UNIVERSITÀ
DEGLI STUDI
FIRENZE

Università degli Studi di Firenze
Dipartimento di Ingegneria dell'Informazione (DINFO)
Corso di Dottorato in Ingegneria dell'Informazione
Curriculum: Automatica, Ottimizzazione e Sistemi Complessi

Decomposition methods for constrained nonlinear optimization

Candidate
Giulio Galvan

Supervisor
Prof. Marco Sciandrone

PhD Coordinator
Prof. Fabio Schoen

ciclo XXXII, 2016-2019

Università degli Studi di Firenze, Dipartimento di Ingegneria dell'Informazione (DINFO).

Abstract

This thesis is concerned with decomposition methods for constrained nonlinear optimization. Decomposition methods are a popular and effective technique to deal with problems which enjoy a special structure either in their feasible set or in the way the function to be minimized is defined. In the present thesis three different classes of problems are considered. Each problem is characterized, in a different way, by a special structure which can be leveraged upon by decomposition methods.

The first part of the thesis is concerned with problems whose feasible set can be described by the intersection of two closed convex sets. A modification of the Augmented Lagrangian Method is proposed to solve this class of problems. The algorithm has a close relationship with the ADMM algorithm against which is thus compared. The convergence of the algorithm is proved for general nonlinear nonconvex differentiable functions. A modification of this scheme for nonsmooth function is also proposed.

The second part of the thesis is concerned with decomposition methods to solve the optimization problems arising from the training of SVMs. The literature on decomposition methods for SVMs training is huge and several different decomposition algorithms have been proposed over the last decades. In state-of-the-art solver, like the popular libSVM, the original problem is decomposed in several smaller problems usually consisting of 2 variables. In this thesis we propose a novel way to decompose the problem into bigger sub-problem (up to 50 variables) and a novel way to solve the sub-problems themselves. An extensive numerical experimentation shows that this strategy often outperforms state-of-art solvers.

Finally, the last part of the thesis deals with derivative free methods. A general nonsmooth optimization problem with convex constraints is "decomposed" into a bi-level optimization problem where the upper-level problem has as an objective a modification of the original objective function and the lower-level one accounts for the constraints. Such problem is proven to be equivalent to the original problem. The special structure of the resulting bi-level problem is simple enough so that the problem can be solved with standard derivative-free methods. Extensive numerical evidence shows that the method is competitive with state-of-art solvers for constrained derivative free optimization and its convergence properties extends nicely to a wider class of problems.

Contents

Chapter 1

Introduction

Decomposition methods have been extensively studied and effectively employed to solve different classes of problems in several fields over the last decades often achieve state-of-art performances. Just as an example they have been employed to solve portfolio selection problems, or in machine learning to solve the training problem of Support Vector Machines (SVMs).

These kind of methods tackle an optimization problem by solving a sequence of sub-problems obtained by fixing some variables of the problem and optimizing w.r.t. to the others. Solving a succession of sub-problems instead of dealing with all the variables at the same time can be a necessity for very large problems where it is impossible to store all the variables into main memory or when we are working in a distributed environment where data is scattered around a number of processing units. The advantages of using decomposition methods, however, are not limited to very large or distributed contexts. In fact, the sub-problems can be considerably easier to solve (maybe with specialized techniques) than the original problem so that solving a succession of sub-problems can be a lot faster than applying standard optimization methods to the original problem. For example

- it may be easy to find the global (or local) minimum of a sub-problem because it is available in closed form, or because a specialized algorithm exists,

- by choosing a specific decomposition scheme some of the sub-problems can be solved in parallel,

- some of the sub-problems can be convex even when the original problem is not,

- the feasible set of the original problem nicely partition over some group of variables so that the sub-problems are considerably easier to solve with standard methods.

Classical decomposition methods for unconstrained optimization work by partitioning a vector of variables $x \in \mathbb{R}^n$ in $m \leq n$ blocks:

$$x = (x_1,\ x_2,\ \ldots,\ x_m),$$

and by minimizing the objective function $f : \mathbb{R}^n \to \mathbb{R}$ w.r.t. to single blocks of variables. Which kind of minimization step is employed and in which order the blocks are selected give rise to different methods with different convergence properties. As a minimization step, for instance, we can find the global or local minimum of the problem or perform a descent step with some kind of local search.

We can distinguish between *parallel* methods where at each iteration all the blocks of variables are considered and the corresponding sub-problems solved in parallel and *sequential* methods where only one block of variables is considered at a time.

In parallel methods after all the solution of the sub-problems are computed is necessary to choose the new iterate in such a way to ensure the convergence of the algorithm. In the *Jacobi* method for instance the new iterate is chosen as the solution of the sub-problems which enjoys the lower objective function value.

Sequential methods instead differ in the way the blocks of variables are selected. In the *Gauss-Seidel* and in the *Block descent* methods, for instance, we loop over all of the m blocks in an orderly fashion while in the *Gauss-Southwell* method at each iteration we select the blocks of variables which mainly violate some optimality condition.

Decomposition methods for unconstrained optimization can be, in most cases, extended to the constrained case under suitable assumption on the feasible set. A special case, in which decomposition methods are particularly useful, is when the feasible set X can be partitioned over the variables so that $X = X_1 \times X_2 \times \cdots \times X_m$ and the constraints can be expressed as $x_1 \in X_1,\ x_2 \in X_2,\ \ldots,\ x_m \in X_m$.

Often, however, problems with a particular structure cannot be directly dealt with general decomposition methods or offer the possibility to devise specialized methods that leverage upon the particular nature of the problem.

In this thesis we concern ourselves with three different classes of problems where specialized decomposition techniques can be effectively used.

Decomposition for the augmented Lagrangian method The first classes of problems we consider consists of problems whose feasible set Ω can be expressed as the intersection of two (or more) sets $\Omega = X \cap Y$. This offer the possibility to modify the problem in such a way that the variables nicely separates over the different sets:

$$\begin{aligned} \min_{x,y} \quad & f(x) \\ s.t. \quad & x \in X,\ y \in Y, \\ & x - y = 0. \end{aligned}$$

A popular way to solve the latter is the augmented Lagrangian method (ALM). In its minimization step the method minimize the augmented Lagrangian function w.r.t. both x and y variables before updating the Lagrange multipliers and the penalty parameter τ:

$$\min_{x,y} \mathcal{L}_\tau(x, y, \lambda) = f(x) + \lambda^T(x - y) + \frac{\tau}{2} \|x - y\|^2 .$$

In this thesis we propose to modify the method by applying a 2 block *Gauss-Seidel* like decomposition scheme to this step. We consider both the case where f is continuously differentiable and derivatives are available and the case where f is still differentiable but its derivatives are not available or too expensive to compute. In the latter we exploit derivative-free techniques together with the two block decomposition.

Two-level decomposition for SVM training The second class of problems we consider is constrained quadratic optimization problems that arise in the training phase of SVMs (in its dual formulation).

Given a data set of n training vectors $v_i \in \mathbb{R}^m$ and their associated labels

$y_i \in \{-1, 1\}$, the problem is given as:

$$\begin{aligned} \min_{\alpha} \; & f(\alpha) = \frac{1}{2}\alpha^T Q \alpha - \alpha^T e \\ & y^T \alpha = 0 \\ & 0 \leq \alpha \leq C, \end{aligned}$$

where $\alpha \in \mathbb{R}^n$ and $Q \in \mathbb{R}^{n \times n}$ is given by $Q_{ij} = y_i y_j K_{ij}$, where K is the kernel matrix with $K_{ij} = \phi(v_i)^T \phi(v_j)$, for some mapping ϕ.

Decomposition methods have been extensively studied over the years and are currently employed in state-of-the-art solvers like *libSVM*. A common approach is to select a group of variables looking at the violation of KKT conditions (in a *Gauss-Southweel* fashion) to form the sub-problems. The dimension of the sub-problem is usually very small often consisting of only 2 variables which yield nice closed form updates.

In this thesis we study a novel way to select the variables that give rise to bigger sub-problems (from 4 to 50 variables) and we use a second level of decomposition to solve the sub-problems themselves.

Bi-level decomposition for constrained derivative free optimization As a last case we consider a general nonsmooth constrained optimization problem

$$\begin{aligned} \min \quad & f(x) \\ s.t. \quad & x \in X, \end{aligned}$$

where f is possibly nonsmooth (and first order information is not available) and X is a closed convex set. We propose a novel formulation which decouples the objective function and the constraints in the upper and lower levels of the following bi-level problem:

$$\begin{aligned} \min_{y} \quad & f(x) + \epsilon \left\| x - y \right\| \\ & x = \operatorname*{argmin}_{z \in X} \left\| z - y \right\|, \end{aligned} \tag{1.1}$$

where ϵ is any positive number. In this thesis we prove the equivalence in terms of local and global minima of the latter formulation to the original problem. Then we show how to apply any given algorithm for nonsmooth unconstrained optimization to solve the proposed formulation. The obtained results compare favorably against the state of the art.

The thesis is organized as follows. Chapter 2 is devoted to the augmented Lagrangian method. We consider the settings when the constraints are general abstract convex sets with no explicit formulation and no additional assumptions are available. We prove the convergence of ALM in this setting filling a gap in the literature. This serves as a basis for Chapter 3 where decomposition in used to modify the ALM scheme. We adapt this approach to the derivative-free setting in Chapter 4. Then we move to the study of the SVM training problem in Chapter 5 and finally, in Chapter 6, we deal with general nonsmooth constrained problems.

Chapter 2

The augmented Lagrangian method for problems with abstract convex constraints

2.1 Introduction

This chapter is concerned with the convergence of the augmented Lagrangian method (ALM) for constrained optimization problems of the form

$$\begin{aligned} &\min f(x) \\ &\quad g(x) \leq 0, \\ &\quad x \in X, \end{aligned} \tag{2.1}$$

where $f : \mathbb{R}^n \to \mathbb{R}$ is a continuously differentiable function, $g : \mathbb{R}^n \to \mathbb{R}^p$, is component-wise convex and continuously differentiable and X is a closed convex set.

The augmented Lagrangian for problem (2.1), following [14], is defined as

$$\mathcal{L}_\tau(x,\ \mu) = f(x) + \frac{\tau}{2} \sum_{i=1}^{p} \left[\max \left\{ 0,\ g_i(x) + \frac{\mu_i}{\tau} \right\} \right]^2,$$

where μ is the vector of Lagrangian multipliers and τ the penalty parameter. With this notation we write the gradient of the augmented Lagrangian as

$$\nabla_x \mathcal{L}_\tau(x,\ \mu) = \nabla f(x) + \tau \sum_{i=1}^{p} \max \left\{ 0,\ g_i(x) + \frac{\mu_i}{\tau} \right\} \nabla g_i(x).$$

Note that $\nabla_x \mathcal{L}_\tau$ is continuous.

Augmented Lagrangian methods work by solving a sequence of subproblems of the form

$$\min_{x \in X} \mathcal{L}_{\tau_k}(x,\ \mu^k) \tag{2.2}$$

where μ^k is the current multiplier estimate and τ_k the penalty parameter. After the new iterate x^{k+1} has been computed by solving (2.2) up to ϵ_k-stationarity (as it will be defined shortly), the multipliers μ_k are updated with

$$\tilde{\mu}_i^{k+1} = \max\{0,\ \mu_i^k + \tau_k g_i(x^{k+1})\}$$

We focus, in particular, on augmented Lagrangian methods that employ multiplier safeguarding techniques [3, 4, 14, 16, 17] in contrast to classical methods [12,28,75,80] as summarized in [46]. In such methods the multiplier estimate is corrected to ensure its boundedness, for example by means of a projection onto a bounded set $[0, \bar{\mu}]$:

$$\mu_i^{k+1} = \Pi_{[0,\ \bar{\mu}]}\left[\tilde{\mu}_i^{k+1}\right]$$

Finally the penalty parameter τ_k may be increased and the accuracy increased (by decreasing ϵ_k). The complete Augmented Lagrangian method is given in Algorithm 1.

Algorithm 1: Augmented Lagrangian Method

Input: $x^0 \in X$, $\mu^0 > 0$, $\delta > 1$, $\sigma \in (0,\ 1)$, $\tau_0 > 0$, a positive (decreasing) sequence $\{\epsilon_k\}$

$\|V^0\| = +\infty$
for $k = 0,\ 1,\ \ldots$ **do**
  $x^{k+1} \leftarrow$ an ϵ_k-stationary point of (2.2)
  $\mu^{k+1} = \Pi_{[0,\ \bar{\mu}]}\left(\mu_i^k + \tau_k g_i(x^{k+1})\right)$
  $V_i^{k+1} = \min\{-g_i(x^{k+1}),\ \frac{\mu_i^k}{\tau_k}\} \quad i = 1,\ \ldots\ p$
  if $\|V^{k+1}\| \leq \sigma \|V^k\|$ **then**
    $\tau_{k+1} = \tau_k$,
  else
    $\tau_{k+1} = \delta \tau_k$.

Our interest lies in contexts where 1) no additional assumptions besides closedness and convexity are made on X and 2) the Lagrangian sub-problems

are solved by computing stationary points with increasing accuracy. The presence of abstract constraints together with the challenge of having to deal with approximate stationary points is what makes this problem interesting.

If one is able to compute the global minimum of each Lagrangian sub-problem, in fact, the convergence analysis is straightforward and no assumptions on X, beside its closedness, are required [11]. Computing the global minimum is however, usually, unsuitable with non-convex objective functions and we thus aim for stationary points. Moreover it is often useful and computationally convenient to start with loosely approximate stationary points in the early stages of the algorithm and require increasingly accurate solutions as the algorithm progresses. The convergence of ALM in the latter scenario is very well understood [10, 11] when $X = \mathbb{R}^n$ while the case where additional regularity assumptions can be made on X has been extensively studied in [4, 14]. On the contrary, to the best of our knowledge, no existing results in the literature prove the convergence to stationary points when we deal with abstract convex constraints without any regularity assumptions.

The aim of our analysis it thus to prove that the limit points of the iterates $\{x_k\}$ produced by the ALM algorithm are feasible, i.e. they belong to the convex set

$$\Omega \triangleq \{x \in \mathbb{R}^n \mid x \in X,\ g(x) \leq 0\},$$

and are stationary points of problem (2.1) as expressed by the following.

Definition 1 (Stationary point)**.** We say that $\bar{x} \in \Omega$ is a stationary point if

$$\nabla f(\bar{x})^T (z - \bar{x}) \geq 0 \quad \forall z \in \Omega. \tag{2.3}$$

It is well known that we can equivalently characterize a stationary point in terms of projection. In particular, it holds the following [11].

Lemma 1. A point $\bar{x} \in \Omega$ is a stationary point iff

$$\bar{x} = \Pi_\Omega \left[\bar{x} - \nabla f(\bar{x})\right], \tag{2.4}$$

where $\Pi_\Omega [x]$ is the projection of point x over the convex set Ω.

A crucial aspect of the analysis is how we compute approximate solutions of the Lagrangian sub-problem. Criteria (2.3) and (2.4) for the augmented Lagrangian sub-problem are written as

$$\nabla_x \mathcal{L}_\tau(\bar{x},\ \mu)^T (z - \bar{x}) \geq 0 \quad \forall z \in X$$

and

$$\bar{x} = \Pi_X \left[\bar{x} - \nabla \mathcal{L}_\tau(\bar{x},\ \mu) \right].$$

In view of the equivalent characterization of stationarity offered by the two criteria we can derive two different ways to define ϵ-stationarity as expressed by the following two definitions.

Definition 2 (ϵ-approximate stationary point ASP1)**.** Let $\epsilon > 0$. We say that $\bar{x} \in X$ is an ϵ-approximate stationary point ASP1 for the Lagrangian sub-problem if

$$\nabla_x \mathcal{L}_\tau(\bar{x}, \mu)^T (z - \bar{x}) \geq -\epsilon \quad \forall z \in X. \tag{2.5}$$

Definition 3 (ϵ-approximate stationary point ASP2)**.** Let $\epsilon > 0$. We say that $\bar{x} \in X$ is an ϵ-approximate stationary point ASP2 for the Lagrangian sub-problem if

$$\left\| \bar{x} - \Pi_X \left[\bar{x} - \nabla_x \mathcal{L}_\tau(\bar{x},\ \mu) \right] \right\| \leq \epsilon. \tag{2.6}$$

As it will emerge from the convergence analysis, while we have a perfect equivalence between (2.3) and (2.4), choosing as a stopping criterion for the sub-problems (2.6) in place of (2.5) makes our analysis somewhat more complex and we have to make additional assumptions on the interaction between the sequences $\{\epsilon_k\}$ and $\{\tau_k\}$. On the other hand, criterion ASP1 is in fact entirely suitable only under boundedness assumptions on the feasible set: indeed, in the unbounded case, (2.5) often holds if and only if the current point is also stationary (i.e. ASP1 holds with $\epsilon = 0$).

