Parallel Majorization Minimization with Dynamically Restricted Domains for Nonconvex Optimization

Yan Kaganovsky
Duke University
Ikenna Odinaka
Duke University
David Carlson
Columbia University
Lawrence Carin
Duke University

Abstract

We propose an optimization framework for nonconvex problems based on majorization-minimization that is particularly well-suited for parallel computing. It reduces the optimization of a high dimensional nonconvex objective function to successive optimizations of locally tight and convex upper bounds which are additively separable into low dimensional objectives. The original problem is then broken into simpler and parallel tasks, while guaranteeing the monotonic reduction of the original objective function and convergence to a local minimum. This framework also allows one to restrict the upper bound to a local dynamic convex domain, so that the bound is better matched to the local curvature of the objective function, resulting in accelerated convergence. We test the proposed framework on a nonconvex support vector machine based on a sigmoid loss function and on nonconvex logistic regression.

1 Introduction

We consider the following optimization problem

$$\theta^* = \arg \min_{\theta \in \Omega_F} C(\theta),$$

(1)

where $\Omega_F$ is a closed convex set and $C$ is generally non-convex and has the following form

$$C(\theta) = \sum_{m=1}^{M} F(\theta; x_m, y_m) + \lambda R(\theta)$$

$$= \sum_{m=1}^{M} f_m(\theta^T x_m) + \lambda R(\theta),$$

(2)

where $F(\theta; x_m, y_m)$ represents the data-mismatch for the $m$th data point $(x_m, y_m) \in \mathbb{R}^p \times \mathbb{R}^d$ and $\theta \in \mathbb{R}^p$ are the parameters (latent variables) to be learned. $f_m$ is a twice continuously differentiable function $\mathbb{R} \to \mathbb{R}$ (possibly nonconvex) that depends on $y_m$. Here $R$ represents a penalty due to prior knowledge (e.g., the 2-norm penalty) that is separable and twice continuously differentiable (possibly non-convex). The class of optimization problems in (1) appears in various fields such as machine learning, signal/image processing, imaging and bioinformatics, to name a few.

Often, these problems can be of huge size (large $p$) and involve big datasets (large $M$). There is a growing need for faster algorithms that can deal with large problems in a numerically efficient way. The availability of high performance multi-core computing platforms makes it increasingly desirable to develop parallel optimization methods.

A general optimization principle that often leads to parallelizable algorithms is majorization-minimization (MM) (Lange, 2000), also known as optimization transfer. The idea is essentially to replace a difficult optimization problem by a sequence of simpler problems. At each iteration, the original objective function is approximated by a locally tight upper bound surrogate so that minimizing the surrogate ensures that the objective function is decreased. The motivation for using MM can be to simplify the optimization process, make it more suitable for parallel computing, or both. The global convergence of this type of algorithms has been established for several cases (Lanckriet and Sriperumbudur, 2009; Mairal, 2015; Ahn et al., 2006; Jacobson and Fessler, 2007). Various well known approaches can be interpreted from an MM point of view, such as expectation-maximization (EM) algorithms in statistics (Dempster, 1977; Neal and Hinton, 1998) and difference-of-convex programming (Lanckriet and Sriperumbudur, 2009). The MM procedure has been successfully used in tomographic imaging (Ahn et al., 2006; Kaganovsky et al., 2015), image processing (Sot-thivirat and Fessler, 2002), and matrix factorization (Lee and Seung, 2001), to name a few.
A related principle is *Successive Convex Approximations (SCA)* (Razaviyayn et al., 2013; Razaviyayn et al., 2014), where as opposed to MM, the surrogate is not necessarily a global upper bound of the objective so that the monotonic reduction of the objective is not guaranteed. However, due to additional assumptions such as Lipchitz smoothness, convergence can be guaranteed if the step sizes are chosen to satisfy certain criteria related to the Lipchitz constant of the gradient. Well known examples that use this approach are gradient-based proximal methods (Beck and Teboulle, 2009; Combettes and Pesquet, 2011). Recently, Razaviyayn et al. (2014) proposed an SCA method for non-convex optimization where one updates blocks of variables *in parallel* (PSCA) while still guaranteeing convergence.

A class of parallelizable MM algorithms using partitioned-separable paraboloidal surrogates for non-convex optimization was proposed by Erdogan and Fessler (1999) and also Sothivirat and Fessler (2002). In this approach, the objective is approximated by a high-dimensional quadratic global upper bound and then an additional upper bound is created by using Jensen’s inequality to obtain a block coordinate descent algorithm. This approach allows one to update blocks of variables *in parallel* while still guaranteeing the monotonic decrease of the original objective and also convergence. A limitation of this method is that it relies on positivity constraints to construct the quadratic upper bound.

The contributions of this work are as follows:

- We propose a new type of parallelizable local majorization-minimization (MM) for nonconvex optimization which uses successive optimizations of convex surrogates. The domain of the surrogate is controlled at each iteration to better match it to the local curvature of the objective function, resulting in accelerated convergence.

- Similar to Erdogan and Fessler (1999) and Sothivirat and Fessler (2002), and in contrast to Razaviyayn (2014), our method guarantees monotonic decrease of the objective function. This has the advantage that deviation from monotonicity serves as indication of error in the code.

