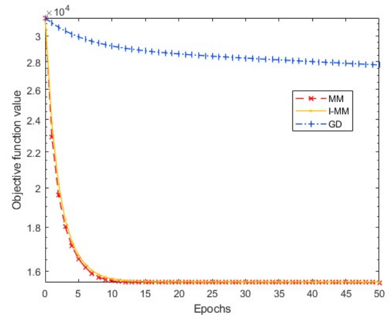
The code and data supporting the results of this work are available on the GitHub page `https://github.com/giofra23/MM_multiclass/settings`.

Conflict of interest

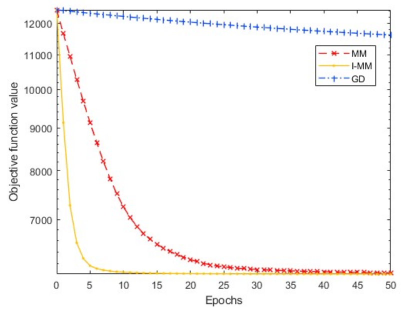
The authors declare that they have no conflict of interest.



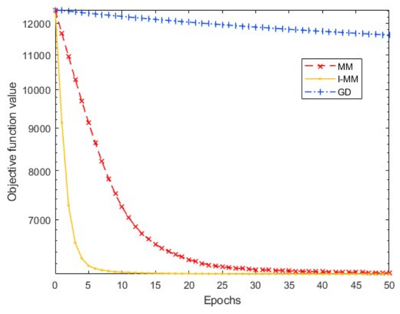
(a) SH + HP.



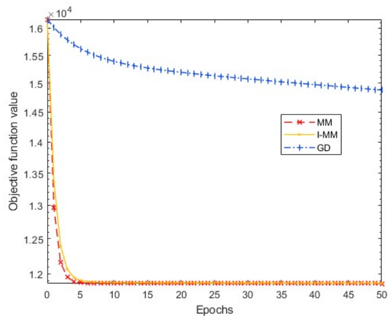
(b) SH + WP.



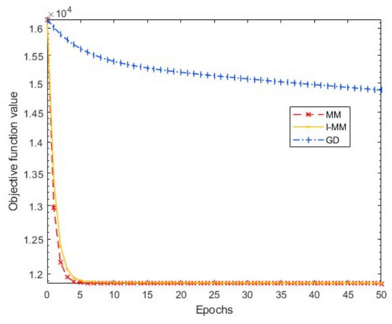
(c) SGM + HP.



(d) SGM + WP.

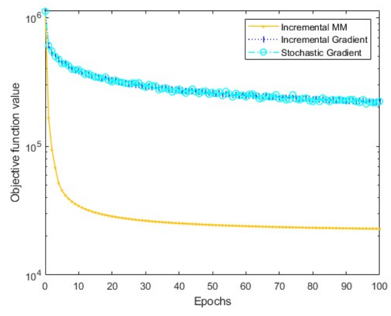


(e) LR + HP.

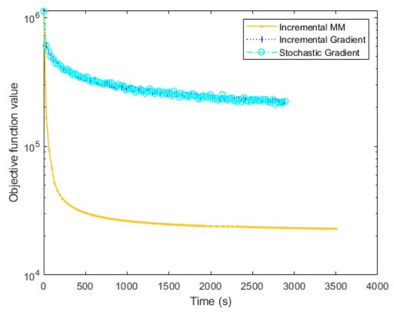


(f) LR + WP.

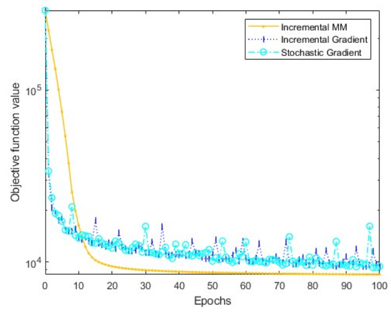
Figure 5: Decrease in the objective function achieved by I-MM, MM, and GD on the `protein` training set for different ρ and φ.



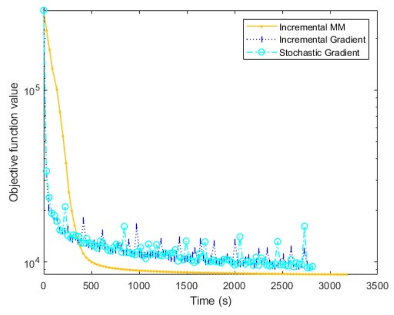
(a) SH + HP (epochs).



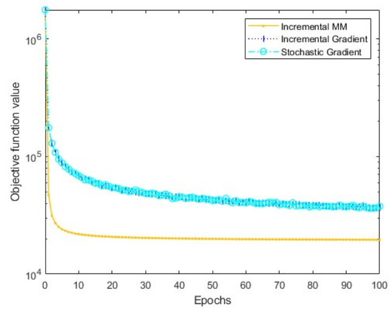
(b) SH + HP (time).