A possible way to overcome this limitation could be modifying ASP1 to

$$\nabla_x \mathcal{L}_\tau(\bar{x},\ \mu)^T (z - \bar{x}) \geq -\epsilon \left\| z - \bar{x} \right\| \quad \forall z \in X,$$

as it has been done in [47, 88] where the convergence of ALM with such stopping criterion is proven. Note that however, while checking ASP1 is reasonably easy for some classes of problems (for instance solving a LP problem when X is described by linear constraints), it is not possible to check in a finite number of iterations whether the latter holds.

Criterion ASP2, on the other hand is easily checked if one is able to make projections on X and has no limitations in the unbounded case. This criterion has been employed in [14] where ALM in the setting here considered is analyzed to some extent. The results contained in [14] are however limited. In particular from Theorems 6.4 and 6.5 can be deduced the following.

Theorem 1. Let $\{x^k\}$ be the sequence produced by the algorithm using ASP2. Assume further that Ω is not empty. Then each cluster point $\bar{x} = \lim_{k\in K} x^k$ is a feasible point of problem (2.1). Moreover it holds that

$$\lim_{\substack{k\to\infty \\ k\in K}} \left\| x^{k+1} - \Pi_X \left[x^{k+1} - \left(\nabla f(x^{k+1}) + \sum_{i=1}^{p} \nabla g_i(x^{k+1})^T \tilde{\mu}_i^{k+1} \right) \right] \right\| = 0 \tag{2.7}$$

and

$$\lim_{\substack{k\to\infty \\ k\in K}} \min\{-g_i(x^{k+1}),\ \tilde{\mu}_i^{k+1}\} = 0. \tag{2.8}$$

Conditions (2.7) and (2.8), introduced and used in [14], are a kind of sequential (as opposed to point-wise) optimality conditions which approximate KKT conditions and are known as approximate KKTs (AKKTs).

Without further conditions on the constraints, however, AKKTs may hold, even in the case $X = \mathbb{R}^n$, for sequences converging to points that do not satisfy first order optimality conditions. This is shown in [72] by a numerical example in two variables (see Remark 3.1 of [72]) concerning the minimization of a convex function over a convex, compact set. In particular, it is shown that a sequence satisfying an approximate gradient projection property (AGP) converges to a feasible point which is not a minimizer of the convex problem. The AGP property, introduced in [72], is strongly related to AKKTs and in particular we have that AGP implies AKKTs (as discussed in [5]) so that the numerical example holds also for AKKTs.

Moreover in the presence of the abstract set of constraints $X \neq \mathbb{R}^n$, to the best of our knowledge, it has not been proven that AKKTs imply any condition of stationarity even with additional conditions on the constraints.

Therefore a proper convergence analysis of ALM in the considered settings is lacking. We carry out the convergence analysis of the algorithm in the following section.

We would like to point out that, although we do not explicitly treat equality constraints in problem (2.1), it is an easy extension which does not add much to the analysis but worsen the readability of the paper and is, hence, omitted. A special case of linear constraints will be dealt with explicitly in Chapter 3. Note that convex equality constraints can only be linear.

2.2 Convergence analysis

We begin the discussion by stating the following assumptions.

Assumption 1. The set $X \subseteq \mathbb{R}^n$ is closed and convex. The set $\Omega = \{x \in \mathbb{R}^n \mid x \in X,\ g(x) \leq 0\}$ is not empty.

Note that Ω is also convex as it is the intersection of two convex sets.

Assumption 2. The sequence $\{\epsilon_k\}$ is such that

$$\lim_{k\to\infty} \epsilon_k = 0.$$

In the subsequent analysis we will make use of the following very well known Lemma (see for instance [14] Theorem 4.1).

Lemma 2. Let $\{x^{k+1}\}, \{\mu_k\}$ and $\{t_k\}$ be produce by algorithm ALM. Let $\bar{x}$ be a limit point of $\{x^{k+1}\}$, i.e., there exists an infinite subset $K \subseteq \{0, 1, \ldots, \}$ such that

$$\lim_{\substack{k\in K \\ k\to\infty}} x^{k+1} = \bar{x},$$

and suppose that $\bar{x} \in \{x \in \mathbb{R}^n \mid g(x) \leq 0,\ x \in X\}$. Then, for all $i = 1, \ldots, p$ such that $g_i(\bar{x}) < 0$ we have

$$\max\{0,\ \mu_i^k + \tau_k g_i(x^{k+1})\} = 0$$

for all $k \in K$ sufficiently large.

We carry out the analysis for ALM when the ASP1 stopping criterion is employed first.

Proposition 1 (Feasibility for ASP1)**.** Let $\{x^{k+1}\}$ be the sequence produced by algorithm ALM with ASP1. Then each cluster point $\bar{x}$ of $\{x^{k+1}\}$ is a feasible point of problem (2.1), i.e. $\bar{x} \in X$ and $g(\bar{x}) \leq 0$.

Proof. Let $K \subset \{0,\ 1,\ \ldots\}$ such that $\lim_{k\to\infty,\ k\in K} x^{k+1} = \bar{x}$. Consider the two cases:

i) the sequence $\{\tau_k\}$ is bounded,

ii) $\lim_{k\to\infty} \tau_k = \infty$.

Case i): since $\{\tau_k\}$ stays bounded there must exist $\bar{k}$ such that $\|V^{k+1}\| \leq \sigma\|V^k\|$ for all $k > \bar{k}$. This means that

$$\lim_{k\to\infty} \|V^{k+1}\| = 0$$

i.e.

$$\lim_{k\to\infty} V_i^{k+1} = \lim_{k\to\infty} \min\left\{-g_i(x^{k+1}),\ \frac{\mu_i^k}{\tau_k}\right\} = 0,$$

which, since $\mu_i^k \geq 0$ for all i and k, implies that

$$g_i(\bar{x}) = \lim_{\substack{k\to\infty \\ k\in K}} g_i(x^{k+1}) \leq 0.$$

Case ii): from the instructions of the algorithm we have, at each iteration, letting

$$\tilde{g}_i^k = \max\left\{0,\ g_i(x^{k+1}) + \frac{\mu_i^k}{\tau_k}\right\},$$

that

$$\nabla f(x^{k+1})^T(x - x^{k+1}) \geq -\epsilon_k - \tau_k \sum_{i=1}^{p} \tilde{g}_i^k \nabla g_i(x^{k+1})^T(x - x^{k+1}) \tag{2.9}$$

holds for all $x \in X$. Since Ω is not empty we can choose $x \in \{x \in X \mid g(x) \leq 0\}$. Using the convexity of g_i we can bound the last term of (2.9) as follows:

$$\begin{aligned} -\tau_k \sum_{i=1}^{p} \tilde{g}_i^k \nabla g_i(x^{k+1})^T(x - x^{k+1}) &\geq \tau_k \sum_{i=1}^{p} \tilde{g}_i^k (g_i(x^{k+1}) - g_i(x)) \\ &\geq \tau_k \sum_{i=1}^{p} \tilde{g}_i^k g_i(x^{k+1}). \end{aligned}$$

The above inequalities hold since g_i are convex (i.e. $g_i(x) \geq g_i(x^{k+1}) + \nabla g_i(x^{k+1})^T(x - x^{k+1})$) and $\tilde{g}_i^k = \max\{0,\ g_i(x^{k+1}) + \frac{\mu_i^k}{\tau_k}\}$ and $-g(x)$ are nonnegative quantities. Using the latter and dividing (2.9) by τ_k we get

$$\frac{\nabla f(x^{k+1})^T(x - x^{k+1})}{\tau_k} \geq -\frac{\epsilon_k}{\tau_k} + \sum_{i=1}^{p} \tilde{g}_i^k g_i(x^{k+1}). \tag{2.10}$$

Suppose now, by contradiction, that $\{i \in \{1,\ \ldots,\ p\} \mid g_i(\bar{x}) > 0\} \neq \varnothing$. Being $\tau_k \to \infty$, $\{\mu^k\}$ bounded and g continuous, we have that for $k \in K$ sufficiently large

$$\tilde{g}_i^k = \max\left\{0,\ g_i(x^{k+1}) + \frac{\mu_i^k}{\tau_k}\right\} = 0 \quad \forall\, i : g_i(\bar{x}) < 0.$$

Moreover, we have

$$\lim_{k\to\infty, k\in K} \sum_{i\,:\,g_i(\bar{x})=0} \max\left\{0,\ g_i(x^{k+1}) + \frac{\mu_i^k}{\tau_k}\right\} g_i(x^{k+1}) = 0.$$

Thus, taking the limits for $k \in K$, $k \to \infty$ in (2.10), we get

$$\begin{aligned} 0 &\geq \lim_{k\in K, k\to\infty} \sum_{i\,:\,g_i(\bar{x})>0} \max\left\{0,\ g_i(x^{k+1}) + \frac{\mu_i^k}{\tau_k}\right\} g_i(x^{k+1}) \\ &= \sum_{i\,:\,g_i(\bar{x})>0} g_i(\bar{x})^2, \end{aligned}$$

which cannot be. Thus the set $\{i \in \{1,\ \ldots,\ p\} \mid g_i(\bar{x}) > 0\}$ must be empty, i.e. $\bar{x}$ is feasible.

□

Proposition 2 (Optimality for ASP1)**.** Let $\{x^k\}$ be the sequence generated by the algorithm with ASP1. Then, every cluster point of $\{x^{k+1}\}$ is a stationary point of problem (2.1).

Proof. Let $\bar{x}$ be a limit point of $\{x^{k+1}\}$, i.e., there exists an infinite subset $K \subseteq \{0, 1, \ldots, \}$ such that

$$\lim_{k\in K, k\to\infty} x^{k+1} = \bar{x}.$$

From Proposition 1 we have that $\bar{x} \in \Omega$. Suppose by contradiction that there exists $z \in X$ such that $g(z) \leq 0$ and

$$\nabla f(\bar{x})^T (z - \bar{x}) < 0. \tag{2.11}$$

By the instructions of the algorithm we have at each iteration,

$$\begin{aligned} &\nabla f(x^{k+1})^T (z - x^{k+1}) \geq \\ &\quad - \epsilon_k - \tau_k \sum_{i=1}^{p} \max\left\{0,\ g_i(x^{k+1}) + \frac{\mu_i^k}{\tau_k}\right\} \nabla g_i(x^{k+1})^T (z - x^{k+1}). \end{aligned}$$

As similarly done in the proof of Proposition 1 we get

$$\begin{aligned}
&\nabla f(x^{k+1})^T(z - x^{k+1}) \\
&\quad \geq -\epsilon_k + \tau_k \sum_{i=1}^{p} \max\left\{0,\ g_i(x^{k+1}) + \frac{\mu_i^k}{\tau_k}\right\} g_i(x^{k+1}) \\
&\quad = -\epsilon_k + \sum_{i=1}^{p} \max\left\{0,\ \tau_k g_i(x^{k+1}) + \mu_i^k\right\} g_i(x^{k+1}) \\
&\quad = -\epsilon_k + \sum_{i\,:\,g_i(\bar{x})<0} \max\left\{0,\ \tau_k g_i(x^{k+1}) + \mu_i^k\right\} g_i(x^{k+1}) \\
&\qquad + \sum_{i\,:\,g_i(\bar{x})=0} \max\left\{0,\ \tau_k g_i(x^{k+1}) + \mu_i^k\right\} g_i(x^{k+1}).
\end{aligned}$$

From Lemma (2) we have, for $k \in K$ sufficiently large, that

$$\sum_{g_i(\bar{x})<0} \max\{0,\ \tau_k g_i(x^{k+1}) + \mu_i^k\} g_i(x^{k+1}) = 0,$$

and thus

$$\begin{aligned}
&\nabla f(x^{k+1})^T(z - x^{k+1}) \geq \\
&\quad - \epsilon_k + \sum_{i\,:\,g_i(\bar{x})=0} \max\{0,\ \tau_k g_i(x^{k+1}) + \mu_i^k\} g_i(x^{k+1}).
\end{aligned}$$

Now, consider the terms

$$\max\{0,\ \tau_k g_i(x^{k+1}) + \mu_i^k\} g_i(x^{k+1})$$

If $g_i(x^{k+1}) \geq 0$ then

$$\begin{aligned}
&\max\{0,\ \tau_k g_i(x^{k+1}) + \mu_i^k\} g_i(x^{k+1}) \\
&\qquad = \max\{0, \tau_k g_i(x^{k+1})^2 + \mu_i^k g_i(x^{k+1})\} \\
&\qquad \geq \max\{0, \mu_i^k g_i(x^{k+1})\} \\
&\qquad \geq \min\{0, \mu_i^k g_i(x^{k+1})\}.
\end{aligned}$$

Otherwise, if $g_i(x^{k+1}) < 0$, we have

$$\begin{aligned}
&\max\{0,\ \tau_k g_i(x^{k+1}) + \mu_i^k\} g_i(x^{k+1}) \\
&\qquad = \min\{0,\ \tau_k g_i(x^{k+1})^2 + \mu_i^k g_i(x^{k+1})\} \\
&\qquad \geq \min\{0,\ \mu_i^k g_i(x^{k+1})\}.
\end{aligned}$$

Therefore we can write

$$\nabla f(x^{k+1})^T(z - x^{k+1}) \geq -\epsilon_k + \sum_{i\,:\,g_i(\bar{x})=0} \min\{0,\ \mu_i^k g_i(x^{k+1})\}.$$

Taking the limits for $k \in K$, $k \to \infty$, recalling that $\epsilon_k \to 0$, $\{\mu_i^k\}$ are bounded and we are considering the indexes i such that $g_i(x^{k+1}) \to 0$, we get

$$\nabla f(\bar{x})^T(z - \bar{x}) \geq 0.$$

which contradicts (2.11). □

We now turn to ALM with the ASP2 criterion. From [14], Thm 6.5. we can deduce that each limit point of the sequence of iterates is feasible as expressed in the following proposition.

Proposition 3 (Feasibility for ASP2)**.** Let $\{x^{k+1}\}$ be the sequence produced by algorithm ALM with ASP2. Then each cluster point $\bar{x}$ of $\{x^{k+1}\}$ is a feasible point of problem (2.1) i.e. $\bar{x} \in X$ and $g(x) \leq 0$.

Next we prove that limit points are also optimal in the sense of (2.3), or equivalently (2.4).

Proposition 4 (Optimality for ASP2)**.** Let $\{x^k\}$ be the sequence generated by the algorithm ALM using ASP2. Suppose that the sequence $\{\epsilon_k \tau_k\}$ is bounded. Then, every cluster point of $\{x^{k+1}\}$ is a stationary point of problem (2.1).