- In contrast to Erdogan and Fessler (1999), Sothivirat and Fessler (2002), and Razaviyayn (2014), our method uses surrogates that are matched to a local compact region of the objective function rather than globally. Also, in contrast to the first two methods, our method does not require positivity constraints.

- The proposed method leads to parallel block coordinate descent algorithms, where blocks are updated simultaneously on different processors while still automatically guaranteeing the reduction of the original objective function. Each of the blocks can be reduced to a small enough size such that second-order methods are practical and can be used to accelerate convergence.

- The proposed method can be used with an arbitrary block size. This enables more flexibility to better explore the trade-off between convergence rate per iteration, and the time per iteration (including parallization costs such as communication between different processors), at the price of a more restrictive class of problems than Razaviyayn (2014). Note that the method by Razaviyayn (2014) is practically limited to small block sizes since it requires computing the lowest eigenvalue of the feature/system sub-matrix corresponding to each block, which becomes computationally expensive as the block size increases.

- We derive guarantees for global convergence of the proposed class of algorithms.

- We test the proposed framework on nonconvex support vector machines based on a sigmoid loss function (similar to the “ramp-loss” used by Ertekin et al. (2011)) and on logistic regression with nonconvex penalties (Mairal, 2015).

- The proposed method is shown to yield comparable or superior classification accuracy relative to many prior methods for non-convex optimization such as PSCA, and popular stochastic optimization methods, e.g., AdaGrad, and RMSProp.

## 2 Majorization Minimization with Dynamically Restricted Domains

### 2.1 Overview of Majorization Minimization

We begin with some basic definitions of the majorization-minimization procedure (MMP) that will be considered in this paper. Unless otherwise stated, all functions are $\mathbb{R}^p \to \mathbb{R}$.

**Definition** $S(\theta; \hat{\theta}, \Omega_M)$ is said to be a local first-order surrogate of a function $\mathcal{C}$ around an expansion point $\hat{\theta}$ if the following conditions are satisfied

\[
S(\theta; \hat{\theta}) \geq \mathcal{C}(\theta) \quad \forall \theta \in \Omega_M, \quad (3)
\]

\[
S(\hat{\theta}; \hat{\theta}) = \mathcal{C}(\hat{\theta}), \quad (4)
\]

\[
\nabla_\theta S|_{\theta=\hat{\theta}} = \nabla_\theta \mathcal{C}|_{\theta=\hat{\theta}}, \quad (5)
\]

\[
\Omega_M \cap \Omega_F \neq \emptyset, \quad (6)
\]

\[
\hat{\theta} \in \text{Int}(\Omega_M), \quad (7)
\]
where $\Omega_M$ is called the majorization domain, $\text{Int}(\cdot)$ denotes the interior of a set and $\Omega_F$ is the feasible set in the original problem in (1). If $S(\theta; \hat{\theta})$ is also convex with respect to $\theta$ on some convex set containing $\Omega_M$, it is said to be a \textit{convex local first-order surrogate}.

The condition in (3) states that $S$ is an upper bound of $C$ in $\Omega_M$. The conditions in (4)–(5) state consistency of the values of the functions and their gradients at the expansion point $\hat{\theta}$. Condition (6) is added to ensure that a feasible solution exists. The condition in (7) ensures that $\hat{\theta}$ is in $\Omega_M$ and not a boundary point.

The basic idea in MMP is to replace the hard problem in (1) by a sequence of easier subproblems

$$\theta^{(t+1)} = \arg\min_{\theta} \{ S(\theta; \theta^{(t)}; \Omega_M) : \theta \in \Omega_F \cap \Omega_M \},$$

where $S(\theta; \hat{\theta}, \Omega_M)$ is the surrogate for the cost function $C$ in (1) with the expansion point $\hat{\theta}$ chosen according to the last iteration $\hat{\theta} = \theta^{(t)}$ and the majorization domain $\Omega_M = \Omega_M^{(t)}$ can also change with iteration. Also note that we do not require $S$ to be strongly convex, so it can have multiple minima; accordingly, in the definition of (8) we use $\leq$ instead of equality. This MM procedure is guaranteed to decrease the original objective function (or leave it unchanged) if the surrogate is decreased (or unchanged), since

$$C(\theta^{(t+1)}) \leq S(\theta^{(t+1)}; \theta^{(t)}) \leq S(\theta^{(t)}; \theta^{(t)}) = C(\theta^{(t)}),$$

where the first inequality follows from (3), the second inequality follows from (8) and the equality follows from (4). We will also show in Sec. 3 that under some mild conditions, this procedure is guaranteed to converge to a local minimum of $C$.

The definition in (3)–(7) includes, as a special case, the paraboloidal surrogates used by Erdogan and Fessler (1999) and also Sotthivirat and Fessler (2002), where the feasible set is assumed to be $\Omega_F = \mathbb{R}^p_+$, the majorization is on the entire feasible region $\Omega_M = \Omega_F$ and the surrogate $S$ is restricted to a particular quadratic form. However, to the best of our knowledge, the possibility of changing the majorization domains s.t. $\Omega_M \subset \Omega_F$ has not been significantly explored. It should be noted that this idea has been mentioned by Jacobson and Fessler (2007), albeit without any specific algorithm that actually uses this approach. As we shall show below, there are important practical advantages to using \textit{local} convex surrogates with dynamic majorization domains.