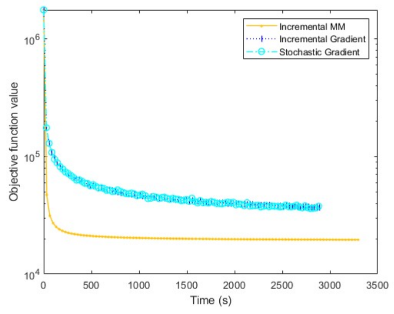
(c) SGM + HP (epochs).



(d) SGM + HP (time).

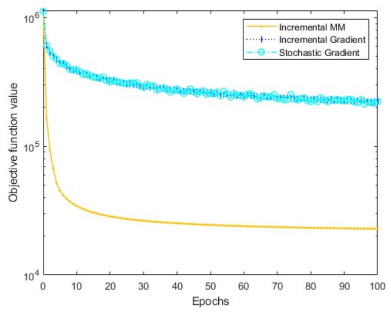


(e) LR + HP (epochs).

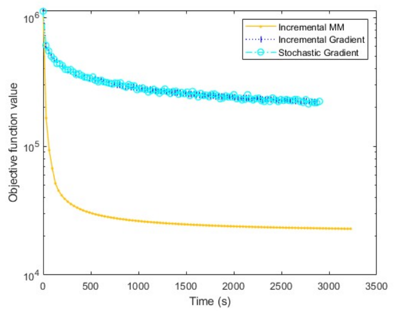


(f) LR + HP (time).

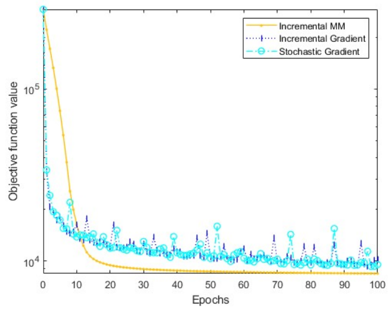
Figure 6: Objective function decrease achieved by I-MM, SG, and IG on the `MNIST` training set for the hyperbolic potential and different φ.



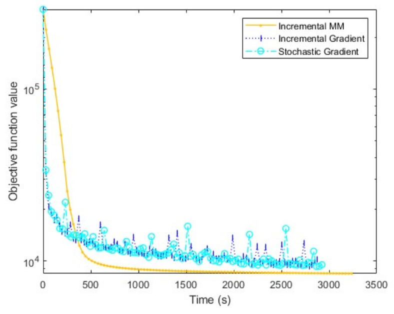
(a) SH + WP (epochs).



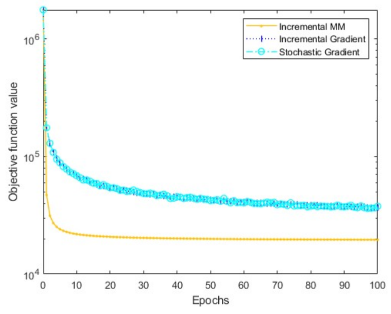
(b) SH + WP (time).



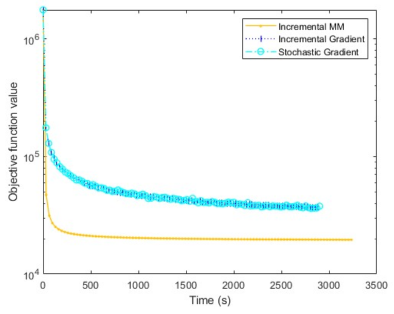
(c) SGM + WP (epochs).



(d) SGM + WP (time).

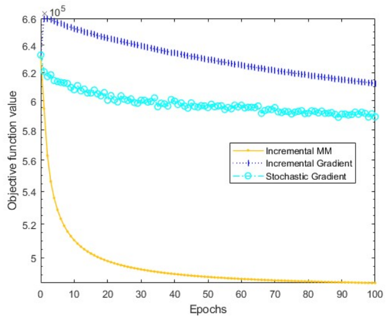


(e) LR + WP (epochs).

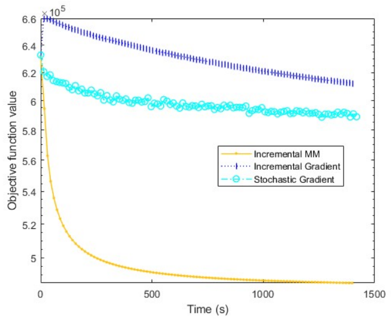


(f) LR + WP (time).

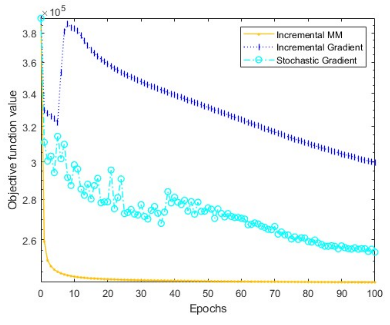
Figure 7: Objective function decrease achieved by I-MM, SG, and IG on the `MNIST` training set for the Welsh potential and different φ.