Proof. Let $\bar{x}$ be a limit point of $\{x^{k+1}\}$, i.e., there exists an infinite subset $K \subseteq \{0, 1, \ldots, \}$ such that

$$\lim_{k \in K, k \to \infty} x^{k+1} = \bar{x}.$$

We know from Proposition 3 that $\bar{x} \in \Omega$. Suppose by contradiction that there exists $z \in X$ such that $g(z) \leq 0$ and

$$\nabla f(\bar{x})^T(z - \bar{x}) < 0. \tag{2.12}$$

From the instructions of the algorithm, letting

$$\tilde{g}_i^k = \max\left\{0,\ g_i(x^{k+1}) + \frac{\mu_i^k}{\tau_k}\right\},$$

and

$$\hat{x}^{k+1} = \Pi_X \left[x^{k+1} - \nabla f(x^{k+1}) - \tau_k \sum_{i=1}^{p} \tilde{g}_i^k \nabla g_i(x^{k+1}) \right],$$

we have that at each iteration

$$\left\| x^{k+1} - \hat{x}^{k+1} \right\| \leq \epsilon_k.$$

From the properties of projection operator we have, for all $x \in X$, that

$$\left(x^{k+1} - \nabla f(x^{k+1}) - \tau_k \left(\sum_{i=1}^{p} \tilde{g}_i^k \nabla g_i(x^{k+1}) \right) - \hat{x}^{k+1} \right)^T \left(x - \hat{x}^{k+1} \right) \leq 0.$$

Adding and subtracting $(x^{k+1} - \hat{x}^{k+1} - \nabla f(x^{k+1}))^T (x^{k+1})$, choosing $x = z$ and rearranging, we get:

$$\begin{aligned} \nabla f(x^{k+1})^T (z - x^{k+1}) \geq & \\ & - \nabla f(x^{k+1})^T (x^{k+1} - \hat{x}^{k+1}) + \|x^{k+1} - \hat{x}^{k+1}\|^2 \\ & + (x^{k+1} - \hat{x}^{k+1})^T (z - x^{k+1}) \\ & - \tau_k \left(\sum_{i=1}^{p} \tilde{g}_i^k \nabla g_i(x^{k+1}) \right)^T (z - \hat{x}^{k+1}) \end{aligned}$$

The last term can be bounded as follows:

$$\begin{aligned} & - \tau_k \left(\sum_{i=1}^{p} \tilde{g}_i^k \nabla g_i(x^{k+1}) \right)^T (z - \hat{x}^{k+1}) \\ & \quad \geq \tau_k \sum_{i=1}^{p} \tilde{g}_i^k (g_i(x^{k+1}) - g_i(z)) - \tau_k \sum_{i=1}^{p} \tilde{g}_i^k \nabla g_i(x^{k+1})^T (x^{k+1} - \hat{x}^{k+1}) \\ & \quad \geq \tau_k \sum_{i=1}^{p} \tilde{g}_i^k g_i(x^{k+1}) - \tau_k \sum_{i=1}^{p} \tilde{g}_i^k \|\nabla g_i(x^{k+1})\| \|x^{k+1} - \hat{x}^{k+1}\| \end{aligned}$$

In the derivations above, we have added and subtracted x^{k+1} to the term $(z - \hat{x}^{k+1})$ and then used the convexity of g_i and $g(z) \leq 0$. The first part of the bound can be further bounded in a similar way as what was done for the ASP1 case. For $k \in K$ sufficiently we have:

$$\begin{aligned} \tau_k \sum_{i=1}^{p} \tilde{g}_i^k g_i(x^{k+1}) &= \tau_k \sum_{i=1}^{p} \max \left\{ 0,\ g_i(x^{k+1}) + \frac{\mu_i^k}{\tau_k} \right\} g_i(x^{k+1}) \\ &\geq \sum_{i\,:\, g_i(\bar{x})=0} \min\{0,\ \mu_i^k g_i(x^{k+1})\}. \end{aligned}$$

For the second part, recalling $\|x^{k+1} - \hat{x}^{k+1}\| \leq \epsilon_k$, we can write

$$-\tau_k \sum_{i=1}^{p} \tilde{g}_i^k \|\nabla g_i(x^{k+1})\| \|x^{k+1} - \hat{x}^{k+1}\| \geq -\tau_k \sum_{i=1}^{p} \tilde{g}_i^k \|\nabla g_i(x^{k+1})\| \epsilon_k. \quad (2.13)$$

Thus, for $k \in K$ sufficiently large, we obtain

$$\begin{aligned} &- \tau_k \left(\sum_{i=1}^{p} \tilde{g}_i^k \nabla g_i(x^{k+1}) \right)^T (z - \hat{x}^{k+1}) \\ &\qquad \geq \sum_{i\,:\,g_i(\bar{x})=0} \min\left\{0, \mu_i^k g_i(x^{k+1})\right\} - \tau_k \epsilon_k \sum_{i=1}^{p} \tilde{g}_i^k \|\nabla g_i(x^{k+1})\| \end{aligned}$$

Putting everything back together we obtain, for $k \in K$ large enough, that

$$\begin{aligned} \nabla f(x^{k+1})^T (z - x^{k+1}) \geq \\ &- \nabla f(x^{k+1})^T (x^{k+1} - \hat{x}^{k+1}) + \|x^{k+1} - \hat{x}^{k+1}\|^2 \\ &+ (x^{k+1} - \hat{x}^{k+1})^T (z - x^{k+1}) \\ &+ \sum_{i\,:\,g_i(\bar{x})=0} \min\{0, \mu_i^k g_i(x^{k+1})\} \\ &- \tau_k \epsilon_k \sum_{i=1}^{p} \tilde{g}_i^k \|\nabla g_i(x^{k+1})\|. \end{aligned}$$

Taking the limits for $k \in K$, $k \to \infty$, recalling that ∇f and ∇g_i, $i = 1, \ldots, p$, are continuous, $\|x^{k+1} - \hat{x}^{k+1}\| \to 0$, $\{\mu^k\}$ is bounded, $\{\tau_k \epsilon_k\}$ is bounded and either $\tau_k \to \infty$ and $\tilde{g}_i^k = \max\{0,\ g_i(x^{k+1}) + \frac{\mu_i^k}{\tau_k}\} \to g_i(\bar{x}) = 0$ or $\tau_k \epsilon_k \to 0$ we get

$$\nabla f(\bar{x})^T (z - \bar{x}) \geq 0,$$

which contradicts (2.12).

□

Remark 1. The role of the boundedness assumption of $\{\tau_k \cdot \epsilon_k\}$ is that of ensuring that, in the case that the penalty parameter τ_k is unbounded, the measure ϵ_k of ASP2 stationarity goes to zero as $1/\tau_k$ or faster. In practice, such condition can be imposed as follows. We can set $\epsilon_{k+1} = \theta \epsilon_k$, with $\theta < 1$. The instructions of the algorithm imply that we have $\tau_{k+1} \leq \alpha \tau_k$ with $\alpha > 1$. Then, the condition holds assuming that $\theta \alpha \leq 1$.

Interestingly in the case of ASP1 criterion of stationarity no additional assumption on the penalty and accuracy parameters is required.

Chapter 3

Two blocks decomposition for the ALM

3.1 Introduction

In this chapter we consider the problem

$$\begin{array}{ll} \min\limits_{x} & f(x) \\ s.t. & x \in X \cap Y, \end{array} \tag{3.1}$$

where X, Y are nonempty, convex, closed sets, and $f : \mathbb{R}^n \to \mathbb{R}$ is a continuously differentiable function. The case of interest is when the constraints defining X and Y are easy to treat when considered separately. Just as an example, consider the problem

$$\begin{array}{ll} \min\limits_{x} & f(x) \\ s.t. & a^T x = b, \\ & l \leq x \leq u, \end{array} \tag{3.2}$$

where $a, l, u \in \mathbb{R}^n$, $l_i < u_i$ for $i = 1, \ldots, n$. Both the sets

$$X = \{x \in \mathbb{R}^n : \ l \leq x \leq u\}, \qquad Y = \{x \in \mathbb{R}^n : \ a^T x = b\}$$

are characterized by "simple" (for instance, in terms of projection) constraints, differently from the feasible set of the original problem (3.2).

With this in mind, let us consider the equivalent reformulation

$$\begin{aligned} \min_{x,y} \quad & f(x) \\ s.t. \quad & x \in X,\ y \in Y, \\ & x - y = 0 \end{aligned}$$

where the special structure of the feasible set has been exploited to form two different blocks of variables with separate and simpler constraints.

A special class of decomposition methods to solve the latter problem is based on the alternate minimization of the augmented Lagrangian function, which is given by

$$\mathcal{L}_\tau(x, y, \lambda) = f(x) + \lambda^T(x - y) + \frac{\tau}{2} \|x - y\|^2,$$

where $\tau > 0$ is the *penalty parameter* and λ the Lagrange multipliers. The special structure of the augmented Lagrangian function makes it possible to decouple the minimizations w.r.t x and y, trying to exploit the individual structures of the sub-problems.

A very popular and effective technique belonging to this class of methods is the Alternating Direction Method of Multipliers (ADMM) [13]. Given the current iterate (x^k, y^k, λ^k), the steps of ADMM are the following

$$x^{k+1} \in \operatorname*{argmin}_{x \in X} f(x) + {\lambda^k}^T(x - y^k) + \frac{\tau}{2} \left\|x - y^k\right\|^2, \tag{3.3}$$

$$y^{k+1} = \operatorname*{argmin}_{y \in Y} {\lambda^k}^T(x^{k+1} - y) + \frac{\tau}{2} \left\|x^{k+1} - y\right\|^2, \tag{3.4}$$

$$\lambda^{k+1} = \lambda^k + \tau(x^{k+1} - y^{k+1}). \tag{3.5}$$

Note that a single iteration of the Gauss-Seidel decomposition method replaces the full (possibly approximated) minimization of the augmented Lagrangian. The theoretical relationship between ALM and ADMM is deeply analyzed in [32]. In the general formulation of ADMM the equality constraint is defined as $Ax + By = c$, by stating suitable assumptions on B. For simplicity, here we restrict to the case $x = y$. Some possible extensions are briefly discussed in a later section.

ADMM has been extensively employed in a wide range of applications. For example the algorithm has been proven to be successful in image restoration [22, 96, 98], matrix factorization [37, 93], wireless networking [50, 86].

Classical proofs of convergence for the ADMM algorithm (see, e.g., [13] and the review in [21]) require the convexity of f and $\tau > 0$ to be fixed.

The value of τ is often a critical factor for the performance of the algorithm in practical applications. Several schemes for varying $\tau = \tau_k$ across the iterations have been proposed in literature [38, 43, 91]. A general rule which comprises most of the approaches [21] is

$$\tau_{k+1} = \begin{cases} \delta_1 \tau_k & \text{if } \left\|x^{k+1} - y^{k+1}\right\| > \mu \left\|y^{k+1} - y^k\right\|, \\ \tau_k / \delta_2 & \text{if } \left\|y^{k+1} - y^k\right\| \quad > \mu \left\|x^{k+1} - y^{k+1}\right\|, \\ \tau_k & \text{otherwise,} \end{cases} \tag{3.6}$$

where $\mu,\ \delta_1,\ \delta_2 > 1$. In practice simply starting from low values of τ and gradually increasing usually works well enough [38, 62]. Convergence, however, is assured only when τ_k is eventually held fixed.

The convergence properties for ADMM in the nonconvex case, instead, are still a vivid topic of research. Convergence to stationary points, in the nonconvex case, have been proved only for special classes of problems although, even in the general nonconvex case, ADMM performs well in practice (see for example [25, 49, 54, 85, 94]). Recently, a family of nonconvex problems, concerning the *consensus* and *sharing* formulations, has been studied in [40] and [92], and convergence results for ADMM have been proved. As noted in these latter papers, there exists a number of works which deal with nonconvex ADMM, for instance [44, 85, 94], which prove the convergence under uncheckable conditions on the sequence of iterates generated by the algorithm. We refer the reader to [40] and [92] for further details.

Note that in the nonconvex case, besides the theoretical issues, the standard ADMM requires to compute a global minimum of a nonconvex problem (see the updating rule of x in (3.3)), and this may be prohibitive in practice.

In this chapter we study a novel algorithm based on the inexact alternate minimization of the augmented Lagrangian, i.e., a modified Gauss-Seidel decomposition strategy, in order to exploit the individual structures of the subproblems. The main peculiarities of the proposed approach are the following: 1) it is not required to compute the global minimum w.r.t. x, but only a point where a "sufficient decrease" is obtained; 2) differently from ADMM, the alternate minimization of the augmented Lagrangian is repeated, before updating the penalty parameter and the multipliers, until an approximated stationary point is attained; 3) the penalty parameter τ_k assumes increasing values. We state a global convergence result without requiring any convexity assumption on f.

3.2 The algorithm

The method can be viewed as an augmented Lagrangian method based on a decomposition strategy. In fact the method can be seen as a specialization of Algorithm 1 for a particular set of constraints and where the sub-problems are solved with an alternate optimization strategy.

Indeed at each iteration k the function $q_{\tau,\lambda}(x,y) \triangleq \mathcal{L}_{\tau_k}(x,y,\lambda^k)$ is considered and alternating (possibly inexact) minimizations with respect to x and y are performed until an approximated stationary point is attained. The above notation on the the augmented Lagrangian is introduced to emphasize that λ and τ are fixed parameters in the alternating minimization process. Then, we update the penalty parameter τ_k (and λ^k) and proceed in solving the resulting new sub-problem until convergence is reached. The key ingredients are, however, how the sub-problem is solved and how τ_k and λ^k are updated. The algorithm may be summarized by the following steps:

(a) given λ^k and τ_k the sub-problem

$$\min_{x\in X,\ y\in Y} q_{\tau_k,\lambda^k}(x,y)$$

is solved approximately (with increasing accuracy) determining the new iterates x^{k+1}, y^{k+1}.

(b) λ^{k+1} is computed by the same rule of ADMM (3.5) appropriately modified to ensure that the updated vector belongs to a prefixed bounded set;

(c) a suitable test is performed in order to decide whether or not the new penalty parameter τ_{k+1} should be larger or remain equal to τ_k.

We now describe in detail these three steps.

Update of x, y The sub-problem is solved to an approximate stationary point in the sense of definition (2.6) (ASP2). More precisely we say that a point $(\bar{x},\bar{y}) \in X \times Y$ is an ϵ-stationary point if

$$\|(\bar{x},\bar{y}) - \Pi_{X\times Y}[(\bar{x},\bar{y}) - \nabla q_{\tau,\lambda}(\bar{x},\bar{y})]\| \leq \epsilon_k, \tag{3.7}$$

where $\epsilon > 0$ and $\Pi : \mathbb{R}^n \times \mathbb{R}^n \to X \times Y$ is the usual projection operator. We propose to find an ϵ-stationary point with a two blocks decomposition

algorithm where at each iteration we update x, keeping y fixed, using any point that yield a sufficient reduction of $q_{\tau,\lambda}$ greater or equal than that obtained by a line search along the projected gradient, and we determine y as the global minimum of $q_{\tau,\lambda}$ keeping x fixed. This scheme can be viewed as a modified two-blocks Gauss-Seidel algorithm based on a line search along a gradient-related direction. Note that the x step allows for different implementations in practical applications preserving convergence properties. For instance, one could use a closed form solution, if available, or employ a specialized solver to find a stationary point (w.r.t. x) to some degree of accuracy. We have reported in the appendix the scheme of the standard Armijo-type line search and its well-known theoretical properties used in the convergence analysis.

Algorithm 2: Alternate Minimization (AM)

Input: $x^0 \in X$, $y^0 \in Y, \epsilon > 0$.

1 $t = 0$.

2 **while** (x^t, y^t) *is not an* ϵ*-stationary point* **do**

3 find x^{t+1} such that

$$q_{\tau,\lambda}(x^{t+1}, y^t) \leq q_{\tau,\lambda}(x^t + \alpha_t d^t, y^t),$$

where $d^t = \Pi_X[x^t - \nabla_x q_{\tau,\lambda}(x^t, y^t)] - x^t$ and α_t is obtained with an Armijo-type line search.

4 find y^{t+1} such that

$$y^{t+1} = \operatorname*{argmin}_{y \in Y} q_{\tau,\lambda}(x^{t+1}, y).$$

5 $t = t + 1$.

Update of λ Once we have determined an ϵ-stationary point of the subproblem we update λ. We modify the update rule to assure that the sequence $\{\lambda^k\}$ is bounded. The proposed update rule is

$$\tilde{\lambda}^{k+1} = \lambda^k + \tau_k(x^{k+1} - y^{k+1}),$$
$$\lambda^{k+1} = \Pi_{[\lambda_m, \lambda_M]}\left[\tilde{\lambda}^{k+1}\right]$$

where both the operations are intended element-wise and $-\infty < \lambda_m < \lambda_M < +\infty$ are two fixed parameters.

Update of τ Finally we update τ_k with

$$\tau_{k+1} = \begin{cases} \tau_k & \text{if } \left\|x^{k+1} - y^{k+1}\right\| \leq \sigma \left\|x^k - y^k\right\|, \\ \delta\tau_k & \text{otherwise,} \end{cases}$$

where $\sigma < 1$ and $\delta > 1$. Note that this scheme is different from (3.6) as we use a condition on the decrease of the distance (in x and y) between successive iterates. Note also that τ_k can never decrease.