### 2.2 Convex Local Majorization

The first step of our MM procedure is to construct a convex \textit{local} first-order surrogate to the nonconvex cost function by utilizing the specific structure in (2). Specifically, our approach relies on the observation that in most models of the form in (2) the scalar functions $f_m$ and $R$ and their derivatives are given in closed-form. Thus, we first construct a convex surrogate for the scalar functions $f_m(z)$, where later we will substitute $z = \theta^T x_m$. We note that since the set of functions $f_m$ generally depend on $y_m$ they can be different for different $m$.

Let $f_m : \mathbb{R} \rightarrow \mathbb{R}$ be a twice continuously differentiable nonconvex function with its second derivative bounded below, i.e.,

$$\min f_m''(z) > -\infty \quad \forall m,$$

where $f_m''$ is the second derivative of $f_m$. Define the minimum second derivative on the interval $[a_m, b_m]$ as

$$c_m(a_m, b_m) := \min_{a_m \leq z \leq b_m} f_m''(z).$$

We now define $\bar{f}_m$ as

$$\bar{f}_m(z; \hat{\theta}) := f_m(z) + [-c_m(a_m, b_m)](z - \hat{\theta})^2 / 2,$$

where $[x]_+ = \max(0, x)$. It is easy to verify that for any particular $m$, the function $\bar{f}_m(z; \hat{\theta})$ in (12) is a global first-order surrogate for $f_m(z)$ (see definition in Sec. 2.1), i.e., $\bar{f}_m(z; \hat{\theta}) \geq f_m(z)$ with equality at $z = \hat{\theta}$ and $f_m'' = \bar{f}_m''$ at $z = \hat{\theta}$. Also, convexity, i.e., $\bar{f}_m'' \geq 0$, is guaranteed only on the domain $[a_m, b_m]$; so in accordance with the definitions in Sec. 2.1, to obtain a convex surrogate the majorization domain should be chosen as $\Omega_M = \{ \theta | a_m \leq \theta^T x_m \leq b_m, \forall m \}$. Note that for $(a_m, b_m) = (-\infty, \infty)$, $[-c_m(a_m, b_m)]$ in (12) equals the absolute value of the global minimum of $f_m''$ if the minimum is negative and zero otherwise, so $\bar{f}_m$ in (12) is convex on the entire real line. Figure 1 illustrates the need for the local surrogate introduced in (12) and why one should consider choosing $a_m > -\infty$ and $b_m < \infty$. When the expansion point $\hat{\theta}$ is located at a convex region ($f_m'' > 0$), adding $[-\min_{z \in \mathbb{R}} f_m''(z)](z - \hat{\theta})^2$ to the function $f_m(z)$ can lead to very high positive curvature of the surrogate and to small steps when minimizing the surrogate using second-order methods. In order to achieve faster convergence, we restrict the surrogate to the interval $[a_m, b_m]$ and choose $c_m$ in (11) according to the local minimum of $f_m''$ (if it is negative), which can result in a much lower curvature of the surrogate. The proposed modification leads to wider surrogates which enable taking larger steps (if needed) when using second-order methods, as illustrated by the red curve in Fig. 1. In fact, if $[a_m, b_m]$ is contained in a convex region, then the quadratic term in (12) is dropped and the original function can be used in this region.

To appreciate the difficulty in choosing the majorization domains $\Omega_M$ or $[a_m, b_m]$, note that the Hessian
for each $m$ to satisfy a multivariate convex surrogate to a function of the form of the nonconvex objective in (2) with respect to $\theta$ has the form $H(\theta) = X\Lambda_1 X^T + \lambda \Lambda_2$, where $X \in \mathbb{R}^{p \times M}$ is the column-wise concatenation of the vectors $x_m$ for different $m$, $A_1 = \text{Diag}\{f'_{m}(\theta^T x_m)\}_{m=1}^M$ with $\text{Diag}\{v_m\}_{m=1}^M$ denoting a diagonal matrix in which the entries on the main diagonal are specified by $v_1, v_2, ..., v_M$, and $\Lambda_2$ is the Hessian of the penalty term. Accordingly, it is generally possible to construct a local minimum point $\theta_{\text{min}}$, where $H(\theta_{\text{min}}) \geq 0$, can also satisfy $f''_{m}(\theta_{\text{min}}^T p_m) < 0$ for some $m$. Therefore, choosing $[a_m, b_m]$ in (12) is not a trivial task as solutions with $f''_{m} < 0$ must also be allowed. We propose a principled approach to tackle this problem in Sec. 4.

Based on the scalar surrogates in (12), we construct a multivariate convex surrogate to a function of the form $\sum m f_m(\theta^T x_m)$ (see (2)) by substituting

$$z_m = \theta^T x_m, \quad \hat{z}_m = \hat{\theta}^T x_m, \quad (13)$$

for each $m$ separately, where $z_m$ and $\hat{z}_m$ are the current and reference values of the argument of $f_m$, respectively. This is summarized in the following lemma.