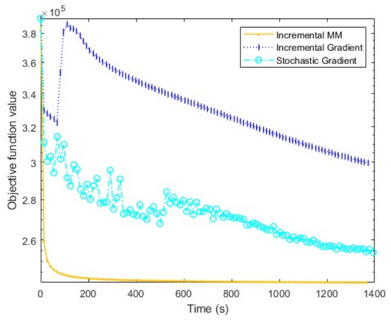
(a) SH + HP (epochs).



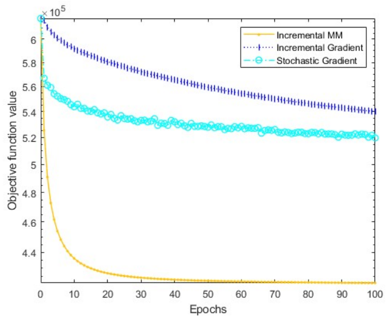
(b) SH + HP (time).



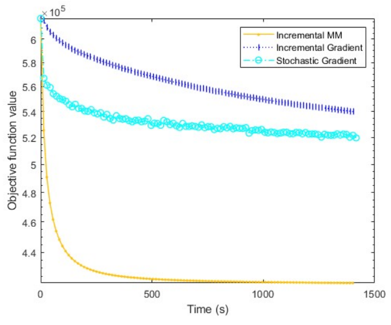
(c) SGM + HP (epochs).



(d) SGM + HP (time).

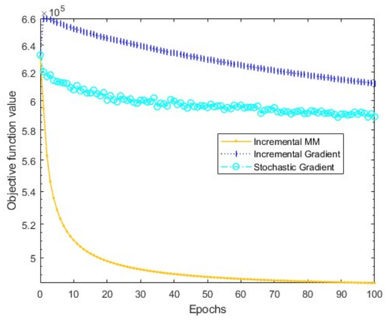


(e) LR + HP (epochs).

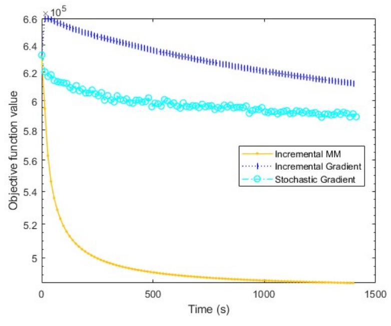


(f) LR + HP (time).

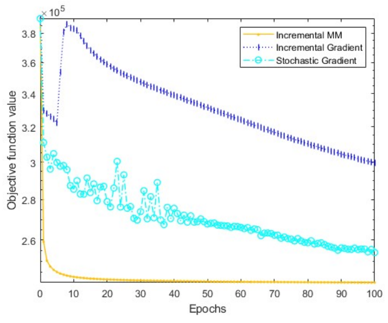
Figure 8: Objective function decrease achieved by I-MM, SG, and IG on the `covtype` training set for the hyperbolic potential and different φ.



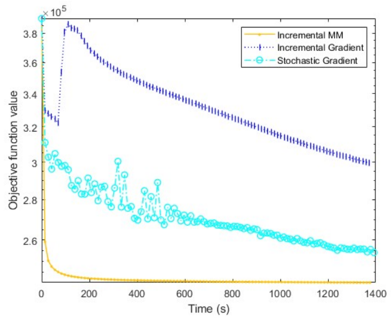
(a) SH + WP (epochs).



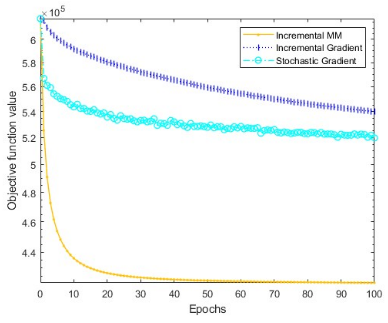
(b) SH + WP (time).



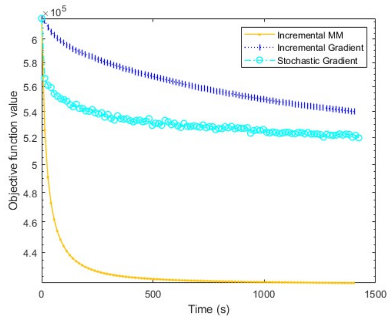
(c) SGM + WP (epochs).



(d) SGM + WP (time).



(e) LR + WP (epochs).



(f) LR + WP (time).

Figure 9: Objective function decrease achieved by I-MM, SG, and IG on the `covtype` training set for the Welsh potential and different φ.