The complete procedure is shown in Algorithm 3.

Algorithm 3: ALTernating Augmented Lagrangian Method (ALTALM)

Input: $(x^0,\ y^0) \in X \times Y,\ \lambda^0$
$\delta > 1,\ \sigma \in (0,\ 1),\ -\infty < \lambda_m < \lambda_M < +\infty,\ \tau_0 > 0,$
a positive sequence $\{\epsilon_k\}$ s.t. $\lim_{k\to+\infty} \epsilon_k = 0$.

for $k = 0,\ 1,\ \ldots$ **do**
  compute an ϵ_k-stationary point (x^{k+1}, y^{k+1}) by Algorithm AM.
  $\tilde{\lambda}^{k+1} = \lambda^k + \tau_k(x^{k+1} - y^{k+1})$
  $\lambda^{k+1} = \Pi_{[\lambda_m, \lambda_M]}\left[\tilde{\lambda}^{k+1}\right]$
  if $\left\|x^{k+1} - y^{k+1}\right\| \leq \sigma \left\|x^k - y^k\right\|$ **then**
    $\tau_{k+1} = \tau_k,$
  else
    $\tau_{k+1} = \delta\tau_k.$

3.3 Convergence analysis

In the convergence analysis we first prove (see Proposition 6) that the algorithm is well defined, i.e., that AM algorithm attains an ϵ-stationary point in a finite number of iterations. To this aim, we need a preliminary result (see Proposition 5). The global convergence of ALTALM is stated by Theorems 2, 3.

We state the following assumptions.

Assumption 3. The sets $X \subseteq \mathbb{R}^n$, $Y \subseteq \mathbb{R}^n$ are convex, closed and nonempty.

Assumption 4. The function $f : \mathbb{R}^n \to \mathbb{R}$ is continuously differentiable and coercive on X, i.e., for all sequences $\{x^k\}$ such that $x^k \in X$ and

$$\lim_{k\to+\infty} ||x^k|| = +\infty$$

we have

$$\lim_{k\to+\infty} f(x^k) = +\infty.$$

Proposition 5. The function $q_{\tau,\lambda}(x, y)$ is coercive on $X \times Y$ for any finite value of τ, λ.

Proof. Let us consider any pair of sequences $\{x^k\}$ and $\{y^k\}$ such that $x^k \in X$, $y^k \in Y$ for all k, and at least one of the following conditions holds

$$\lim_{k\to+\infty} \|x^k\| = +\infty, \tag{3.8}$$

$$\lim_{k\to+\infty} \|y^k\| = +\infty. \tag{3.9}$$

Assume by contradiction that there exists an infinite subset $K \subseteq \{0, 1, \dots, \}$ such that

$$\limsup_{k\in K, k\to+\infty} q_{\tau,\lambda}(x^k, y^k) \neq +\infty. \tag{3.10}$$

Suppose first that there exists an infinite subset $K_1 \subseteq K$ such that

$$\|x^k - y^k\| \leq M \tag{3.11}$$

for some $M > 0$ and for all $k \in K_1$. Recalling that f is coercive on X, from (3.8), (3.9) we have that $f(x^k) \to +\infty$ for $k \in K_1, k \to +\infty$. From (3.11), as

$$\lambda^T(x^k - y^k) \geq -\|x^k - y^k\| \, \|\lambda\| \,,$$

we obtain

$$\lim_{k\in K_1, k\to+\infty} q_{\tau,\lambda}(x^k, y^k) = \lim_{k\in K_1, k\to+\infty} f(x^k) + \lambda^T(x^k - y^k) + \frac{\tau}{2} \left\|x^k - y^k\right\|^2 = +\infty,$$

and this contradicts (3.10).

Then we must have

$$\lim_{k\in K, k\to+\infty} \|x^k - y^k\| = +\infty.$$

As f is coercive and continuous, it is also bounded below on X. Thus, we have

$$q_{\tau,\lambda}(x^k, y^k) \geq \bar{f} - \|\lambda\|\|x^k - y^k\| + \frac{\tau}{2}\left\|x^k - y^k\right\|^2,$$

which implies that $q_{\tau,\lambda}(x^k, y^k) \to +\infty$ for $k \in K, k \to +\infty$.

Then, we can conclude that for any infinite set K we have

$$\lim_{k\in K, k\to+\infty} q_{\tau,\lambda}(x^k, y^k) = +\infty,$$

and this contradicts (3.10). □

In the next proposition we show that Algorithm AM determines in a finite number of iterations an ϵ-stationary point. The proof technically exploits the fact that the x iterate has function value lower or equal than the function value obtainable through an Armijo-type line search performed along the descent direction computed by the projected gradient method. Therefore, we have reported in the appendix, besides the scheme of the standard Armijo-type line search, its well-known theoretical properties (Proposition 12) used in the proof of the proposition.

Proposition 6. Algorithm AM determines in a finite number of iterations a point $(x(\epsilon), y(\epsilon))$ which is an ϵ-stationary point for problem

$$\min_{x\in X, y\in Y} q_{\tau,\lambda}(x, y).$$

Proof. Suppose Algorithm AM generates an infinite sequence $\{x^t, y^t\}$. The instructions of the algorithm imply

$$q_{\tau,\lambda}(x^t, y^t) \leq q_{\tau,\lambda}(x^0, y^0),$$

and hence, for all $t \geq 0$ the point (x^t, y^t) belongs to the level set $\mathcal{L}_0 = \{(x, y) \in X \times Y \mid q_{\tau,\lambda}(x, y) \leq q(x^0, y^0)\}$. From Proposition 5 it follows that $\mathcal{L}_0$ is a compact set. Therefore, the sequence $\{x^t, y^t\}$ admits cluster points. Let $(\bar{x}, \bar{y})$ be a cluster point of $\{x^t, y^t\}$, i.e., there exists $T \subseteq \{0, 1, \ldots, \}$ such that

$$\lim_{t\in T, t\to+\infty} (x^t, y^t) = (\bar{x}, \bar{y}).$$

We first show that $(\bar{x}, \bar{y})$ is a critical point, i.e.,

$$(\bar{x}, \bar{y}) = \Pi_{X \times Y}[(\bar{x}, \bar{y}) - \nabla q_{\tau,\lambda}(\bar{x}, \bar{y})]. \tag{3.12}$$

From the instructions of the algorithm, as

$$y^t = \arg\min_{y \in Y} q_{\tau,\lambda}(x^t, y),$$

we get

$$y^t = \Pi_Y[y^t - \nabla_y q_{\tau,\lambda}(x^t, y^t)],$$

from which, taking the limits for $t \in T$ and $t \to +\infty$, recalling the continuity of the gradient and of the projection operator, we obtain

$$\bar{y} = \Pi_Y[\bar{y} - \nabla_y q_{\tau,\lambda}(\bar{x}, \bar{y})]. \tag{3.13}$$

Let $\hat{x}^t = \Pi_X[x^t - \nabla_x q_{\tau,\lambda}(x^t, y^t)]$, so that $d^t = \hat{x}^t - x^t$. The instructions of the algorithm imply

$$q_{\tau,\lambda}(x^{t+1}, y^{t+1}) \leq q_{\tau,\lambda}(x^{t+1}, y^t) \leq q_{\tau,\lambda}(x^t + \alpha_t d^t, y^t) < q_{\tau,\lambda}(x^t, y^t). \tag{3.14}$$

Recalling the continuity of the gradient and the projection operator, it follows $d^t \to \bar{d}$ for $t \in T$, $t \to +\infty$, which implies that $\|d^t\| \leq M$ for some $M > 0$ and for $t \in T$. The sequence $\{q_{\tau,\lambda}(x^t, y^t)\}$ is monotone decreasing, $q_{\tau,\lambda}(x, y)$ is continuous, and hence we have

$$\lim_{t \to +\infty} q_{\tau,\lambda}(x^t, y^t) = q_{\tau,\lambda}(\bar{x}, \bar{y}).$$

From (3.14) it follows

$$\lim_{t \in T, t \to +\infty} q_{\tau,\lambda}(x^t, y^t) - q_{\tau,\lambda}(x^t + \alpha_t d^t, y^t) = 0.$$

Then, the hypothesis of Proposition 12 are satisfied and hence we obtain

$$\lim_{t \in T, t \to +\infty} \nabla_x q(x^t, y^t)^T d^t = 0. \tag{3.15}$$

By the properties of the projection operator we have

$$(x^t - \nabla_x q_{\tau,\lambda}(x^t, y^t) - \hat{x}^t)^T (x - \hat{x}^t) \leq 0 \quad \forall x \in X,$$

and hence we can write

$$\nabla_x q_{\tau,\lambda}(x^t, y^t)^T d^t \leq -\left\|x^t - \hat{x}^t\right\|^2. \tag{3.16}$$

Thus, from (3.15) and (3.16), recalling the continuity of the gradient and of the projection operator we obtain

$$\bar{x} = \Pi_X[\bar{x} - \nabla_x q_{\tau,\lambda}(\bar{x}, \bar{y})]. \tag{3.17}$$

From (3.17) and (3.13) it follows that (3.12) holds.

Then, by the continuity of $\nabla q_{\tau,\lambda}$ and $\Pi_{X\times Y}$, it follows that for t sufficiently large we have

$$\|(x^t, y^t) - \Pi_{X\times Y}[(x^t, y^t) - \nabla q_{\tau,\lambda}(x^t, y^t)]\| \leq \epsilon,$$

i.e., (x^t, y^t) is an ϵ-stationary point.

□

We now state the main convergence result.

First we prove that the sequence $\{x^k, y^k\}$ admits cluster points and each cluster point $\{\bar{x}, \bar{y}\}$ is such that $\bar{x} = \bar{y}$ and $\bar{x} \in X \cap Y$ in Theorem 2. Then we prove that $\bar{x}$ is also a stationary point for the original problem in Theorem 3. Note that Theorem 2 and 3 roughly corresponds to Theorems 3 and 4 which however dealt with general inequality constraints in place of the special case of linear constraints considered here.

We make the following additional assumption.

Assumption 5. At least one of following conditions holds:

(a) the set X is compact;

(b) the set Y is compact and $\|\nabla f(x)\| \leq M$, with $M > 0$, for all $x \in X$.

By adapting the proof of Theorem 6.5 in [15] we can prove the following result.

Theorem 2. Let $\{(x^k, y^k)\}$ be the sequence generated by Algorithm ALTALM. Then, the sequence $\{(x^k, y^k)\}$ admits cluster points and every cluster point $(\bar{x}, \bar{y})$ is such that $\bar{x} = \bar{y} \in X \cap Y$.

Proof. Suppose first that (a) holds. The sequence $\{x^k\}$ belongs to X and hence admits cluster points. In order to prove that $\{(x^k, y^k)\}$ admits cluster points it is sufficient to show that

$$\lim_{k\to+\infty} \|x^{k+1} - y^{k+1}\| = 0. \tag{3.18}$$

The instructions of the algorithm imply

$$\tau_{k+1} \geq \tau_k,$$

so that either $\tau_k \to +\infty$ for $k \to +\infty$ or $\tau_k \to \bar{\tau} < +\infty$ for $k \to +\infty$. In the latter case we necessarily have $\tau_k = \bar{\tau}$ for k sufficiently large, i.e., for $k \geq \bar{k}$. Then, if the sequence $\{\tau_k\}$ is bounded, taking into account the updating rule of τ_k, for $k \geq \bar{k}$ we can write

$$\|x^{k+1} - y^{k+1}\| \leq (\sigma)^{k+1-\bar{k}} \|x^{\bar{k}} - y^{\bar{k}}\|,$$

and this implies that (3.18) holds.

Consider the case that $\tau_k \to +\infty$ for $k \to +\infty$. The instructions of the algorithm imply

$$\|x^{k+1} - \Pi_X[x^{k+1} - \nabla_x q_{\tau_k,\lambda_k}(x^{k+1}, y^{k+1})]\| \leq \epsilon_k \tag{3.19}$$

where

$$\nabla_x q_{\tau_k,\lambda_k}(x^{k+1}, y^{k+1}) = \nabla f(x^{k+1}) + \lambda_k + \tau_k(x^{k+1} - y^{k+1}),$$

and

$$\|y^{k+1} - \Pi_Y[y^{k+1} - \nabla_y q_{\tau_k,\lambda_k}(x^{k+1}, y^{k+1})]\| \leq \epsilon_k, \tag{3.20}$$

where

$$\nabla_y q_{\tau_k,\lambda_k}(x^{k+1}, y^{k+1}) = -\lambda_k - \tau_k(x^{k+1} - y^{k+1}).$$

Since $\tau_k \to +\infty$ for $k \to +\infty$, we have $1/\tau_k < 1$ for k sufficiently large. From (3.19) and (3.20), using the nondecreasing property of projections, we can write

$$\|x^{k+1} - \Pi_X[x^{k+1} - \nabla f(x^{k+1})/\tau_k - \lambda_k/\tau_k - (x^{k+1} - y^{k+1})]\| \leq \epsilon_k, \tag{3.21}$$

$$\|y^{k+1} - \Pi_Y[y^{k+1} + \lambda_k/\tau_k + (x^{k+1} - y^{k+1})]\| \leq \epsilon_k. \tag{3.22}$$

Assume by contradiction that (3.18) does not hold. Then, there exists an infinite subset $K \subseteq \{0, 1, \ldots\}$ such that

$$\|x^{k+1} - y^{k+1}\| \geq \eta > 0 \qquad \forall k \in K. \tag{3.23}$$

As $\{x^k\}$ belongs to the compact set X, there exists a further subset $K_1 \subseteq K$ such that

$$\lim_{k \in K_1, k \to +\infty} x^{k+1} = \hat{x}.$$

Since $\{\lambda^k\}$ is a bounded sequence and $\epsilon_k \to 0$ for $k \to +\infty$, taking into account the continuity of the projection, from (3.22) we get

$$\lim_{k\in K_1, k\to+\infty} \|y^{k+1} - \Pi_Y(x^{k+1})\| = 0,$$

which implies

$$\lim_{k\in K_1, k\to+\infty} y^{k+1} = \hat{y} = \Pi_Y(\hat{x}).$$

From (3.21) and (3.22) we get

$$\|\hat{x} - \Pi_X[\hat{x} - (\hat{x} - \hat{y})]\| = 0 \qquad \text{and} \qquad \|\hat{y} - \Pi_Y[\hat{y} - (\hat{y} - \hat{x})]\| = 0,$$

i.e., that $(\hat{x}, \hat{y})$ is a solution of the convex problem

$$\min_{x\in X, y\in Y} \frac{1}{2}\|x - y\|^2.$$

Since $X \cap Y \neq \varnothing$, it follows that a point $(x^\star, y^\star)$ is a solution of the above problem if and only if $x^\star = y^\star$. Then, we have $\hat{x} = \hat{y}$, which implies that for $k \in K_1$ and k sufficiently large we can write

$$\|x^{k+1} - y^{k+1}\| \leq \eta/2,$$

and this contradicts (3.23) and hence the thesis is proven under assumption (a).

Suppose now that (b) holds. The sequence $\{y^k\}$ belongs to Y and hence admits cluster points. In order to prove that $\{(x^k, y^k)\}$ admits cluster points it is sufficient to show that (3.18) holds.

Reasoning as in the preceding case we can prove (3.21) and (3.22).

Assume by contradiction that (3.18) does not hold. Then, there exists an infinite subset $K \subseteq \{0, 1, \ldots\}$ such that (3.23) is satisfied for all $k \in K$. As $\{y^k\}$ belongs to the compact set Y, there exists a further subset $K_1 \subseteq K$ such that

$$\lim_{k\in K_1, k\to+\infty} y^{k+1} = \hat{y}.$$

Since $\{\lambda^k\}$ and $\{\nabla f(x^k)\}$ are bounded sequences, and $\epsilon_k \to 0$ for $k \to +\infty$, taking into account the continuity of the projection, from (3.21) we get

$$\lim_{k\in K_1, k\to+\infty} \|x^{k+1} - \Pi_X(y^{k+1})\| = 0,$$

which implies

$$\lim_{k\in K_1, k\to+\infty} x^{k+1} = \hat{x} = \Pi_X(\hat{y}).$$

The rest of the proof is similar to case (a).