**Lemma 2.1** Let $F(\theta) = \sum m f_m(\theta^T x_m)$ with $f_m$ a twice continuously differentiable (possibly nonconvex) function. Define

$$\tilde{F}(\theta; \hat{\theta}) := F(\theta) + \frac{1}{2}(\theta - \hat{\theta})^T XDX^T(\theta - \hat{\theta}), \quad (14)$$

where $\hat{\theta} \in \Omega_F$ is an expansion point, $X \in \mathbb{R}^{p \times M}$ is a matrix with the $m$th column equal to $x_m$ and $D$ is a diagonal matrix

$$D := \text{Diag}\{[-c_m(a_m, b_m)]_{m=1}^M\}, \quad (15)$$

where $c_m(a_m, b_m)$ is defined in (11). The function $\tilde{F}(\theta; \hat{\theta})$ is a convex local first-order surrogate (see (3)–(5)) for $F$ on the majorization domain $\Omega_M := \{\theta | a_m \leq \theta^T x_m \leq b_m, \forall m\}$ when $a_m, b_m$ are chosen such that $\hat{\theta} \in \text{Int}(\Omega_M)$. (See proof in supplementary material).

$\tilde{F}$ in (14) is generally not convex outside $\Omega_M$. In order to allow the use of any general algorithm for convex functions, we extend the surrogate to be globally convex. This extension will also be essential for our decomposition method in Sec. 2.3 that relies on Jensen’s inequality and requires the function to be globally convex. First, we introduce an extension to $\tilde{F}$ from (12) in Eq. (16) (see next page), where $f'_m(z; \hat{z})$ and $f''_m(z; \hat{z})$ denote the first and second derivatives of $f_m$ with respect to $z$. Note that $f_m$ in Eq. (16) is globally convex even in regions where it is not an upper bound of $f_m$ and it is also twice continuously differentiable. The extension of $\tilde{F}$ from (14) can now be written as

$$\tilde{F}(\theta; \hat{\theta}) = \sum_m \tilde{f}_m(\theta^T x_m; \hat{\theta}^T x_m). \quad (17)$$

The proposed MMP with Dynamically Restricted Majorization Domains (DRD) is summarized in Algorithm 2.1. It is important to note that it is not necessary to solve the surrogate subproblem in (8) exactly. It is sufficient just to decrease the value of the surrogate which will result in the decrease of $C$ in (1). Since the surrogate is a local approximation of $C$ it might explore a small region in $\theta$ space and it is more beneficial to re-expand the surrogate at a new point than to minimize the current surrogate exactly. In addition, the extra constraint due to $\Omega_M$ in (8) might make the problem more difficult to solve. Instead, we first minimize the surrogate in Algorithm 2.1 without constraining the solution to lie inside the majorization domain $\Omega_M$ and then check if it lies in $\Omega_M$. If not, we project the solution to the boundary of $\Omega_M$ along the line connecting the current solution $\theta_0^{(t+1)}$ and the previous one $\theta_0^{(t)}$. The following lemma asserts that this projection always exists and necessarily leads to a decrease of the surrogate which implies the decrease of $C$. We also note that $\Omega_M$ can be modified as needed, so in cases where these projections slow down the convergence, one can expand $\Omega_M$ to allow larger step sizes.

**Lemma 2.2** Let $\Omega_F$ and $\Omega_M$ be non-empty convex sets with $\Omega_F \cap \Omega_M \neq \emptyset$ and $\Omega_M := \{\theta | a_m \leq \theta^T x_m \leq b_m, \forall m\}$. Let $\theta_1 \in \Omega_F \cap \Omega_M$ and let $\theta_2 \notin \Omega_M$ s.t. $F(\theta_2) < F(\theta_1)$ with $F$ a convex function. In addition, assume $a_m, b_m$ are chosen such that $\theta_1 \in \text{Int}(\Omega_M)$. Then there exists $\lambda \in (0, 1)$ s.t. $\theta^* := \lambda \theta_2 + (1 - \lambda) \theta_1$ satisfies $\theta^* \in \Omega_F \cap \Omega_M$. Furthermore, any such point satisfies $F(\theta^*) < F(\theta_1)$. (Proof is provided in the supplementary material).
\[ f_m(z; \bar{z}) := \begin{cases} f_m(a_m) + \bar{f}_m(b_m)(z - b_m) + \bar{f}_m(b_m)(z - b_m)^2 / 2, & z < a_m \\ f_m(z; \bar{z}), & a_m \leq z \leq b_m \\ f_m(b_m) + \bar{f}_m(b_m)(z - b_m) + \bar{f}_m(b_m)(z - b_m)^2 / 2, & z > b_m \end{cases} \] (16)

**Algorithm 2.1: MM-DRD**

for \( t \leftarrow 1 \) to \( T \) (or until converged)

for \( m \leftarrow 1 \) to \( M \)

Choose \( [a_m^{(t)}, b_m^{(t)}] \)

comment: see Sec. 4 for details

Compute \( c_m^{(t)} \) in (11) using \([a_m^{(t)}, b_m^{(t)}]\)

Find \( \bar{\theta}^{(t+1)} \in \text{arg min}_{\theta \in \Omega_F} \{ \bar{F}(\theta; \bar{\theta}^{(t)}, \Omega_M^{(t)}) \} \)

comment: \( \bar{F} \) is given in Eq. (17)

Let \( \Omega_M^{(t)} := \{ \theta | a_m^{(t)} \leq \theta^T x_m \leq b_m^{(t)}, \forall m \} \)

if \( \bar{\theta}^{(t+1)} \notin \Omega_M^{(t)} \) then

Find \( \lambda \in (0, 1) \) s.t.