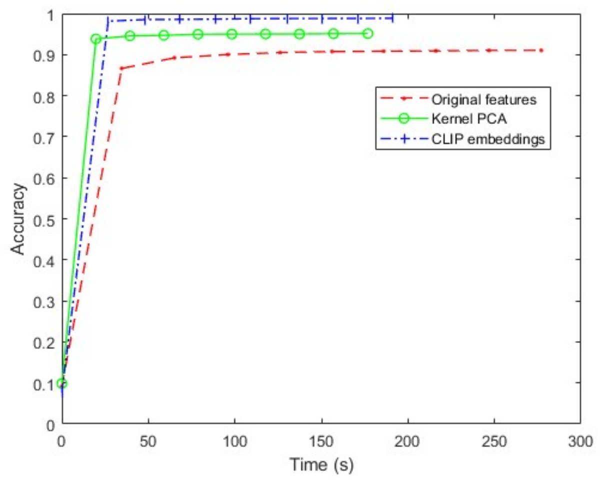


Figure 10: Accuracy behaviour on the original and remapped `MNIST` dataset obtained by solving problem (6) using the I-MM method

Bibliography

[1] A. Acharya, A. Hashemi, P. Jain, S. Sanghavi, I. S. Dhillon, and U. Topcu, *Robust training in high dimensions via block coordinate geometric median descent*, in Proceedings of The 25th International Conference on Artificial Intelligence and Statistics, vol. 151, 2022, pp. 11145–11168.

[2] M. Allain, J. Idier, and Y. Goussard, *On global and local convergence of half-quadratic algorithms*, IEEE Transactions on Image Processing, 15 (2006), pp. 1130–1142.

[3] E. Allwein, R. Schapire, and Y. Singer, *Reducing multiclass to binary: A unifying approach for margin classifiers*, Journal of Machine Learning Research, 1 (2000), pp. 113–141.

[4] H. Attouch, J. Bolte, P. Redont, and A. Soubeyran, *Proximal alternating minimization and projection methods for nonconvex problems: an approach based on the kurdyka- lojasiewicz inequality*, Math. Oper. Res., 35 (2010), pp. 438–457.

[5] H. Attouch, J. Bolte, and B. Svaiter, *Convergence of descent methods for semialgebraic and tame problems: proximal algorithms, forward backward splitting, and regularized Gauss-Seidel methods*, Mathematical Programming, (2011), pp. 1–39.

[6] M. Awais, M. Naseer, S. Khan, R. M. Anwer, H. Cholakkal, M. Shah, M.-H. Yang, and F. S. Khan, *Foundational models defining a new era in vision: A survey and outlook*, 2023, `https://arxiv.org/abs/2307.13721`.

[7] S. Bellavia, N. Krejić, B. Morini, and S. Rebegoldi, *A stochastic first-order trust-region method with inexact restoration for finite-sum minimization*, Comput. Optim. Appl., 84 (2023), pp. 53–84.

[8] A. Benfenati, E. Chouzenoux, G. Franchini, S. Latva-Äijö, D. Narnhofer, J.-C. Pesquet, S. J. Scott, and M. Yousefi, *Majoration-minimization for sparse SVMs*, in Advanced Techniques in Optimization for Machine Learning and Imaging, A. Benfenati, F. Porta, T. A. Bubba, and M. Viola, eds., Singapore, 2024, Springer Nature Singapore, pp. 31–54.

[9] D. P. Bertsekas and J. N. Tsitsiklis, *Gradient convergence in gradient methods with errors*, SIAM Journal on Optimization, 10 (2000), pp. 627–642.

[10] J. Bolte, S. Sabach, , and M. Teboulle., *Proximal alternating linearized minimization for nonconvex and nonsmooth problems*, Math. Program., 146 (2014), pp. 459–494.

[11] L. Bottou, F. E. Curtis, and J. Nocedal, *Optimization methods for large-scale machine learning*, SIAM Review, 60 (2018), pp. 223–311.

[12] L. Briceño-Arias, G. Chierchia, E. Chouzenoux, and J.-C. Pesquet, *A random block-coordinate Douglas-Rachford splitting method with low computational complexity for binary logistic regression*, Comput. Optim. Appl., 72 (2019), pp. 707–726.

[13] O. Chapelle, *Training a support vector machine in the primal*, Neural Computation, 19 (2007), pp. 1155–1178.

[14] 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), 2016, pp. 1–6, `https://doi.org/10.1109/MLSP.2016.7738833`.

[15] F. Chorobura and I. Necoara, *Random coordinate descent methods for nonseparable composite optimization*, SIAM Journal on Optimization, 33 (2023), pp. 2160–2190.

[16] E. Chouzenoux and J. B. Fest, *SABRINA: A stochastic subspace majorization-minimization algorithm*, Journal of Optimization Theory and Applications, 195 (2022), pp. 919–952.

[17] 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 (2013), pp. 563–591.

[18] E. Chouzenoux and J.-C. Pesquet, *A stochastic majorize-minimize subspace algorithm for online penalized least squares estimation*, IEEE Transactions on Signal Processing, 65 (2015), pp. 4770–4783.

[19] E. Chouzenoux and J.-C. Pesquet, *Optimization Methods for Signal Processing*, in Source Separation in Physical-Chemical Sensing, no. Chapter 2, IEEE Press, 2023.