□

In order to prove the global convergence of Algorithm ALTALM we need to introduce, as stated in the next theorem, a further condition on the parameter ϵ_k (related to the degree of approximation in solving the sub-problems) and on the penalty parameter τ_k that may tend to infinity.

Theorem 3. Let $\{(x^k, y^k)\}$ be the sequence generated by Algorithm ALTALM. Suppose that the sequence $\{\epsilon_k \tau_k\}$ is bounded. Then, every cluster point $(\bar{x}, \bar{y})$ is such that $\bar{x} = \bar{y}$, and $\bar{x}$ is a stationary point of problem (3.1), i.e.,

$$\nabla f(\bar{x})^T (z - \bar{x}) \geq 0 \quad \forall z \in X \cap Y. \tag{3.24}$$

Proof. By Theorem 2 the sequence $\{x^{k+1}, y^{k+1}\}$ admits cluster points. Let $(\bar{x}, \bar{y})$ be a cluster point of $\{(x^{k+1}, y^{k+1})\}$, i.e., there exists an infinite subset $K \subseteq \{0, 1, \dots, \}$ such that

$$\lim_{k\in K, k\to+\infty} (x^{k+1}, y^{k+1}) = (\bar{x}, \bar{y}),$$

where $\bar{x} = \bar{y} \in X \cap Y$. In order to prove (3.24), assume by contradiction that it does not hold, that is, there exists a point $\hat{z} \in X \cap Y$ such that

$$\nabla f(\bar{x})^T (\hat{z} - \bar{x}) < 0. \tag{3.25}$$

Let

$$(\hat{x}^{k+1}, \hat{y}^{k+1}) = \Pi_{X\times Y}((x^{k+1}, y^{k+1}) - \nabla q_{\tau_k, \lambda_k}(x^{k+1}, y^{k+1})).$$

By the properties of the projection we have

$$\begin{aligned}
&\left[\hat{x}^{k+1} - (x^{k+1} - \nabla_x q_{\tau_k,\lambda_k}(x^{k+1}, y^{k+1}))\right]^T (\hat{z} - \hat{x}^{k+1}) + \\
&\left[\hat{y}^{k+1} - (y^{k+1} - \nabla_y q_{\tau_k,\lambda_k}(x^{k+1}, y^{k+1}))\right]^T (\hat{z} - \hat{y}^{k+1}) \geq 0,
\end{aligned}$$

from which it follows

$$\begin{aligned}
&\nabla f(x^{k+1})^T (\hat{z} - \hat{x}^{k+1}) + \lambda^{k^T} (\hat{z} - \hat{x}^{k+1}) + \tau_k (x^{k+1} - y^{k+1})^T (\hat{z} - \hat{x}^{k+1}) \\
&\quad + \lambda^{k^T} (\hat{y}^{k+1} - \hat{z}) + \tau_k (x^{k+1} - y^{k+1})^T (\hat{y}^{k+1} - \hat{z}) \\
&\qquad \geq -(\hat{x}^{k+1} - x^{k+1})^T (\hat{z} - \hat{x}^{k+1}) - (\hat{y}^{k+1} - y^{k+1})^T (\hat{z} - \hat{y}^{k+1}),
\end{aligned}$$

which can be rewritten as

$$\nabla f(x^{k+1})^T(\hat{z}-\hat{x}^{k+1}) \geq (x^{k+1}-\hat{x}^{k+1})^T(\hat{z}-\hat{x}^{k+1}) + (y^{k+1}-\hat{y}^{k+1})^T(\hat{z}-\hat{y}^{k+1}) + \lambda^{k^T}(\hat{x}^{k+1}-\hat{y}^{k+1}) + \tau_k(x^{k+1}-y^{k+1})^T(\hat{x}^{k+1}-\hat{y}^{k+1}).$$

The instructions of the algorithm imply $\left\|x^{k+1}-\hat{x}^{k+1}\right\| \leq \epsilon_k$, $\left\|y^{k+1}-\hat{y}^{k+1}\right\| \leq \epsilon_k$, so that we have

$$(x^{k+1}-\hat{x}^{k+1})^T(\hat{z}-\hat{x}^{k+1}) \geq -\epsilon_k \left\|\hat{z}-\hat{x}^{k+1}\right\|,$$
$$(y^{k+1}-\hat{y}^{k+1})^T(\hat{z}-\hat{y}^{k+1}) \geq -\epsilon_k \left\|\hat{z}-\hat{y}^{k+1}\right\|.$$

Thus we can write

$$\nabla f(x^{k+1})^T(\hat{z}-\hat{x}^{k+1}) \geq -\epsilon_k \left(\left\|\hat{z}-\hat{x}^{k+1}\right\| + \left\|\hat{z}-\hat{y}^{k+1}\right\|\right) + \lambda^{k^T}(\hat{x}^{k+1}-\hat{y}^{k+1}) + \tau_k(x^{k+1}-y^{k+1})^T(\hat{x}^{k+1}-\hat{y}^{k+1}). \tag{3.26}$$

Note that the last term in the latter inequality can be bounded as follows

$$\begin{aligned}
&\tau_k(x^{k+1}-y^{k+1})^T(\hat{x}^{k+1}-\hat{y}^{k+1}) \\
&\quad = \tau_k(x^{k+1}-y^{k+1})^T(\hat{x}^{k+1}-\hat{y}^{k+1}+x^{k+1}-y^{k+1}-x^{k+1}+y^{k+1}) \\
&\quad = \tau_k\left\|x^{k+1}-y^{k+1}\right\|^2 + \tau_k(x^{k+1}-y^{k+1})^T(\hat{x}^{k+1}-x^{k+1}) \\
&\quad - \tau_k(x^{k+1}-y^{k+1})^T(\hat{y}^{k+1}-y^{k+1}) \\
&\quad \geq -2\tau_k\cdot\epsilon_k\left\|x^{k+1}-y^{k+1}\right\|.
\end{aligned}$$

Recalling that $\|\hat{x}^{k+1}-x^{k+1}\| \to 0$, $\|\hat{y}^{k+1}-y^{k+1}\| \to 0$, $\|x^{k+1}-y^{k+1}\| \to 0$, we have that both $\hat{x}^{k+1}$ and $\hat{y}^{k+1}$ converge to $\bar{x} \in X \cap Y$. Thus, taking limits in (3.26) for $k \in K$ and $k \to \infty$, recalling that $\{\lambda^k\}$ and $\{\tau_k \cdot \epsilon_k\}$ are bounded we obtain

$$\nabla f(\bar{x})^T(\hat{z}-\bar{x}) \geq 0,$$

and this contradicts (3.25). □

3.4 Extensions

Here we discuss how to modify the algorithm in order to be able to deal with the case of a feasible set partitioned in more than two subsets. We consider, thus, the problem:

$$\begin{aligned}
\min_x \quad & f(x) \\
s.t. \quad & x \in X_1 \cap X_2 \cap \cdots \cap X_N,
\end{aligned}$$

which is equivalently stated as

$$\begin{array}{ll} \min\limits_{x} & f(x) \\ s.t. & x_i \in X_i \quad i \in [1,\dots,N], \\ & x_i = x_1 \quad i \in [2,\dots,N]. \end{array} \tag{3.27}$$

We can apply the two-block version of the algorithm (both ADMM and ALTALM) to the problem (3.27), where the two blocks are identified by $x = x_1$ and $y = (x_2,\dots,x_N)$. The minimization step w.r.t. to the second block is defined as follows:

$$(x_2^{t+1},\dots,x_N^{t+1}) \in \operatorname*{argmin}_{x_2\in X_2,\dots,x_n\in X_N} q_{\tau_k,\lambda^k}(x^t,y) = \sum_{i=2}^{N} {\lambda_i^k}^T (x_i - x_1^t) + \frac{\tau_k}{2} \left\|x_i - x_1^t\right\|^2,$$

where t is the counter of the inner iterations of Algorithm AM. Such problem can be decomposed into $N-1$ separable problems and hence solved (possibly in parallel) by performing $N-1$ separate minimizations. Hence the second minimization step can be replaced by the following $N-1$ steps

$$x_i \in \operatorname*{argmin}_{x_i\in X_i} {\lambda_i^k}^T (x_i - x_1^t) + \frac{\tau_k}{2} \left\|x_i - x_1^t\right\|^2 \quad i \in [2,\dots,N].$$

Finally, as second extension, we can replace the constraint $x = y$ by the more general constraint $Ax + By = c$, where $x \in \mathbb{R}^n$, $y \in \mathbb{R}^m$, $c \in \mathbb{R}^d$, $A \in \mathbb{R}^d \times \mathbb{R}^n$ and $B \in \mathbb{R}^d \times \mathbb{R}^m$. The sub-problem is modified as follows

$$\begin{array}{lll} \min\limits_{x,y} & q_{\tau,\lambda}(x,y) \triangleq & f(x) + \lambda^T(Ax + By - c) + \frac{\tau}{2} \|Ax + By - c\|_2^2 \\ \text{s.t.} & & x \in X, \\ & & y \in Y. \end{array}$$

Under the assumption (often stated in the literature) that B has full column rank, all the convergence results hold with no or minor modifications.

3.5 A computational example

In this section we show, as an example, a nonconvex problem which can be solved by means of ALTALM. The approach we propose has a resemblance to ADMM, so we compare its performance with ADMM and its modifications which employ a varying τ. We do not intend this to be an extensive numerical study but only a computational experiment to assess the practical advantages of the modifications introduced by the proposed approach.

We consider the Support Vector Machine (SVM) training problem for classification tasks [83]. In particular, given a training set of input-target pairs $(u_i, a_i), i = 1, \ldots, n$ with $u_i \in \mathbb{R}^d$, and $a_i \in \{-1, 1\}$, the constrained optimization problem for SVM training takes the following form

$$\min_x f(x) \triangleq \frac{1}{2} x^T Q x - e^T x \tag{3.28}$$

$$\text{s.t. } a^T x = 0, \tag{3.29}$$

$$0 \leq x_i \leq C \quad \forall\, i \in 1, \ldots, n, \tag{3.30}$$

where e is the vector of all ones and $Q \in \mathbb{R}^{n \times n}$. The generic element q_{ij} of the matrix Q is given by $a_i a_j K(u_i, u_j)$, where $K(u, z) = \phi(u)^T \phi(z)$ is the kernel function related to the nonlinear function ϕ that maps the data from the input space R^d onto the feature space. The matrix Q measures similarity between data points and must be positive semidefinite to satisfy the Mercer's condition. However, in the literature there are kernel matrices formed using similarity measures which are not positive semidefinite, and therefore the above problem becomes nonconvex. We will focus on nonconvex problems obtained by the following indefinite kernel functions proposed in the SVM literature (see, e.g., [76]).

- The *sigmoid kernel*:

  $$K(u, v) = \tanh(a u^T v + b),$$

  with either $a < 0$ or $b < 0$; in our experiments we set $a = 0.2$ and $b = -1$.

- The *gaussian combination kernel*:

  $$k(u, v) = \exp\left(-\frac{\|u - v\|^2}{\sigma_1}\right) + \exp\left(-\frac{\|u - v\|^2}{\sigma_2}\right) - \exp\left(-\frac{\|u - v\|^2}{\sigma_3}\right).$$

  In our experiments we set $\sigma_1 = \sigma_2 = 0.01$, $\sigma_3 = 100$.

Assigning x to constraint (3.30) and y to constraint (3.29), the augmented Lagrangian-based formulation is the following

$$\begin{aligned} &\min_{x,y} q_{\tau,\lambda}(x, y) \triangleq \frac{1}{2} x^T Q x - e^T x + \lambda^T (x - y) + \frac{\tau}{2} \|x - y\|^2 \\ &\text{s.t. } a^T y = 0, \\ &\quad 0 \leq x_i \leq C \quad \forall\, i \in 1, \ldots, n. \end{aligned}$$

Note that, as the set $X = \{x \in \mathbb{R}^n : 0 \leq x \leq C\}$ is compact, the existence of cluster points of the sequence $\{(x^k, y^k)\}$ generated by Algorithm ALTALM is ensured by Theorem 2.

The solution to the sub-problem w.r.t. y can be easily obtained in closed form by the KKT conditions:

$$\begin{aligned} -\lambda - \tau(x - y^*) + \mu^* a &= 0, \\ a^T y^* &= 0, \end{aligned}$$

from which we obtain

$$\begin{aligned} \mu^* &= \frac{a^T \lambda + \tau a^T x}{\|a\|^2}, \\ y^* &= \frac{\lambda - \mu^\star a}{\tau} + x. \end{aligned}$$

As for the sub-problem w.r.t. x, there are many solvers for quadratic bound-constrained problems. In our implementation we used the BLEIC algorithm from [18] checking the condition of "sufficient decrease" at Step 3 of Algorithm AM and stopping the algorithm when an approximate stationary point w.r.t. x has been attained.

The training set used for the experiments is the *liver-disorders* dataset [55], where the number n of instances is 145 and the number d of features is 5.

The parameter C of (3.30) has been set to the default value of 1. The stopping criterion for all of the tested algorithms has been defined taking into account the optimality conditions for problem (3.28)-(3.30) reported below (see, e.g., [58]).

Proposition 7. A feasible point $x^\star$ is a solution of problem (3.28)-(3.30) only if

$$m(x^\star) \leq M(x^\star),$$

where

$$\begin{aligned} m(x^\star) &= \max\{\max_{x_i^\star < C, a_i < 0} -\nabla f(x^\star)_i, \max_{x_i^\star > 0, a_i > 0} \nabla f(x^\star)_i\}, \\ M(x^\star) &= \min\{\min_{x_i^\star < C, a_i > 0} \nabla f(x^\star)_i, \min_{x_i^\star > 0, a_i < 0} -\nabla f(x^\star)_i\}. \end{aligned}$$

Notice that both x_k and y_k could be chosen as the final output of the algorithm. We do however choose x^k as the final solution. In this way the

box constraint is always satisfied by design and we can measure the feasibility violation with the quantity $|a^T x^k|$. Thus, on the basis of Proposition 7, a suitable stopping criterion is the following

$$|a^T x| \leq 10^{-3}, \tag{3.31}$$

$$m(x^k) \leq M(x^k) + 10^{-2}. \tag{3.32}$$

Note that there exist several specialized and very efficient algorithms to solve the *SVM training* problem, even in the indefinite, nonconvex case (see for instance [26]). Here, however, we do not compare our algorithm with the state-of-art algorithms. We perform, instead, a comparison of

- standard ADMM (fixed τ),
- ADMM with adaptive τ_k as in (3.6) with $\mu = 10, \alpha_1 = \alpha_2 = 2$ as suggested in [21],
- ADMM with increasing τ_k obtained setting $\mu = 10, \alpha_1 = 2, \alpha_2 = 1$,
- ALTALM with $\sigma = 0.99$, $\alpha = 1.0003$. As stopping criterion for the AM algorithm we employ (3.7). The sequence $\{\epsilon_k\}$ is computed as $\epsilon_0 = 100$, $\epsilon_{k+1} = (1 - 3 \cdot 10^{-4})\epsilon_k$.

For all of the algorithms the initial point is $0 \in \mathbb{R}^n$, as usual for SVM problems, and λ^0 is set to 0. We say that a run is successful if it satisfies the stopping criterion (3.31), (3.32) in at most 30000 iterations. For each kernel we run the algorithms for different values of the parameter τ_0 in $\{2^n \mid n = 0, 1, \ldots, 12\}$.

The comparison is carried out in number of iterations. Note that this is a sound comparison, since all of the algorithms, at each iteration, perform the same exact minimization steps, one w.r.t x, one w.r.t. y (with possibly different values of λ and τ).

In Figures 3.1-3.2, we report the results, for the two used kernel functions, in terms of number of iterations. We consider both successful and unsuccessful runs. Thus, when the number of iterations is 30000 the run was unsuccessful.