\( \bar{\theta} = \lambda \bar{\theta}^{(t+1)} + (1 - \lambda) \bar{\theta}^{(t)} \in \Omega_M^{(t)} \cap \Omega_F \)

set \( \bar{\theta}^{(t+1)} = \bar{\theta}^* \)

return \( \bar{\theta}^{(t+1)} \)

One practical solution to performing the projection is to use the point on the boundary of \( \Omega_M \) along the line connecting the previous and current solution, which has the following simple solution

\[ \lambda = \min \left( \left\{ \frac{b_m^{(t)} - x_m^T \theta^{(t)}}{x_m^T (\bar{\theta}^{(t+1)} - \bar{\theta}^{(t)})}, \frac{x_m^T (\bar{\theta}^{(t+1)} - \bar{\theta}^{(t)})}{x_m^T (\bar{\theta}^{(t+1)} - \bar{\theta}^{(t)})} \right\}_m \cap (0, 1) \right). \] (18)

It should be noted that the convex surrogate in (17) can be minimized by any standard software package for convex optimization, which is already more convenient than minimizing the original nonconvex cost function.

### 2.3 Jensen-Type Decomposition for Parallel Block Coordinate Descent

As shown in Sec. 3, a necessary condition for Algorithm 2.1 to converge is that the surrogate is decreased. This requires one to repeatedly evaluate the surrogate in order to determine the step-size (line-search). However, for high dimensional problems, with a large number of variables and data points, this operation is extremely time consuming. In order to alleviate this difficulty, we make use of a powerful decomposition procedure that is based on Jensen’s inequality (Boyd and Vandenberghe, 2004) and originally proposed by De Pierro (1994) that results in a surrogate that is additively separable with respect to blocks of variables (Erdogan and Fessler, 1999). This leads to a parallel block coordinate descent algorithm where all blocks are updated simultaneously while still guaranteeing the decrease of the objective function. Since the optimization using each block potentially involves a small number of variables, it allows the use of second-order methods. Note that naively minimizing the objective with respect to blocks of variables in parallel without using the decomposition we present here would not guaranty the decrease of the objective.

We can now introduce the above mentioned decomposition. Separate \( \bar{\theta} = [\theta^1, \theta^2; \ldots; \theta^K] \) into \( K \) arbitrary blocks. Define \( \mathbb{S}^k \) as the set of all indexes used in the \( k \)th block with the corresponding variables denoted by the column vector \( \theta^k \) and \(|\mathbb{S}^k|\) as the number of variables in \( \theta^k \). We define the following function

\[ S_m(\theta; \hat{\theta}) := \sum_{k=1}^K S_m^k(\theta^k; \hat{\theta}^k, r_m^k, x_m^k) \]

\[ := \sum_{k=1}^K r_m^k \mathcal{f}_m(\hat{\theta}^k x_m + (\theta^k - \hat{\theta}^k)^T x_m / r_m^k), \] (19)

where \( \mathcal{f}_m \) is defined in (16), \( x_m^k \in \mathbb{R}^{\mathbb{S}_k} \) is a subset of \( x_m \) corresponding to \( \theta^k \) and \( r_m^k \) is defined as

\[ r_m^k = \|x_m^k\|_1 / \|x_m\|_1. \] (20)

Note that \( S_m \) is an additively separable function with respect to the blocks (\( S_m^k \) corresponds to the \( k \)th block), so it naturally leads to a parallel block coordinate descent approach where all blocks are updated simultaneously while still guaranteeing the decrease of the objective function.

**Lemma 2.3** The function \( S := \sum_m S_m(\theta; \hat{\theta}) \) with \( S_m \) defined in (19) is a twice continuously differentiable convex local first-order surrogate (see (3)–(5)) for \( F = \sum_m f_m(\theta^T x_m) \) on the majorization domain \( \Omega_m := \{ \theta | a_m \leq \theta^T x_m \leq b_m, \forall m \} \) when \( a_m \) and \( b_m \) are chosen such that \( \theta \in \text{int}(\Omega_M) \). (Proof is provided in the supplementary material).

**Remark** The function in (19) is a surrogate for \( F \) on \( \Omega_M \) only if \( f \) is globally convex since Jensen’s inequality only holds in this case. More specifically, consider the argument of \( \mathcal{f}_m(\theta^T x_m + (\theta^k - \hat{\theta}^k)^T x_m / r_m^k) \) and note that the argument can be outside \([a_m, b_m]\) even if \( \theta \in \Omega_M \), so in that case, Jensen’s inequality would not hold if \( f \) was not convex outside \([a_m, b_m]\). This provides another motivation for (16).