[20] E. Chouzenoux, J.-C. Pesquet, and A. Repetti, *Variable metric forward-backward algorithm for minimizing the sum of a differentiable function and a convex function*, Journal of Optimization Theory and Applications, 162 (2014), pp. 107–132.

[21] R. Collobert, F. Sinz, J. Weston, and L. Bottou, *Large scale transductive SVMs*, Journal of Machine Learning Research, 7 (2006), pp. 1687–1712.

[22] A. Defazio, F. Bach, and S. Lacoste-Julien, *SAGA: A fast incremental gradient method with support for non-strongly convex composite objectives*, Advances in neural information processing systems, 27 (2014).

[23] U. Doğan, T. Glasmachers, and C. Igel, *A unified view on multi-class support vector classification*, The Journal of Machine Learning Research, 17 (2016), pp. 1550–1831.

[24] P. Felzenszwalb, R. Girshick, D. McAllester, and D. Ramanan, *Object detection with discriminatively trained part-based models*, IEEE Transactions on Pattern Analysis and Machine Intelligence, 32 (2010), pp. 1627–1645.

[25] G. Franchini, F. Porta, V. Ruggiero, and I. Trombini, *A line search based proximal stochastic gradient algorithm with dynamical variance reduction*, 2023.

[26] C.-W. Hsu and C.-J. Lin, *A comparison of methods for multiclass support vector machines*, IEEE Transactions on Neural Networks, 13 (2002), pp. 415–425.

[27] M. W. Jacobson and J. A. Fessler, *An expanded theoretical treatment of iteration-dependent Majorize-Minimize algorithms*, IEEE Transactions on Image Processing, 16 (2007), pp. 2411–2422.

[28] R. Johnson and T. Zhang, *Accelerating stochastic gradient descent using predictive variance reduction*, Advances in neural information processing systems, 26 (2013).

[29] S. S. Keerthi and D. M. DeCoste, *A modified finite Newton method for fast solution of large scale linear SVMs*, Journal of Machine Learning Research, 6 (2005), pp. 341–361.

[30] S. S. Keerthi, S. Sundararajan, K.-W. Chang, C.-J. Hsieh, and C.-J. Lin, *A sequential dual method for large scale multi-class linear SVMs*, in Proceedings of the 14th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, 2008, pp. 408–416.

[31] Y. J. Lee and O. L. Mangasarian, *SSVM: A smooth support vector machine for classification*, Computational Optimization and Applications, 20 (2001), pp. 5–22.

[32] J. Mairal, *Stochastic majorization-minimization algorithms for large-scale optimization*, in Proceedings of the 27th International Conference on Neural Information Processing Systems - Volume 2, NIPS'13, Red Hook, NY, USA, 2013, Curran Associates Inc., p. 2283–2291.

[33] J. Mairal, *Incremental majorization-minimization optimization with application to large-scale machine learning*, SIAM J. Optim., 25 (2014), pp. 829–855.

[34] O. L. Mangasarian, *A finite Newton method for classification*, Optimization Methods and Software, 17 (2002), pp. 913–929.

[35] S. Martin, Y. Huang, F. Shakeri, J.-C. Pesquet, and I. B. Ayed, *Transductive zero-shot and few-shot CLIP*, 2024, https://arxiv.org/abs/2405.18437.

[36] L. M. Nguyen, J. Liu, K. Scheinberg, and M. Takáč, *SARAH: A novel method for machine learning problems using stochastic recursive gradient*, in International Conference on Machine Learning, 2017, pp. 2613–2621.

[37] Papers With Code web site, 2022, https://paperswithcode.com/sota/image-classification-on-mnist.

[38] S. N. Parizi, K. He, R. Aghajani, S. Sclaroff, and P. Felzenszwalb, *Generalized majorization-minimization*, in Proceedings of the 36th International Conference on Machine Learning, K. Chaudhuri and R. Salakhutdinov, eds., vol. 97 of Proceedings of Machine Learning Research, PMLR, 09–15 Jun 2019, pp. 5022–5031.

[39] A. Patrascu and I. Necoara, *Efficient random coordinate descent algorithms for large-scale structured nonconvex optimization*, J. Glob. Optim., 61 (2015), pp. 19–46.

[40] D. N. Phan, S. Bartz, N. Guha, and H. M. Phan, *Stochastic variance-reduced majorization-minimization algorithms*, SIAM Journal on Mathematics of Data Science, 6 (2024), pp. 926–952.

[41] A. Radford, J. W. Kim, C. Hallacy, A. Ramesh, G. Goh, S. Agarwal, G. Sastry, A. Askell, P. Mishkin, J. Clark, G. Krueger, and I. Sutskever, *Learning transferable visual models from natural language supervision*, 2021, https://arxiv.org/abs/2103.00020.

[42] A. Repetti and Y. Wiaux, *Variable metric forward-backward algorithm for composite minimization problems*, SIAM Journal on Optimization, 31 (2021), pp. 1215–1241.

[43] R. T. Rockafeller, *Convex Analysis*, Princeton University Press, Princeton, NJ, 1970.