The obtained results clearly show the effectiveness of the proposed ALTALM with respect to the other ADMM methods. Incremental ADMM sometimes also proved to be effective, but still less efficient. The other two methods were almost systematically unsuccessful. On the whole, the computational results, although limited to a class of nonconvex problems, show the

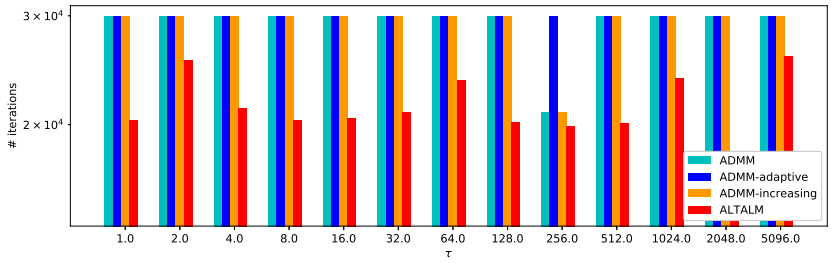


Figure 3.1: Graphical comparison for all of the algorithms in terms of number of iterations for the sigmoidal kernel.

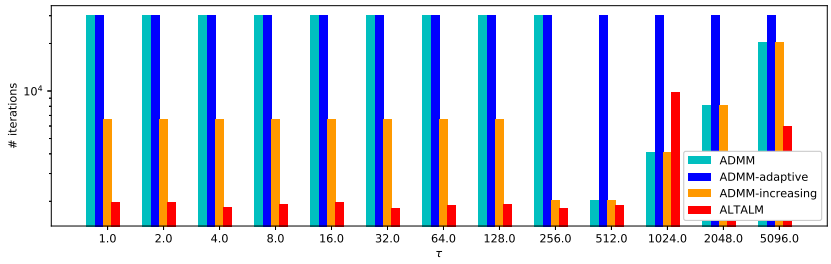


Figure 3.2: Graphical comparison for all of the algorithms in terms of number of iterations for the Gaussian mixture kernel.

potential, practical advantages of the proposed algorithm, which combines the alternate minimization of the two blocks of variables with the updating of the penalty parameter and of the multipliers.

Finally, we observe that similar computational comparisons between classical ADMM and ALM based on block Gauss-Seidel decomposition strategies were performed in [32]. The obtained results showed that ADMM is more computationally efficient than ALM using block coordinate minimization methods. However, the penalty parameter, differently from our algorithm, remains constant throughout all the algorithms tested in [32]. This explains the difference of the obtained results, and seems to show the practical usefulness of varying the penalty parameter according to the strategy of our decomposition approach embedded in ALM. As observed in [32], varying the penalty parameter from one iteration to another is more problematic for the

ADMM than ALM, even from a theoretical point of view.

3.6 Conclusion

In this chapter we proposed an algorithm which combines the classical augmented Lagrangian method with the alternate minimization strategy of a modified decomposition Gauss-Seidel algorithm, in order to exploit the possible structure of a nonconvex problem, whose feasible set is defined by two groups of constraints that are easy to treat when considered separately.

The contribution to the literature is twofold. First, an alternate minimization phase to solve the sub-problem is introduced. Such step allows us to deal with the nonconvex case as it does not require to find the global minima of nonconvex sub-problems.

Second, the global convergence to critical points of the whole algorithm is proved under mild conditions and without assuming the convexity of the objective function.

The theoretical results established in the paper can be considered valuable taking into account that the convergence properties for alternate minimization-based methods in the nonconvex case are still a vivid topic of research and the convergence of the whole algorithm cannot be derived from known results.

The computational experience, although limited to a class of problems, seems to show that the proposed algorithm may be computationally advantageous compared with the classical ADMM.

Chapter 4

Two blocks decomposition for the ALM without derivatives

4.1 Introduction

In this chapter we modify the decomposition approach introduced in Chapter 3 to deal with problems where the derivatives of the objective function F are not available or too expensive to compute. We still assume F to be continuously differentiable but we suppose F to be given as the sum of a "smooth" part and a "black-box" part. We use the term "black-box" for the g component to indicate that the derivatives of g are not available.

More precisely we write F as the sum of $f : R^n \to R$ which is continuously differentiable and for which we are able to compute the gradient and $g : R^n \to R$ which is also continuously differentiable but whose derivatives are not available or too expensive to compute.

The problem we consider is thus

$$\begin{array}{ll} \min\limits_{x} & F(x) = f(x) + g(x) \\ s.t. & x \in X \cap Y, \end{array} \tag{4.1}$$

where, as in Chapter 3, the feasible set is given as the intersection of two convex sets X and Y.

The latter problem can be equivalently stated as

$$\begin{aligned} \min_{x,\ y} \quad & f(x) + g(y) \\ & x \in X, \\ & y \in Y, \\ & x = y, \end{aligned}$$

and hence be solved with an alternating augmented Lagrangian approach (ALTALM). Namely, we solve a sequence of sub problems of the form

$$\min_{x \in X,\ y \in Y} q_{\tau_k, \lambda_k}(x, y) \triangleq f(x) + g(y) + \lambda_k^T (x - y) + \tau_k \|x - y\|^2 \tag{4.2}$$

up to approximate stationarity (as in (3.7)) using the alternate minimization strategy (AM). After such a point $(x_{k+1},\ y_{k+1})$ has been obtained the penalty parameter τ_k,the accuracy parameter ϵ_k and the Lagrange multiplier estimates are updated as usual:

$$\bar{\lambda}_{k+1} = \lambda_k + \tau_k (x_{k+1} - y_{k+1}) \tag{4.3}$$

$$\lambda_{k+1} = \Pi_{[\lambda_{min}, \lambda_{max}]} \left[\bar{\lambda}_k \right] \tag{4.4}$$

$$\tau_{k+1} = \left\{ \begin{array}{ll} \tau_k & \text{if } \|x^{k+1} - x^k\| \leq \theta \|x^k - y^k\| \\ \delta \tau_k & \text{otherwise} \end{array} \right\}, \tag{4.5}$$

where $\theta \in (0, 1)$ and $\delta > 1$.

The AM algorithm allow to both 1) separate the the two sets of constraints as in Chapter 3 and 2) separate the "smooth" and "black-box" part of the objective. The AM algorithm works by performing alternate minimization on $q_{\tau_k, \bar{\lambda}_k}(x, y)$ w.r.t to x (the smooth part) and y the "black-box" part. The minimization w.r.t. x can carried out with standard smooth optimization tools, although from a theoretical point of view it is sufficient to perform a simple projected gradient step. The minimization w.r.t. y is carried out with an (iterative) derivative-free procedure. Such a scheme can be described by the repetition of the following two steps:

$$\begin{aligned} x^{t+1} &= x^t + \alpha_t d^t \\ y^{t+1} &= H(y^t), \end{aligned}$$

where:

- d^t is, for instance, the projected gradient, and α_t is computed by an Armijo-type line search;
- $H : R^n \to R^n$ is a *derivative-free* mapping.

4.2 The algorithm

The ALTALM scheme does not need any modification to what was already described in Chapter 3 but we report it in Algorithm 4 for easy of reference.

Algorithm 4: ALTALM

Input: $\lambda^0,\ (x^0,\ y^0) \in X \times Y$,
$\delta > 1,\ \sigma \in (0,\ 1),\ \tau_0 > 0,\ -\infty < \lambda_m < \lambda_M < +\infty$,
a positive sequence $\{\epsilon_k\}$ s.t. $\lim_{k\to+\infty} \epsilon_k = 0$.

for $k = 0, 1, \ldots$ **do**
  $(x^{k+1},\ y^{k+1})$ = find an ϵ_k stationary point of $q_{\tau_k,\lambda^k}(x, y)$ with `AM`
  $\tilde{\lambda}^{k+1} = \lambda_k + \tau_k(x^{k+1} - y^{k+1})$
  $\lambda^{k+1} = \Pi_{[\lambda_m,\lambda_M]}\left[\tilde{\lambda}^{k+1}\right]$
  if $\left\|x^{k+1} - y^{k+1}\right\| \leq \sigma \left\|x^k - y^k\right\|$ **then**
    $\tau_{k+1} = \tau_k$,
  else
    $\tau_{k+1} = \delta\tau_k$.

The AM procedure, used in Algorithm 4, needs, instead, some modifications to take into account the "black-box" part.

As already noticed, to carry out the smooth part in AM we simply require to take a step along the projected gradient direction while for the "black-box" part we employ the same derivative free technique that, at each iteration k, is able to find a point y^{t+1} such that

$$\left\|y^{t+1} - \Pi_Y\left[y^{t+1} - \nabla q_{\tau,\lambda}(x^{t+1}, y^{t+1}\right]\right\| \leq \mu_t, \tag{4.7}$$

where $\{\mu_t\}$ is a sequence of positive number such that $\mu_t \to 0$. This allows to employ iterative derivative free schemes with increasing level of accuracy and not necessarily going for exact stationarity. Note the difference with the setting studied in Chapter 3 where we could easily take the *argmin* for the y component.

In the following, we focus on the case where Y is defined by simple box constraints, namely $Y = \{l \leq y \leq u\}$ and we exploit as a derivative free mapping the DFBox algorithm proposed in [68].

According to this algorithm at each iteration the $2n$ coordinate direction are explored and if a sufficient decrease of the objective function is found

Algorithm 5: Alternate Minimization (AM) - DF case

Input: $x^0 \in X$, $y^0 \in Y$, $\{\mu_t\}$ s.t. $\mu_t > 0$ and $\mu_t \to 0.$, $\epsilon > 0$

1 $t = 0$

2 **while** (x^t, y^t) *is not an* ϵ*-stationary point* **do**

3 find x^{t+1} such that

$$q_{\tau,\lambda}(x^{t+1}, y^t) \leq q_{\tau,\lambda}(x^t + \alpha_t d^t, y^t),$$

where $d^t = \Pi_X[x^t - \nabla_x q_{\tau,\lambda}(x^t, y^t)] - x^t$ and α_t is obtained with an Armijo-type line search.

4 find y^{t+1} such that $q_{\tau,\lambda}(x^{t+1}, y^{t+1}) \leq q_{\tau,\lambda}(x^{t+1}, y^t)$ and

$$\left\| y^{t+1} - \Pi_Y \left[y^{t+1} - \nabla q_{\tau,\lambda}(x^{t+1}, y^{t+1}) \right] \right\| \leq \mu_t \qquad (4.6)$$

$t = t + 1$.

the point is updated and the search is started again. Otherwise the initial step length is reduced. Notice that by controlling μ_t we roughly control the number of internal iterations (and hence number of coordinate searches) of DFBox for each external iteration k. It is usually convenient to start with few iterations at the beginning and allow the algorithm to run for more iterations later in the process. Note also that, as will be clearer after the convergence analysis, we can compute the update y^{t+1} without actually computing the gradient w.r.t. to y as (4.6) seems to require.

The pseudocode of this procedure is given in Algorithm 6 w.r.t. the

problem $\min_{y \in Y} g(y)$.

Algorithm 6: DFBox

Input: g : function to be minimized, $y_0 \in Y$, $\theta \in (0,1)$, $\gamma > 0$, $d_i^i = e^i$ for $i = 1, \ldots, 1$.

for $k = 0, 1, \ldots$ **do**
  $z_{k+1}^0 = y_k$
  for $i = 1, \ldots, n$ **do**
    $\alpha = \texttt{linesearch}\ (z_{k+1}^{i-1}, d_k^i, \tilde{\alpha}_k^i)$
    if $\alpha > 0$ **then**
      $\alpha_k^i = \alpha,\ \tilde{\alpha}_{k+1}^i = \alpha_k^i,\ d_{k+1}^i = d_k^i$
    else
      $\alpha = \texttt{linesearch}\ (z_{k+1}^{i-1}, -d_k^i, \tilde{\alpha}_k^i)$
      if $\alpha > 0$ **then**
        $\alpha_k^i = \alpha, \tilde{\alpha}_{k+1}^i = \alpha_k^i,\ d_{k+1}^i = -d_k^i$
      else
        $\alpha_k^i = 0,\ \tilde{\alpha}_{k+1}^i = \theta \tilde{\alpha}_k^i$
    $z_{k+1}^i = z_k^{i-1} + \alpha_k^i d_k^i$
  find $y_{k+1} \in Y$ s.t. $g(y_{k+1}) \leq g(z_{k+1}^n)$

def *linesearch*$(y, d, \tilde{\alpha}, \gamma, \delta)$**:**
  compute α_{max} s.t. $y + \alpha_{max} d \in Y$
  $\alpha = \min\{\tilde{\alpha},\ \alpha_{max}\}$
  if $\alpha \leq 0$ *or* $g(y + \alpha d) > g(x) - \gamma \alpha^2$ **then**
    return 0
  else
    $\alpha = \tilde{\alpha}$
    while $\frac{\alpha}{\delta} \leq \alpha_{max}$ *and* $g(y + \frac{\alpha}{\delta} d) \leq g(y) - \gamma(\frac{\alpha}{\delta})^2$ **do**
      $\alpha = \frac{\alpha}{\delta}$
    return α

4.3 Convergence analysis

The convergence analysis will be carried out in a bottom up fashion:

- we first show how we can satisfy the approximate stationarity criterion (4.7) for the black-box part with DFBox,
- then we show that AM converges to such approximate solution in a

finite number of iterations,

- finally we show that the sequence of approximate solutions produced by the whole Algorithm converges to a stationary point of the original problem.

The first part of the analysis is thus devoted to showing that applying DFBox to the minimization w.r.t. y of $q_{\tau_k,\lambda_k}(x_{k+1}, y)$ we can find a point y^{t+1} s.t. criterion (4.7) can be enforced.

4.3.1 Convergence of DFBox

Consider the problem (with an abuse of notation)

$$\begin{aligned} \min \quad & f(x) \\ & l \leq x \leq u, \end{aligned}$$

where f is a continuously differentiable function $f : R^n \to R$. Define the reduced gradient as:

$$\nabla_i^{red} f(x) = \begin{cases} \max\{\nabla_i f(x),\ 0\} & x_i = u_i, \\ \min\{\nabla_i f(x),\ 0\} & x_i = l_i, \\ \nabla_i f(x) & l < x < u. \end{cases} \tag{4.8}$$

From Proposition 3 in [68] we have

Proposition 8. Suppose that f is bounded below on the feasible, that its gradient is Lipschitz continuous (with constant L). Let $\{x_{k+1}\}$ be the sequence produced by Algorithm DFBox. Then,there exist a constants $C > 0$ such that

$$\left\|\nabla_i^{red} f(x^{k+1})\right\| \leq Cn \max_i\{\tilde{\alpha}_k^i\} \tag{4.9}$$

It can be shown that the reduced gradient has close relationship with the projected gradient which we employ in our analysis. In fact, it holds the following.

Proposition 9. Let $f : R^n \to R$ be a continuous differentiable function and $[l, u]$ represent the box constraints then

$$\left\|\nabla^+ f(x)\right\| \leq \left\|\nabla^{red} f(x)\right\| \quad \forall x \in R^n, \tag{4.10}$$

where $\nabla^+ f(x) = x - \Pi_{[l,u]}\left[x - \nabla f(x)\right]$.

Proof. We distinguish between the following cases:

i) $l_i < x < u_i$,

iia) $x_i = l_i$ and $\nabla_i f(x) = \epsilon \geq 0$,

iib) $x_i = l_i$ and $\nabla_i f(x) = -\epsilon < 0$,

iiia) $x_i = u_i$ and $\nabla_i f(x) = -\epsilon \leq 0$,

iiib) $x_i = u_i$ and $\nabla_i f(x) = \epsilon > 0$.

In case i) we have $\nabla_i^{red} f(x) = \nabla f_i(x)$ and

$$\nabla_i^+ f(x) = \begin{cases} \nabla_i f(x) & \text{if } l_i < x_i - \nabla_i f(x) < u_i \\ x - u_i & \text{if } x - u_i > \nabla_i f(x),\ \nabla_i f(x) < 0 \\ x - l_i & \text{if } x - l_i < \nabla_i f(x),\ \nabla_i f(x) > 0. \end{cases}$$

And thus, $\left|\nabla_i^+ f(x)\right| \leq |\nabla_i f(x)| = \left|\nabla_i^{red} f(x)\right|$.

In case iia) we have $\nabla_i^{red} f(x) = 0$ and $x_i - \nabla_i f(x) < l_i$ which implies $\nabla_i^+ f(x) = x_i - l_i = 0$. Thus $\left|\nabla_i^+ f(x)\right| = \left|\nabla_i^{red} f(x)\right|$.