It is worth noting that for a smaller number of groups \( K \), the surrogate in (19) will be a tighter bound to
the original objective function with the trivial case of \( S_m(\theta) = \hat{f}_m(\theta^T x_m) \) for all \( m \) when \( K = 1 \). Decreasing \( K \) will also result in harder optimization problems involving more variables for each group. On the other hand, increasing \( K \) leads to better parallelization, while resulting in a less tight bound, which can slow down the convergence per iteration but can also reduce computation time per iteration. Note that in this approach, there is complete freedom to choose any group size and to explore the above trade-off according to the available computational resources. The resulting block-coordinate algorithm is described by Algorithm 2.1 by replacing the minimization of \( \tilde{F} \) by the minimization described in Algorithm 2.2 where \( \Omega^k_F \) is the projection of \( \Omega_F \) onto the subspace of \( \theta^k \). We call this algorithm Parallel Majorization-Minimization with Dynamically Restricted Domains (PMM-DRD).

3 Convergence Analysis

In order for the proposed iterative procedure to be generally useful, it must converge to a local optimum or a stationary point from all initialization states without exhibiting divergence or oscillation. To understand this analysis, we first introduce the notion of a point-to-set mapping. A point-to-set map \( A \) from a set \( X \) into a set \( Y \) is defined as \( A : X \rightarrow \mathcal{M}(Y) \), which assigns a subset of \( Y \) to each point of \( X \), where \( \mathcal{M}(Y) \) denotes the power set of \( Y \). Next, we introduce several definitions related to the properties of point-to-set mappings. A point-to-set map is said to be closed at \( x \in X \) if \( x(t) \rightarrow x \) as \( t \rightarrow \infty \) with \( x(t) \in X \) and \( y(t) \rightarrow y \) as \( t \rightarrow \infty \) with \( y(t) \in A(x(t)) \) implies \( y \in A(x) \). A point-to-set map is said to be closed on \( W \subset X \) if it is closed at every point of \( W \). A fixed point of the map \( A \) is a point for which \( x \in A(x) \). A generalized fixed point of a map is a point for which \( x \in A(x) \). \( A \) is said to be monotonic with respect to \( \phi \) whenever \( y \in A(x) \) implies \( \phi(y) \leq \phi(x) \). Many iterative algorithms (including the ones presented here) can be described using this notion of point-to-set maps.

Let \( X \) be a set and \( x(0) \in X \) a given initial point. Then an algorithm \( A \) with initial point \( x(0) \) is a point-to-set map \( A : X \rightarrow \mathcal{M}(X) \) which generates a sequence \( \{x(t)\}_{t=1}^{\infty} \) via the rule \( x(t+1) = A(x(t)) \), \( t = 0, 1, 2, \ldots \). \( A \) is said to be globally convergent if for any chosen initial point \( x(0) \), the sequence \( \{x(t)\}_{t=1}^{\infty} \) converges to a point for which a necessary condition of optimality holds. The property of global convergence expresses, in a sense, the certainty that the algorithm works. It is important to stress the fact that it does not imply (contrary to what the term might suggest) convergence to a global optimum for all initial points \( x(0) \). We analyze Algorithm 2.1 using the above tools. We start by stating Zangwill’s global convergence theorem (Zangwill, 1969, Convergence theorem A page 91).

**Theorem 3.1 (Zangwill, 1969)** Let \( A : X \rightarrow \mathcal{M}(X) \) be a point-to-set map that gives a point \( x_0 \in X \) generates a sequence \( \{x(t)\}_{t=1}^{\infty} \) through \( x(t+1) \in A(x(t)) \). Also, let a solution set \( \Gamma \subset X \) be given and

1. All points \( \{x(t)\}_{t=1}^{\infty} \) are in a compact set \( W \subset X \).
2. There is a continuous function \( \phi : X \rightarrow R \) s.t.
   - (a) \( x(t) \notin \Gamma \Rightarrow \phi(y) < \phi(x(t)) \), \( \forall y \in A(x(t)) \)
   - (b) \( x(t) \in \Gamma \Rightarrow \phi(y) \leq \phi(x(t)) \), \( \forall y \in A(x(t)) \)
   - (c) \( A \) is closed at \( x(t) \) if \( x(t) \notin \Gamma \)

Then all limit points\(^4\) of \( \{x(t)\}_{t=1}^{\infty} \) are in \( \Gamma \). Furthermore, \( \lim_{t \rightarrow \infty} \phi(x(t)) = \phi(x^*) \) for all limit points \( x^* \).

The general idea in showing the global convergence of an algorithm \( A \) is to invoke Theorem 3.1 by appropriately defining \( \phi \) and \( \Gamma \). For an algorithm that solves the minimization problem \( \min \{f(x) : x \in \Omega\} \), the solution \( \Gamma \) is usually chosen to be the set of corresponding stationary points and \( \phi \) can be chosen to be the objective function itself if it is continuous. Our approach would be to define \( \Gamma \) as the set of generalized fixed points and then use the fact that any generalized fixed point is also a stationary point, as established by the lemma below.

**Lemma 3.2** Suppose \( x^* \) is a generalized fixed point of \( A \) described by Algorithm 2.1 (or Algorithm 2.1 with the minimization of \( \tilde{F} \) replaced by Algorithm 2.2). Assume that the constraints as specified by \( \Omega_F \) in (1) are qualified at \( x^* \). Furthermore, assume that \( \Omega_F = \{g_i(\theta) \leq 0, h_{ij}(\theta) = 0, i = 1, \ldots, I, j = 1, \ldots, J\} \) where \( g_i(\theta) \) and \( h_{ij}(\theta) \) are differentiable convex functions. Then, \( x^* \) is a stationary point of the program in (1). (See proof in the supplementary material).