[44] L. Rosasco, S. Villa, and B. Vũ, *Stochastic forward–backward splitting for monotone inclusions*, J. Optim. Theory Appl., 169 (2016), pp. 388–406.

[45] S. Salzo and S. Villa, *Parallel random block-coordinate forward–backward algorithm: a unified convergence analysis*, Math. Program., 193 (2022), pp. 225–269.

[46] B. Schölkopf, A. Smola, and K.-R. Müller, *Kernel principal component analysis*, in Artificial Neural Networks — ICANN'97, Berlin, Heidelberg, 1997, Springer Berlin Heidelberg, pp. 583–588.

[47] S. Shalev-Shwartz, Y. Singer, N. Srebro, and A. Cotter, *Pegasos: Primal estimated sub-gradient solver for SVM*, Mathematical Programming, 127 (2011), pp. 3–30.

[48] H. Song, L. Dong, W.-N. Zhang, T. Liu, and F. Wei, *CLIP models are few-shot learners: Empirical studies on vqa and visual entailment*, 2022, `https://arxiv.org/abs/2203.07190`.

[49] B. Sriperumbudur, D. Torres, and G. Lanckriet, *A majorization-minimization approach to the sparse generalized eigenvalue problem*, Machine Learning, 85 (2011), pp. 3–39.

[50] Y. Sun, P. Babu, and D. P. Palomar, *Majorization-minimization algorithms in signal processing, communications, and machine learning*, IEEE Transactions on Signal Processing, 65 (2017), pp. 794–816.

[51] L. Wang, J. Zhu, and H. Zou, *The doubly regularized support vector machine*, Statistica Sinica, 16 (2006), pp. 589–615.

[52] Y. Wang and C. Scott, *Weston-Watkins hinge loss and ordered partitions*, in Advances in Neural Information Processing Systems, H. Larochelle, M. Ranzato, R. Hadsell, M. Balcan, and H. Lin, eds., vol. 33, Curran Associates, Inc., 2020, pp. 19873–19883.

[53] Y. Wang and C. Scott, *An exact solver for the Weston-Watkins SVM subproblem*, in International Conference on Machine Learning, PMLR, 2021, pp. 10894–10904.

[54] J. Weston and C. Watkins, *Support vector machines for multi-class pattern recognition*, in Proc. 7th European Symposium on Artificial Neural Networks, 1999.

[55] J. Xu and K. Lange, *Power k-means clustering*, in Proceedings of the 36th International Conference on Machine Learning, PMLR, 2019.

[56] Y. Xu and W. Yin, *A block coordinate descent method for regularized multiconvex optimization with applications to nonnegative tensor factorization and completion*, SIAM J. Imaging Sci., 6 (2013), pp. 1758–1789.

[57] S. Zhang and D. Metaxas, *On the challenges and perspectives of foundation models for medical image analysis*, 2023, `https://arxiv.org/abs/2306.05705`.

[58] Z. Zhang, J. Kwok, and D. Yeung, *Surrogate maximization/minimization algorithms and extensions*, Machine Learning, 69 (2007), pp. 1–33.

[59] A. Zien, F. De Bona, and C. S. Ong, *Training and approximation of a primal multiclass support vector machine*, in Proceedings of the 12th International Conference on Applied Stochastic Models and Data Analysis (ASMDA 2007), 2007, pp. 1–8.

[60] H. Zou and T. Hastie, *Regularization and variable selection via the elastic net*, Journal of the Royal Statistical Society Series B: Statistical Methodology, 67 (2005), pp. 301–320.

.1 Proof of Proposition 0.2

This proof relies on the notation introduced in Subsection 0.2.3 for the matrix formulation of the objective function.

As a first step, we rewrite the objective function given in (13) as the sum of three terms:

$$\Phi(\boldsymbol{\theta}) = \Phi_1(\boldsymbol{\theta}) + \Phi_2(\boldsymbol{\theta}) + \Phi_3(\boldsymbol{\theta}),$$

where

$$\begin{aligned}\Phi_1(\boldsymbol{\theta}) &= \mathcal{L}_{WW}(\boldsymbol{\theta}) = g_K(\mathbf{L}\boldsymbol{\theta}) - K\rho(0),\\ \Phi_2(\boldsymbol{\theta}) &= \frac{\eta}{2}\|\boldsymbol{\theta}_w\|_2^2,\\ \Phi_3(\boldsymbol{\theta}) &= \lambda\sum_{i=1}^{Qn}\varphi\left((\boldsymbol{\theta}_w)_i\right).\end{aligned}$$