In case iib) we have $\nabla_i^{red} f(x) = \nabla_i f(x)$. Thus if $x_i + \epsilon < u_i$ we have $\nabla_i^+ f(x) = \nabla_i f(x) = \nabla_i^{red} f(x)$ otherwise $\nabla_i^+ f(x) = x_i - u_i > -\epsilon$ and thus $\left|\nabla_i^+ f(x)\right| \leq \left|\nabla_i^{red} f(x)\right|$.

Cases iiia) and iiib) are analogous to iia) and iib).

Therefore we have $\left|\nabla_i^+ f(x)\right| \leq \left|\nabla_i^{red} f(x)\right| \forall i$ and thus the thesis follows. □

Corollary 1. Suppose that f is bounded below on the feasible, that its gradient is Lipschitz continuous (with constant L). Let $\{x_{k+1}\}$ be the sequence produced by Algorithm DFBox. Then,there exist a constants $C > 0$ such that

$$\left\|\nabla^+ f(x)\right\| \leq \left\|\nabla^{red} f(x)\right\| \leq Cn \max_i \{\tilde{\alpha}_k^i\}.$$

4.3.2 Convergence of AM

We now prove the convergence of the AM decomposition algorithm. By simple modifications to Proposition 5, under Assumption 4, we immediately get the following.

Proposition 10. The function $q_{\tau,\lambda}(x, y)$ is coercive on $X \times Y$.

We then can state the convergence of AM.

Proposition 11. Algorithm AM determines in a finite number of iterations a point $(x(\epsilon), y(\epsilon))$ which is an ϵ-stationary point for problem

$$\min_{x \in X, y \in Y} q_{\tau,\lambda}(x, y).$$

Proof. Suppose Algorithm AM generates an infinite sequence $\{x^t, y^t\}$. The instructions of the algorithm imply

$$q_{\tau,\lambda}(x^t, y^t) \leq q_{\tau,\lambda}(x^0, y^0),$$

and hence, for all $t \geq 0$ the point (x^t, y^t) belongs to the level set $\mathcal{L}_0 = \{(x, y) \in X \times Y \mid q_{\tau,\lambda}(x, y) \leq q(x^0, y^0)\}$. From Proposition 10 it follows that $\mathcal{L}_0$ is a compact set. Therefore, the sequence $\{x^t, y^t\}$ admits cluster points. Let $(\bar{x}, \bar{y})$ be a cluster point of $\{x^t, y^t\}$, i.e., there exists $T \subseteq \{0, 1, \dots, \}$ such that

$$\lim_{t \in T, t \to +\infty} (x^t, y^t) = (\bar{x}, \bar{y}).$$

We first show that $(\bar{x}, \bar{y})$ is a critical point, i.e.,

$$(\bar{x}, \bar{y}) = \Pi_{X \times Y}[(\bar{x}, \bar{y}) - \nabla q_{\tau,\lambda}(\bar{x}, \bar{y})]. \tag{4.11}$$

From the instructions of the algorithm we get

$$\left\| y^t = \Pi_Y[y^t - \nabla_y q_{\tau,\lambda}(x^t, y^t)] \right\| \leq \mu_t,$$

from which, taking the limits for $t \in T$ and $t \to +\infty$, since $\mu_t \to 0$, recalling the continuity of the gradient and of the projection operator, we obtain

$$\bar{y} = \Pi_Y[\bar{y} - \nabla_y q_{\tau,\lambda}(\bar{x}, \bar{y})]. \tag{4.12}$$

Let $\hat{x}^t = \Pi_X[x^t - \nabla_x q_{\tau,\lambda}(x^t, y^t)]$, so that $d^t = \hat{x}^t - x^t$. The instructions of the algorithm imply

$$q_{\tau,\lambda}(x^{t+1}, y^{t+1}) \leq q_{\tau,\lambda}(x^{t+1}, y^t) \leq q_{\tau,\lambda}(x^t + \alpha_t d^t, y^t) < q_{\tau,\lambda}(x^t, y^t). \tag{4.13}$$

Recalling the continuity of the gradient and the projection operator, it follows $d^t \to \bar{d}$ for $t \in T$, $t \to +\infty$, which implies that $\|d^t\| \leq M$ for some $M > 0$ and for $t \in T$. The sequence $\{q_{\tau,\lambda}(x^t, y^t)\}$ is monotone decreasing, $q_{\tau,\lambda}(x, y)$ is continuous, and hence we have

$$\lim_{t \to +\infty} q_{\tau,\lambda}(x^t, y^t) = q_{\tau,\lambda}(\bar{x}, \bar{y}).$$

From (4.13) it follows

$$\lim_{t \in T, t \to +\infty} q_{\tau,\lambda}(x^t, y^t) - q_{\tau,\lambda}(x^t + \alpha_t d^t, y^t) = 0.$$

Then, the hypothesis of Proposition 12 are satisfied and hence we obtain

$$\lim_{t \in T, t \to +\infty} \nabla_x q(x^t, y^t)^T d^t = 0. \tag{4.14}$$

By the properties of the projection operator we have

$$(x^t - \nabla_x q_{\tau,\lambda}(x^t, y^t) - \hat{x}^t)^T (x - \hat{x}^t) \leq 0 \quad \forall x \in X,$$

and hence we can write

$$\nabla_x q_{\tau,\lambda}(x^t, y^t)^T d^t \leq - \left\| x^t - \hat{x}^t \right\|^2. \tag{4.15}$$

Thus, from (4.14) and (4.15), recalling the continuity of the gradient and of the projection operator we obtain

$$\bar{x} = \Pi_X[\bar{x} - \nabla_x q_{\tau,\lambda}(\bar{x}, \bar{y})]. \tag{4.16}$$

From (4.16) and (4.12) it follows that (4.11) holds.

Then, by the continuity of $\nabla q_{\tau,\lambda}$ and $\Pi_{X \times Y}$, it follows that for t sufficiently large we have

$$\|(x^t, y^t) - \Pi_{X \times Y}[(x^t, y^t) - \nabla q_{\tau,\lambda}(x^t, y^t)]\| \leq \epsilon,$$

i.e., (x^t, y^t) is an ϵ-stationary point.

□

Remark 2. In practice we can check ϵ-stationarity with $\epsilon = Cn\bar{\epsilon}$ by separately checking if

$$\max_i \{\tilde{\alpha}_k^i\} \leq \epsilon$$

after the y^{t+1} update and if

$$\left\| x^{t+1} - \Pi_X \left[x^{t+1} - \nabla_x q_{\tau_t, \lambda_t}(x^{t+1}, y^{t+1}) \right] \right\| \leq \epsilon.$$

4.3.3 Convergence of ALTALM

We have seen in the preceding sections that algorithm AM (where DFBox is used for the black-box part) produces iterates (x^k, y^k) that are ϵ_k-stationary points for the augmented Lagrangian sub-problem (4.2). Hence we have

$$\left\|(x^k, y^k) - \Pi_{X\times Y}\left[(x^k, y^k) - \nabla q_{\tau_k,\lambda_k}(x^k, y^k)\right]\right\| \leq \epsilon_k,$$

as required by Algorithm ALTALM.

The following can be deduced from Theorem 3.

Theorem 4. Let $\{(x^k, y^k)\}$ be the sequence generated by Algorithm ALTALM. Suppose that the sequence $\{\epsilon_k\tau_k\}$ is bounded. Then, every cluster point $(\bar{x}, \bar{y})$ is such that $\bar{x} = \bar{y}$, and $\bar{x}$ is a stationary point of problem (4.1), i.e.,

$$\nabla F(\bar{x})^T(z - \bar{x}) \geq 0 \quad \forall z \in X \cap Y. \tag{4.17}$$

4.3.4 Conclusion

In this chapter we extened the ALTALM approach introduced in Chapter 3 to deal with problems where first order information for the objective function is not completely available. The overall ALTALM scheme is not modified altough modifications have to be made in the AM procedure. In particular we employ a derivative-free iterative procedure for the resulting "black-box" minimization part. With this modification we are able to prove convergence to stationary point as done in the "smooth" setting.

Chapter 5

Two-Level decomposition for training SVMs

5.1 Introduction

This chapter is concerned with the optimization problem that arise in the training phase of Support Vector Machines [19, 29]. In particular we refer to its dual formulation, which is often preferred for non-linear kernels.
Given a data set of n training vectors $v_i \in \mathbb{R}^m$ and their associated labels $y_i \in \{-1, 1\}$, the following linearly and bound constrained quadratic optimization problem is defined:

$$\begin{aligned} \min_{\alpha} \ & f(\alpha) = \frac{1}{2}\alpha^T Q \alpha - \alpha^T e \\ & y^T \alpha = 0 \\ & 0 \leq \alpha \leq C, \end{aligned} \tag{5.1}$$

where e is the n-dimensional vector of all ones, $\alpha \in \mathbb{R}^n$ and $Q \in \mathbb{R}^{n \times n}$ is positive semi-definite. Matrix Q is given by $Q_{ij} = y_i y_j K_{ij}$, where K is the kernel matrix with $K_{ij} = \phi(v_i)^T \phi(v_j)$, for some mapping ϕ to a higher (possibly infinite) dimensional space.

Problem (5.1) has been extensively studied (see the work of [20] or [78] for an extensive survey on optimization methods for SVM training) by both the machine learning and optimization communities over the years especially as data sets are becoming ever larger. Indeed, when the number of data point n is huge (as in many big data applications) the Hessian matrix Q, which

is dense, cannot be fully stored in memory so that standard methods for quadratic programming cannot be used.

Decomposition methods are particularly well suited to deal with this issue, since the original problem is divided into a sequence of smaller sub-problems obtained by fixing subsets of variables. The key aspects of a decomposition algorithm are hence 1) how we decompose the problem in smaller sub-problems and 2) how we solve the sub-problems themselves.

We analyze both aspects. Firstly we study how to effectively solve the sub-problems. In particular we focus on solving sub-problems of more than two variables, which originate from the novel working set selection rule that is the second contribution of this work.

This chapter is organized as follows. We review the literature on decomposition methods for SVM in Section 5.2. Then we summarize the proposed algorithm in Section 5.3 before giving the details on solution of the sub-problem in Section 5.3.1 and on the novel working set selection rule in Section 5.3.2. Extensive numerical experiments are presented in Section 5.4, where, after providing setup details (Section 5.4.1) we analyze first the efficiency of the solver for the sub-problem (Section 5.4.2), then the performance of different working set selection rules (Sections 5.4.3, 5.4.4 and 5.4.5), before comparing the whole algorithm against the state-of-art solver LIBSVM [24] in Section 5.4.6.

5.2 Decomposition methods for SVM training

The literature on decomposition methods is wide and their convergence properties very well understood. General decomposition algorithms, however, are either applied to unconstrained optimization problems or when the feasible set has a simple structure, for instance the Cartesian product of subsets in smaller spaces. In SVM training problems, however, the feasible set cannot be simply partitioned into blocks and hence, custom decomposition methods are needed.

In a decomposition method for SVM training problems, at each iteration k two sets of indexes $W \subset \{1, \dots, n\}$ (referred to as working set (WS)) and $\overline{W} = \{1, \dots, n\} \setminus W$ are identified, with $q = |W| \ll n$, and a sub-problem of

the following form is solved.

$$\begin{aligned} \min_{\alpha_W} \; & f(\alpha_W, \alpha_{\overline{W}}^k) = \frac{1}{2}\alpha_W^T Q_{WW}\alpha_W + p_W^T \alpha_W \\ & y_W^T \alpha_W = -y_{\overline{W}}^T \alpha_{\overline{W}}^k, \\ & 0 \le \alpha_W \le C, \end{aligned} \tag{5.2}$$

where $p_W = -e_W + Q_{W\overline{W}}\alpha_{\overline{W}}$. The feasible set $\mathcal{F}(\alpha_{\overline{W}})$ for the sub-problem is given by $\{\alpha_W \in \mathbb{R}^q \mid 0 \le \alpha_W \le C,\ y_W^T \alpha_W = -y_{\overline{W}}^T \alpha_{\overline{W}}\}$. To construct the sub-problem, exploiting the symmetry of Q, columns Q_W of the Hessian matrix corresponding to the indexes in W are needed:

$$Q_W = \begin{bmatrix} Q_{WW} \\ Q_{\overline{W}W} \end{bmatrix} \tag{5.3}$$

The general framework of a decomposition scheme is described in Algorithm 7.

Algorithm 7: General Decomposition Framework for SVM Training Problems

$\alpha^0 = 0$, $\nabla f(\alpha^0) = -e$, $k = 0$
while *the stopping criterion is not satisfied* **do**
  select the working set W^k
  retrieve the columns Q_W identified by (5.3)
  set $W = W^k$ and compute a solution $\alpha_W^\star$ of sub-problem (5.2)
  set $\alpha_i^{k+1} = \begin{cases} \alpha_i^\star & \text{for } i \in W \\ a_i^k & \text{otherwise} \end{cases}$
  set

$$\nabla f(\alpha^{k+1}) = \nabla f(\alpha^k) + Q_W(\alpha^{k+1} - \alpha^k) = \nabla f(\alpha^k) + \sum_{i \in W} Q_i(\alpha_i^{k+1} - \alpha_i^k)$$

  where Q_i is the i-th column of Q
  set $k = k + 1$
return $\alpha^\star = \alpha^k$

In terms of computational cost, the most expensive step at each iteration of a decomposition method is, especially for large data sets, the computation of the columns of the Hessian matrix corresponding to the indices in the working set W. In particular, if each K_{ij} costs $\mathcal{O}(m)$ (as with typically employed kernels), the cost for one column is $\mathcal{O}(nm)$. These columns are

needed in (5.3) for setting up sub-problem (5.2) and then at step 6 of Algorithm 7 for updating the gradient. The other operations that may require a computational effort are, as we will clarify in the following, the working set selection procedure itself, the solution of sub-problems and the update of gradient.

The cardinality q of the working set, i.e. the dimension of the sub-problem, has to be strictly greater than 1, otherwise we would have $\alpha^{k+1} = \alpha^k$. Based on q, two cases can be distinguished:

- Sequential Minimal Optimization (SMO) algorithms, where $q = 2$;
- General Decomposition Algorithms, where $q > 2$.

The main difference between SMO and General Decomposition Algorithms lies in how the sub-problems are solved. When $q = 2$, variables can be updated by an analytical formula, as shown by [79], whereas for larger working sets the adoption of an iterative method is necessary, e.g., the software MINOS [77], the LOQO primal-dual interior point method [45, 81, 90] or gradient projection [30, 95]. In General Decomposition Algorithms, the working set rarely contains more than a few tens of variables, otherwise a single iteration might become too expensive. In fact, the inefficiency and complexity of QP solvers is the main reason why researchers stopped using working sets of $q > 2$ variables and the interest in SMO algorithms arose.

Concerning the working set selection rule (WSSR), let us first consider SMO algorithms, whose related literature is older and wider. At a feasible point α, the index sets

$$\begin{aligned} R(\alpha) &= \{h \in \{1, \ldots, n\} \mid 0 < \alpha_h < C \ \vee \ (\alpha_h = 0 \wedge y_h = 1) \ \vee \ (\alpha_h = C \wedge y_h = -1)\} \\ S(\alpha) &= \{h \in \{1, \ldots, n\} \mid 0 < \alpha_h < C \ \vee \ (\alpha_h = 0 \wedge y_h = -1) \ \vee \ (\alpha_h = C \wedge y_h = 1)\} \end{aligned} \tag{5.4}$$

characterize the status of the variables. In particular, it can be shown (see, e.g., [59]) that α is a solution of (5.1) if and only if

$$\max_{i \in R(\alpha)} \{-y_i \nabla f(\alpha)_i\} \leq \min_{j \in S(\alpha)} \{-y_j \nabla f(\alpha)_j\}. \tag{5.5}$$

Given a non-optimal, feasible point α, a pair $\{i, j\} \in R(\alpha) \times S(\alpha)$ such that

$$-y_i \nabla f(\alpha)_i > -y_j \nabla f(\alpha)_j$$

is referred to as a violating pair. If

$$i^\star \in \underset{i \in R(\alpha)}{\operatorname{argmax}} \{-y_i \nabla f(\alpha)_i\} \qquad j^\star \in \underset{j \in S(\alpha)}{\operatorname{argmin}} \{-y_j \nabla f(\alpha)_j\}, \tag{5.6}$$

then $\{i^\star, j^\star\}$ is called a most violating pair (MVP).