We are now ready to state our main result.

**Theorem 3.3 (Global convergence)** Let \( \{\theta(t)\}_{t=1}^{\infty} \) be any sequence generated by the algorithm described by

\(^4\) A limit point is defined as the limit of a convergent subsequence.
Algorithm 2.1 (or Algorithm 2.1 with the minimization of $F$ replaced by Algorithm 2.2) using any initialization point $\theta^{(0)} \in \Omega_F$. Assume $\Omega_F$ in (1) is closed and bounded and given by $\Omega_F = \{g_i(\theta) \leq 0, h_j(\theta) = 0, i = 1, ..., I, j = 1, ..., J\}$ where $g_i(\theta)$ and $h_j(\theta)$ are differentiable convex functions. Assume $f$ and $R$ in (2) are twice continuously differentiable functions. Then assuming suitable constraint qualifications (given by $\Omega_F$ and $\Omega_M$), any sequence $(\theta^{(i)})_{i=1}^\infty$ has a limit point which is a stationary point of the program in (1), i.e., it satisfies the first-order KKT optimality conditions (Proof is provided in the supplementary material).

4 Choosing the Majorization Domains

Here we describe the algorithm for choosing the majorization domains $[a_m, b_m]$ in (11). A preprocessing step involves storing all the inflection points $\mathcal{I}_m = \{z \mid f''_m(z) = 0\}$, points with minimum negative 2nd derivatives $\mathcal{M}_m = \{z \mid z \in \arg \min f''_m(z), f''_m(z) < 0\}$, and the corresponding values of the second derivative $\chi_m = \{f''_m(z) \mid z \in \mathcal{M}_m\}$, for each $m = 1, 2, ..., M$. We denote $\mathcal{M}_m[\mathcal{I}]$ as the $q$th element in the ordered set $\{\mathcal{M}_m[\mathcal{I}]\}_{q=1}^Q$ consisting of elements of $\mathcal{M}_m$ and satisfying $\mathcal{M}_m[1] \leq \mathcal{M}_m[2] \leq \ldots \leq \mathcal{M}_m[Q]$. We define a similar ordered set for $\mathcal{I}$. For simplicity, we assume that $\chi_m$ is a singleton set, but $\mathcal{M}_m$ can still contain multiple points. Algorithm 4.1 describes the procedure for selecting the majorization domain, where two parameters $0 < \alpha, \beta < 1$ need to be chosen. For each $m$, Algorithm 4.1 returns an interval $[a_m, b_m]$ for which the minimal curvature is computed and used in (11). An example for using this procedure is shown in the supplementary material. In case there are multiple local minimum values of $f''(z)$, i.e., $\chi_m$ contains multiple different values, Algorithm 4.1 is repeated for each minimum value, after which $[a_m, b_m]$ are obtained by taking the intersection of all the majorization domains.

Algorithm 4.1: DOMAINS($\alpha, \beta$)

for $m \leftarrow 1$ to $M$

\begin{align*}
    a_m &= -\infty \\
    b_m &= \infty \\
    \text{Compute } z_m &= \theta^T x_m \\
    \text{Compute } \psi_m &= f''_m(z_m) \\
    \text{if } 0 < |\psi_m| < \alpha |\chi_m| &\text{ comment: shallow region} \\
    [a_m, b_m] &= \text{Localize}(z_m, \beta, \mathcal{M}_m) \\
    \text{if } \alpha |\chi_m| \leq |\psi_m| &\leq |\chi_m| \\
    \text{comment: medium curved convex region} \\
    [a_m, b_m] &= \text{Localize}(z_m, \beta, \mathcal{I}_m) \\
\end{align*}

return $\{[a_m, b_m]_{m=1}^M\}$

5 Numerical Results

As an application of our proposed framework for non-convex optimization, we consider two binary linear classification problems that follow the general form in Eq. (2). The first example we consider is the sigmoid-loss linear support vector machine (SVM). Standard SVM utilizes a hinge loss function as a surrogate to the 0-1 loss, which is easier to optimize and introduces a margin that allows good generalization. However, to retain a convex loss function it overly penalizes examples with large negative margins. To remedy this, Ertekin et al. (2011) proposed a ramp loss function (difference of two hinge functions). The sigmoid loss considered here is a smooth version of the ramp-loss and is given by $\ell(\alpha) = 1 - \tanh(\alpha)$ (see Fig. 1 in the supplementary material).

An example we consider is logistic regression with a log penalty (Mairal, 2015) that produces sparser solutions than the L1 penalty but makes the overall problem non-convex. The regularizer is given by $R(\theta) = \lambda \sum_{j=1}^{p} \log(\theta_j^2 + \epsilon)$ where $\epsilon > 0$ is chosen to prevent the log penalty from blowing up when $\theta_j = 0$. We compare the proposed method PMM-DRD, which is described by Algorithm 2.1 with Algorithm 2.2 replacing the minimization therein, against previously proposed methods in the literature for solving non-convex optimization problems such as Gradient Descent, L-BFGS (Schmidt, 2005), RMSProp (Tieleman and Hinton, 2012), AdaGrad (Duchi et al., 2011), and PSCA (Razaviyayn et al., 2014). We also compare to the case where the proposed method does not use restricted majorization domains, denoted PMM.