We now show that each function Φ_i, for $i \in \{1,2,3\}$, admits a quadratic majorant $h_i(\cdot;\boldsymbol{\theta}')$ at $\boldsymbol{\theta}' \in \mathbb{R}^{Q(n+1)}$ of the form:

$$h_i(\boldsymbol{\theta};\boldsymbol{\theta}') = \Phi_i(\boldsymbol{\theta}') + \nabla\Phi_i(\boldsymbol{\theta}')^\top(\boldsymbol{\theta}-\boldsymbol{\theta}') + \frac{1}{2}(\boldsymbol{\theta}-\boldsymbol{\theta}')^\top\mathbf{A}_i(\boldsymbol{\theta}')(\boldsymbol{\theta}-\boldsymbol{\theta}').$$

(i) Quadratic majorant for Φ_1. First, we derive an upper bound for the function $g_K(\mathbf{L}\boldsymbol{\theta})$.

$$\begin{aligned}g_K(\mathbf{L}\boldsymbol{\theta}) &= \sum_{k=1}^{K}\sum_{q=1}^{Q}\rho\left((\mathbf{L}\boldsymbol{\theta})_{q,k}\right) \qquad (36)\\ &\le \sum_{k=1}^{K}\sum_{q=1}^{Q}\left(\rho\left((\mathbf{L}\boldsymbol{\theta}')_{q,k}\right) + \rho'\left((\mathbf{L}\boldsymbol{\theta}')_{q,k}\right)(\mathbf{L}(\boldsymbol{\theta}-\boldsymbol{\theta}'))_{q,k} + \frac{\beta}{2}(\mathbf{L}(\boldsymbol{\theta}-\boldsymbol{\theta}'))^2_{q,k}\right)\\ &= g_K(\mathbf{L}\boldsymbol{\theta}') + \left(\mathbf{L}^\top\nabla g(\mathbf{L}\boldsymbol{\theta})\right)^\top(\boldsymbol{\theta}-\boldsymbol{\theta}') + \frac{\beta}{2}(\boldsymbol{\theta}-\boldsymbol{\theta}')^\top\mathbf{L}^\top\mathbf{L}(\boldsymbol{\theta}-\boldsymbol{\theta}')\\ &= g_K(\mathbf{L}\boldsymbol{\theta}') + \nabla\Phi_1(\boldsymbol{\theta}')^\top(\boldsymbol{\theta}-\boldsymbol{\theta}') + \frac{\beta}{2}(\boldsymbol{\theta}-\boldsymbol{\theta}')^\top\mathbf{L}^\top\mathbf{L}(\boldsymbol{\theta}-\boldsymbol{\theta}')\end{aligned}$$

where the first equality follows from the definition of g_K, the first inequality from the β-Lipschitz smoothness property of ρ, and the third equality from (16).

From (36), if we define $\mathbf{A}_1(\boldsymbol{\theta}') = \beta \mathbf{L}^\top \mathbf{L}$, it follows that the function

$$h_1(\boldsymbol{\theta};\boldsymbol{\theta}') = \Phi_1(\boldsymbol{\theta}') + \nabla\Phi_1(\boldsymbol{\theta}')^\top(\boldsymbol{\theta}-\boldsymbol{\theta}') + \frac{1}{2}(\boldsymbol{\theta}-\boldsymbol{\theta}')^\top \mathbf{A}_1(\boldsymbol{\theta}')(\boldsymbol{\theta}-\boldsymbol{\theta}')$$

is a tangent majorant for Φ_1 at $\boldsymbol{\theta}'$.

(ii) Quadratic majorant for Φ_2. Given any $\varepsilon > 0$, we have

$$\nabla^2\Phi_2(\boldsymbol{\theta}) = \eta \begin{pmatrix} \mathbf{I}_{Qn} & \mathbf{O}_{Qn\times Q} \\ \mathbf{O}_{Q\times Qn} & \mathbf{O}_{Q\times Q} \end{pmatrix} \leq \begin{pmatrix} \eta\mathbf{I}_{Qn} & \mathbf{O}_{Qn\times Q} \\ \mathbf{O}_{Q\times Qn} & \varepsilon\mathbf{I}_{Q} \end{pmatrix} := \mathbf{A}_2,$$

where $\mathbf{O}_{m\times n}$ is a zero matrix of m rows and n columns. According to [19, Property 1.3 (ii)], by defining $\mathbf{A}_2(\boldsymbol{\theta}') = \mathbf{A}_2$, the function

$$h_2(\boldsymbol{\theta};\boldsymbol{\theta}') = \Phi_2(\boldsymbol{\theta}') + \nabla\Phi_2(\boldsymbol{\theta}')^\top(\boldsymbol{\theta}-\boldsymbol{\theta}') + \frac{1}{2}(\boldsymbol{\theta}-\boldsymbol{\theta}')^\top \mathbf{A}_2(\boldsymbol{\theta}')(\boldsymbol{\theta}-\boldsymbol{\theta}')$$

is a tangent majorant for Φ_2 at $\boldsymbol{\theta}'$.