In the following examples we consider binary classification tasks of distinguishing between digits 3 and 5 in the MNIST$^2$ dataset, newsgroups 1 and 20 in the 20Newsgroup$^3$ dataset, and HIV positive and negative patients in the TB$^4$ dataset. The L-BFGS algorithm

Algorithm 4.2: LOCALIZE($z, \beta, P$)

if $z < P[1]$

\begin{align*}
    a &= -\infty \\
    b &= (1 - \beta)z + \beta P[1]
\end{align*}

else if $z > P[end]$

\begin{align*}
    b &= \infty \\
    a &= (1 - \beta)z + \beta P[end]
\end{align*}

else

\begin{align*}
    \text{Find } i \text{ s.t. } P[i] < z < P[i + 1] \\
    a &= (1 - \beta)z + \beta P[i] \\
    b &= (1 - \beta)z + \beta P[i + 1]
\end{align*}

return $(a, b)$

2http://yann.lecun.com/exdb/mnist/
3http://qwone.com/ jason/20Newsgroups/
Table 1: Classification accuracy (%) on the training set using SVM. LIBLINEAR (Fan et al., 2008) uses the standard linear SVM, and the other methods use the nonconvex sigmoid-loss. For the latter, 10 different random initializations were used and the mean and standard deviation are presented. GD = Gradient Descent, RProp = RMSProp, AGrad = AdaGrad, PMM = Parallel Majorization-Minimization, DRD = Dynamically Restricted Domain.

<table>
<thead>
<tr>
<th>Method</th>
<th>MNIST</th>
<th>20 News</th>
<th>TB</th>
</tr>
</thead>
<tbody>
<tr>
<td>LIBLIN</td>
<td>97.13</td>
<td>100</td>
<td>86.55</td>
</tr>
<tr>
<td>LBFGS</td>
<td>96.92 ± 0.02</td>
<td>99.63 ± 0.21</td>
<td>65.99 ± 0</td>
</tr>
<tr>
<td>CG</td>
<td>96.91 ± 0.026</td>
<td>98.36 ± 0.72</td>
<td>65.99 ± 0</td>
</tr>
<tr>
<td>GD</td>
<td>95.13 ± 1.04</td>
<td>95.16 ± 5.61</td>
<td>65.99 ± 0</td>
</tr>
<tr>
<td>PSCA</td>
<td>89.23 ± 0.07</td>
<td>79.29 ± 0.30</td>
<td>65.99 ± 0</td>
</tr>
<tr>
<td>RProp</td>
<td>96.74 ± 0.32</td>
<td>99.87 ± 0.09</td>
<td>65.99 ± 0</td>
</tr>
<tr>
<td>AGrad</td>
<td>94.99 ± 0.24</td>
<td>81.41 ± 0.63</td>
<td>65.99 ± 0</td>
</tr>
<tr>
<td>PMM</td>
<td>97.02 ± 0.02</td>
<td>100 ± 0</td>
<td>100 ± 0</td>
</tr>
<tr>
<td>PMM-DRD</td>
<td>97 ± 0.01</td>
<td>100 ± 0</td>
<td>100 ± 0</td>
</tr>
</tbody>
</table>

was used with 1-fold cross validation to learn the regularization coefficient $\lambda$ in (2), which was then fixed to the same value for all nonconvex optimization methods. All algorithms ran till convergence, i.e., till one of the stopping criteria were met (see supplementary material for details). Due to the possible existence of local minima in the nonconvex models, we used 10 different random initializations and presented the mean results with standard deviations.

Table 1 shows the classification accuracy of the sigmoid-loss SVM on the training set. One can see that among the methods for nonconvex optimization, the highest accuracy for all datasets is achieved by PMM-DRD. For the TB dataset, PMM significantly outperforms all the other methods. Figure 2 shows the nonconvex objective vs training time for the TB dataset, where it can be seen that all methods get stuck in a bad local minimum, except for the proposed PMM. Note that the restricted majorization domain in PMM-DRD leads to a convergence speed-up by a factor $\geq 5$. Table 2 shows the classification accuracy of SVM on the test set. Table 3 shows the classification accuracy of logistic regression on the training set (see Table 1 in the supplementary material for details). Due to the possible existence of local minima in the nonconvex models, we used 10 different random initializations and presented the mean results with standard deviations.

All experiments use a 64-core machine. In RMSProp and AdaGrad we use mini-batches which are 10% of the examples. The rest of the methods use the entire dataset (batch). PMM methods use blocks of 500 features. For PMM-DRD we use $(\alpha, \beta) = (0.1, 0.8)$ in algorithm 4.1. The minimization of the surrogates in PMM methods was done using the trust region conjugate gradient algorithm (Lin, 2008). We implemented PMM in MATLAB using C/C++ mex with OpenMP enabled. PSCA uses the cyclic version (Razaviyayn et al., 2014) with 64 processors, 40 scalar features per processor, and a learning rate that is initially 1 and decreases as $1/k^{1.01}$, where $k$ is the iteration number.

**Acknowledgements**

This research was supported in part by ARO, DARPA, DOE, NGA, ONR and NSF.
References