(iii) Quadratic majorant for Φ_3. According to [19, Property 1.7], and noting that the function $\varphi : \mathbb{R} \to \mathbb{R}$ not only satisfies the assumptions of this property but is also even, it follows that, for every $v' \in \mathbb{R}$,

$$(\forall v \in \mathbb{R}) \quad \varphi(v) \leq \varphi(v') + \varphi'(v')(v-v') + \frac{1}{2}\frac{\varphi'(v')}{v'}(v-v')^2.$$

Then, by defining $\psi : v \mapsto \varphi'(v)/v$, we have

$$\begin{aligned}
\Phi_3(\boldsymbol{\theta}) &= \lambda \sum_{i=1}^{Qn} \varphi\left((\boldsymbol{\theta}_w)_i\right) \\
&\leq \lambda \sum_{i=1}^{Qn} \left(\varphi((\boldsymbol{\theta}'_w)_i) + \varphi'((\boldsymbol{\theta}'_w)_i)((\boldsymbol{\theta}_w)_i - (\boldsymbol{\theta}'_w)_i) + \frac{1}{2}\frac{\varphi'((\boldsymbol{\theta}'_w)_i)}{(\boldsymbol{\theta}'_w)_i}((\boldsymbol{\theta}_w)_i - (\boldsymbol{\theta}'_w)_i)^2\right) \\
&= \lambda \sum_{i=1}^{Qn} \left(\varphi((\boldsymbol{\theta}'_w)_i) + \varphi'((\boldsymbol{\theta}'_w)_i)((\boldsymbol{\theta}_w)_i - (\boldsymbol{\theta}'_w)_i) + \frac{1}{2}\psi((\boldsymbol{\theta}'_w)_i)((\boldsymbol{\theta}_w)_i - (\boldsymbol{\theta}'_w)_i)^2\right) \\
&= \Phi_3(\boldsymbol{\theta}') + \nabla\Phi_3(\boldsymbol{\theta}')^T(\boldsymbol{\theta}-\boldsymbol{\theta}') + \frac{1}{2}(\boldsymbol{\theta}-\boldsymbol{\theta}')^T\mathbf{A}_3(\boldsymbol{\theta}')(\boldsymbol{\theta}-\boldsymbol{\theta}') := h_3(\boldsymbol{\theta};\boldsymbol{\theta}'),
\end{aligned}$$

where, for each $\boldsymbol{\theta} \in \mathbb{R}^{Q(n+1)}$,

$$\boldsymbol{\theta} = \begin{bmatrix} \boldsymbol{\theta}_w \\ \boldsymbol{\theta}_b \end{bmatrix} = \begin{bmatrix} \boldsymbol{\theta}_1 \\ \vdots \\ \boldsymbol{\theta}_{Qn} \\ \boldsymbol{\theta}_{Qn+1} \\ \vdots \\ \boldsymbol{\theta}_{Q(n+1)} \end{bmatrix},$$

$$\nabla\Phi_3(\boldsymbol{\theta}) = \lambda \begin{bmatrix} \varphi'(\boldsymbol{\theta}_1) \\ \varphi'(\boldsymbol{\theta}_2) \\ \vdots \\ \varphi'(\boldsymbol{\theta}_{Qn}) \\ 0 \\ \vdots \\ 0 \end{bmatrix},$$

and

$$\mathbf{A}_3(\boldsymbol{\theta}) = \mathrm{Diag}\left(\begin{bmatrix} \lambda\psi(\boldsymbol{\theta}_1) \\ \lambda\psi(\boldsymbol{\theta}_2) \\ \vdots \\ \lambda\psi(\boldsymbol{\theta}_{Qn}) \\ 0 \\ \vdots \\ 0 \end{bmatrix}\right).$$

As a consequence, by definition, $h_3(\cdot;\boldsymbol{\theta}')$ is a tangent majorant for Φ_3 at $\boldsymbol{\theta}'$.

The result now follows by defining the tangent majorant $h(\cdot;\boldsymbol{\theta}')$ of Φ at $\boldsymbol{\theta}' \in \mathbb{R}^{Q(n+1)}$ as the sum

$$\begin{aligned}(\forall \boldsymbol{\theta} \in \mathbb{R}^{Q(n+1)}) \quad h(\boldsymbol{\theta};\boldsymbol{\theta}') &= h_1(\boldsymbol{\theta};\boldsymbol{\theta}') + h_2(\boldsymbol{\theta};\boldsymbol{\theta}') + h_3(\boldsymbol{\theta};\boldsymbol{\theta}') \\ &= \Phi(\boldsymbol{\theta}') + \nabla\Phi(\boldsymbol{\theta}')^\top(\boldsymbol{\theta}-\boldsymbol{\theta}') + \frac{1}{2}(\boldsymbol{\theta}-\boldsymbol{\theta}')^\top\mathbf{A}(\boldsymbol{\theta}')(\boldsymbol{\theta}-\boldsymbol{\theta}'),\end{aligned}$$

where the matrix $A(\boldsymbol{\theta})$ is given by

$$A(\boldsymbol{\theta}) = A_1(\boldsymbol{\theta}) + A_2(\boldsymbol{\theta}) + A_3(\boldsymbol{\theta})$$

and coincides with the expression provided in (21).