A classical way of choosing the working set for SMO is choosing a most violating pair [45, 48]. This selection is cheap to compute, as it is done in $\mathcal{O}(n)$ time, and it can be proved that SMO algorithm with this working set selection strategy is globally convergent [56, 57]. In the following we will refer to rule as WSS1. WSS1 is based on first order information: indeed, $i^\star$ and $j^\star$ identify the direction with only two non-null (unit) components that minimizes the first order approximation

$$f(\alpha^k + d) \simeq f(\alpha^k) + \nabla f(\alpha^k)^T d.$$

Other working set selection rules with global convergence properties are described by [27], [23], [60], [66] and [33]. Amongst those, the most widely employed one in practice (e.g., in LIBSVM) is the one from [33], which exploits second order information. We will refer to this this rule as WSS2. Since f is quadratic, the exact reduction of the objective value is given by:

$$f(\alpha^k) - f(\alpha^k + d) = -\nabla f(\alpha^k)^T d - \frac{1}{2} d^T \nabla^2 f(\alpha^k) d. \tag{5.7}$$

Unfortunately, searching the pair of indices that identify the feasible direction with two non-zero components maximizing the objective decrease requires $\mathcal{O}(n^2)$ operations and is thus impractical. The idea is therefore that of choosing one variable as in WSS1 and, having that fixed, identifying the other index so that (5.7) is maximized. This is done, assuming the kernel is positive-definite, by setting

$$\begin{aligned} i &\in \underset{t \in R(\alpha^k)}{\operatorname{argmax}} \{-y_t \nabla f(\alpha^k)_t\} \\ j &\in \underset{h \in S(\alpha^k)}{\operatorname{argmin}} \left\{ -\frac{\delta_{ih}^2}{\rho_{ih}} \mid -y_h \nabla f(\alpha^k)_h < -y_i \nabla f(\alpha^k)_i \right\}, \end{aligned} \tag{5.8}$$

where

$$\rho_{ih} = K_{ii} + K_{hh} - 2K_{ih} \qquad \delta_{ih} = -y_i \nabla f(\alpha^k)_i + y_h \nabla f(\alpha^k)_h, \tag{5.9}$$

being K the kernel matrix. This procedure has cost $\mathcal{O}(n)$, even though it is in fact slightly more expensive than WSS1.

The notion of MVP gives also rise to the simple and widely employed stopping criterion

$$m(\alpha^k) \leq M(\alpha^k) + \epsilon, \tag{5.10}$$

where $\epsilon > 0$ and

$$m(\alpha) = \max_{h \in R(\alpha)} -y_h \nabla f(\alpha)_h \qquad M(\alpha) = \min_{h \in S(\alpha)} -y_h \nabla f(\alpha)_h. \tag{5.11}$$

Note that $m(\alpha) \leq M(\alpha)$ is exactly the optimality condition (5.5). It has been proved [59] that all rules selecting *constant-factor* violating pairs [27] generate sequences $\{\alpha^k\}$ such that $m(\alpha^k) - M(\alpha^k) \to 0$, i.e. algorithms of this type satisfy stopping criterion (5.10) in a finite number of iterations for any $\epsilon > 0$. A violating pair $\{i, j\}$ is referred to as a constant-factor violating pair if

$$y_i \nabla f(\alpha^k)_i - y_j \nabla f(\alpha^k)_j \leq \sigma \left(y_{i^\star} \nabla f(\alpha^k)_{i^\star} - y_{j^\star} \nabla f(\alpha^k)_{j^\star} \right), \tag{5.12}$$

being $0 < \sigma \leq 1$ and $\{i^\star, j^\star\}$ the MVP. WSS1 and WSS2 are indeed instances of the constant-factor violating pair rule and thus lead to finite termination.

In Algorithm 8 we summarize the SMO decomposition method with WSS1, also referred to as SMO-MVP algorithm.

Algorithm 8: SMO-MVP Algorithm

1 $\alpha^0 = 0$, $\nabla f(\alpha^0) = -e$, $k = 0$
2 **while** $m(\alpha^k) - M(\alpha^k) \leq \epsilon$ **do**
3 select the working set $W^k = \{i^k, j^k\}$ according to

$$i^k \in \underset{i \in R(\alpha^k)}{\operatorname{argmax}} -y_i \nabla f(\alpha^k)_i$$

$$j^k \in \underset{j \in S(\alpha^k)}{\operatorname{argmin}} -y_j \nabla f(\alpha^k)_j$$

4 set $W = W^k, \bar{W} = \{1, \ldots, n\} \setminus W$ and analytically compute a solution $\alpha^\star_W$ of sub-problem (5.2)
5 set $\alpha_h^{k+1} = \begin{cases} \alpha_h^\star & \text{for } h \in W \\ a_h^k & \text{otherwise} \end{cases}$
6 set $\nabla f(\alpha^{k+1}) = \nabla f(\alpha^k) + Q_{i^k}(\alpha_{i^k}^{k+1} - \alpha_{i^k}^k) + Q_{j^k}(\alpha_{j^k}^{k+1} - \alpha_{j^k}^k)$
7 set $k = k + 1$
8 **return** α^k

As for General Decomposition Methods, a first working set selection rule has been proposed by [45] and the asymptotic convergence of the decomposition method based on such rule is proved by [56]. The finite termination is shown by [59]. The scheme used in the SVM$^{\text{light}}$ software, selects a working set W with an even number q of variables by solving the following problem:

$$\begin{aligned} \min_{d} \ & \nabla f(\alpha^k)^T d \\ & y^T d = 0, \quad -1 \leq d_i \leq 1 \ \ \forall\, i = 1, \ldots, n, \\ & d_i \geq 0 \text{ if } \alpha_i^k = 0, \quad d_i \leq 0 \text{ if } \alpha_i^k = C, \\ & |\{d_i \mid d_i \neq 0\}| \leq q. \end{aligned} \tag{5.13}$$

With the above problem, the steepest feasible direction with at most q nonzero components is identified. Optimal W according to (5.13) can be efficiently found, similarly as for WSS1, by selecting the $q/2$ most violating pairs. In fact, WSS1 is the particular case of (5.13) when $q = 2$. This selection strategy guarantees global convergence, at the cost of a limited freedom in choosing variables. We will also refer to this WSSR in the following as extended-WSS1.

The theoretical issue about convergence with more arbitrary selection rules is open. Asymptotic convergence to optimal solutions has not been proven in this scenario. However, [97] proved finite termination of Algorithm 7 with stopping criterion (5.10) under the assumption that at each iteration at least one pair of indexes $\{i, j\} \in R(\alpha^k) \times S(\alpha^k)$ is present in the working set such that

$$-y_i \nabla f(\alpha^k)_i > -y_j \nabla f(\alpha^k)_j + \epsilon.$$

This result improves the work of [89], where relaxed definitions of sets $R(\alpha^k)$ and $S(\alpha^k)$ are considered.

Note that, by the above result, any working set selection rule that inserts the MVP into the WS is guaranteed to generate a finite sequence of iterates. This fact is quite important, allowing to consider sophisticated mixed selection strategies without any risk of non-termination. In particular, selection rules can safely be considered that better exploit the commonly used caching technique, consisting of storing the recently used columns of the Hessian matrix in order to avoid their recalculation. [35], [67] and [60] have studied decomposition methods designed to couple convergence properties and the exploitation of the caching strategy. [84] proposed an experimentally efficient way of choosing the working set based on MVPs and caching, but asymptotic convergence properties have not been proven for this rule.

Finally, a different approach is proposed by [71] where the primal problem is modified so that the dual problem has only box constraints. [71] propose Successive Over-relaxation to solve the problem, while [42] designed a specific decomposition method, making use of a specialized version of TRON [61] to solve the sub-problems.

5.3 The proposed algorithm

In this work we propose a new variant of the non-SMO decomposition method for kernel SVMs training which is based on two main ideas:

- the q-variables sub-problems are solved by means of an inner SMO procedure, so that no line-search procedures are required;
- a novel working set selection rule for $q > 2$ variables that exploits both first and second order information is employed.

An outline of the algorithm is summarized by the pseudocode in Algorithm 9.

Algorithm 9: Two-Level Decomposition Method for SVM Training

1 $\alpha^0 = 0$, $\nabla f(\alpha^0) = -e$, $k = 0$
2 **while** $m(\alpha^k) - M(\alpha^k) > \epsilon$ **do**
3 select the variables which define the working set $W \subset \{1, \ldots, n\}$
4 optionally add some variables to W from the cached columns of the Hessian matrix.
5 compute α_W^{k+1} solving $\min_{\alpha_W \in \mathcal{F}(\alpha_{\overline{W}}^k)} f(\alpha_W, \alpha_{\overline{W}}^k)$
6 $\alpha_{\overline{W}}^{k+1} = \alpha_{\overline{W}}^k$
7 $\nabla f(\alpha^{k+1}) = \nabla f(\alpha^k) + \sum_{h \in W} Q_h(\alpha_h^{k+1} - \alpha_h^k)$
8 set $k = k + 1$
9 **return** α^k

We first detail the solution of the sub-problems (step 5) and than describe the novel proposed working set selection rule (step 3).

5.3.1 Solving the subproblems

The sub-problem with the q variables in the working set W generated at each iteration k by the decomposition procedure is given by (5.2). For the

sake of simplicity, let us change the notation to

$$\begin{aligned} \min_x \ & \frac{1}{2} x^T \tilde{Q} x + p^T x \\ & a^T x = b, \\ & 0 \leq x \leq C. \end{aligned} \tag{5.14}$$

Except for the fact that the constant term of the equality constraint is not zero and $p \neq e$, the sub-problems have exactly the same form as (5.1). Thus, classical SVM decomposition schemes can be applied also to the solution of the sub-problems (5.14). In fact, we will show that SMO technique works extremely well not only on large problems, but also with small problems such as the sub-problems herein considered.

The efficiency of SMO algorithms mainly comes from the fact that all the updates of the variables are performed in closed form, not requiring the computational effort of a line-search algorithm.

These observations led us to adopt SMO-MVP to solve the sub-problems. We can hence see the whole training algorithm as a two-level decomposition scheme, where the inner sub-problems are solved by Algorithm 8. In the following, we will also refer to this two-level decomposition scheme as TLD-ISMO (Two Level Decomposition with Inner Sequential Minimal Optimization).

More in detail, our sub-problems solver is described in Algorithm 10. Let $W = \{h_1, \ldots, h_q\}$ so that x_i corresponds to α_{h_i} and a_i corresponds to y_{h_i}.

Given a feasible point x for the sub-problem, we can define, similarly as (5.4), the sets

$$\begin{aligned} \tilde{R}(x) &= \{\ell \in \{1, \ldots, q\} \mid 0 < x_\ell < C \ \vee \ (x_\ell = 0 \wedge a_\ell = 1) \vee (x_\ell = C \wedge a_\ell = -1)\}, \\ \tilde{S}(x) &= \{\ell \in \{1, \ldots, q\} \mid 0 < x_\ell < C \ \vee \ (x_\ell = 0 \wedge a_\ell = -1) \vee (x_\ell = C \wedge a_\ell = 1)\}. \end{aligned} \tag{5.15}$$

Then, the most violating pair $\{i, j\} \in \tilde{R}(x) \times \tilde{S}(x)$ at x can be defined, similarly as (5.6), as

$$i = \operatorname*{argmax}_{\ell \in \tilde{R}(x)} \{-a_\ell((\tilde{Q}x)_\ell + p_\ell)\} \qquad j = \operatorname*{argmin}_{\ell \in \tilde{S}(x)} \{-a_\ell((\tilde{Q}x)_\ell + p_\ell)\}. \tag{5.16}$$

Finally, we can introduce the quantities

$$\tilde{M}(x) = \max_{\ell \in \tilde{R}(x)} \{-a_\ell((\tilde{Q}x)_\ell + p_\ell)\} \qquad \tilde{m}(x) = \min_{\ell \in \tilde{S}(x)} \{-a_\ell((\tilde{Q}x)_\ell + p_\ell)\}, \tag{5.17}$$

similarly as (5.11), in order to define the stopping criterion

$$\tilde{m}(x) \leq \tilde{M}(x) + \epsilon_{\text{in}}, \tag{5.18}$$

where ϵ_{in} is a tolerance.

At each inner iteration κ we select two variables from W via WSS1. The small dimension of the sub-problems may not require more sophisticated selection rules. We then solve the sub-sub-problem

$$\begin{aligned} \min_{x_i, x_j} \; & \frac{1}{2} \begin{pmatrix} x_i & x_j \end{pmatrix} \begin{pmatrix} \tilde{Q}_{ii} & \tilde{Q}_{ij} \\ \tilde{Q}_{ji} & \tilde{Q}_{jj} \end{pmatrix} \begin{pmatrix} x_i \\ x_j \end{pmatrix} + \sum_{\substack{\ell=1 \\ \ell \neq i,j}}^{q} (\tilde{Q}_{\ell j} x_\ell^\kappa x_j + \tilde{Q}_{\ell i} x_\ell^\kappa x_i) + p_i x_i + p_j x_j \\ & a_i x_i + a_j x_j = a_i x_i^\kappa + a_j x_j^\kappa \\ & 0 \leq x_i, x_j \leq C, \end{aligned} \tag{5.19}$$

which can be done in closed form as described, e.g., by [24]. The subprocedure employed to solve the sub-problems is summarized in Algorithm 10. It is worth remarking that, differently from the case of problem (5.1), matrix $\tilde{Q}$ is fully available to Algorithm 10 that solves (5.14).

Algorithm 10: Inner SMO Algorithm

```
x^0 = α_W^k, κ = 0
while m̃(x^κ) − M̃(x^κ) > ε_in do
    select {i, j} ⊂ W according to (5.16)
    compute x_i^{κ+1}, x_j^{κ+1} by solving analytically problem (5.19)
    set x_ℓ^{κ+1} = x_ℓ^κ for all ℓ ≠ i, j
    update the gradients
    set κ = κ + 1
return x^{κ+1}
```

The inner solver stops when KKT conditions are (approximately) satisfied for the sub-problem. Note that, since we employ Algorithm 8, this is proven to happen in a finite number of iterations for any $\epsilon_{\text{in}} > 0$.

5.3.2 A novel working set selection rule

SMO algorithms are widely considered the state-of-the-art decomposition methods for Support Vector Machines training (see e.g. [78]), and WSS2 introduced by [33] is accepted as more efficient than WSS1 originally introduced by [48] although numerical experience shows that, while WSS2 is

indeed generally better in terms of number of iterations, WSS1 can still be competitive in terms of computational time in some cases (as numerical evidence will show in Section 5.4.4).

The core idea of the proposed working set selection rule is, thus, leveraging on the efficiency of the sub-problem solver, to exploit the benefits of both WSS1 and WSS2 selecting two variables according to WSS1 and other two variables according to WSS2. Formally, we choose the working set $W = \{i_1, i_2, j_1, j_2\}$ as follows:

$$\begin{aligned} i_1 &= \operatorname*{argmax}_{h \in R(\alpha^k)} \left\{ -y_h \nabla_h f(\alpha^k) \right\}, \\ j_1 &= \operatorname*{argmin}_{h \in S(\alpha^k)} \left\{ -y_h \nabla_h f(\alpha^k) \right\}, \\ i_2 &= \operatorname*{argmax}_{h \in R(\alpha^k), h \neq i_1} \left\{ -y_h \nabla_h f(\alpha^k) \right\}, \\ j_2 &= \operatorname*{argmin}_{h \in S(\alpha^k), h \neq j_1} \left\{ -\frac{\delta_{i_2 h}^2}{p_{i_2 h}} \;\middle|\; -y_h \nabla_h f(\alpha^k) < -y_{i_2} \nabla_{i_2} f(\alpha^k) \right\}. \end{aligned} \tag{5.20}$$

We will refer to the working set selection rule (5.20) as WSS-MIX. As a matter of fact, swapping the order of the two pairs, i.e., selecting the first pair by WSS2 and the second one by WSS1, is a possibility. In fact, from preliminary experiments we observed that such change does not significantly affect the performance.

A crucial technique employed in all of state-of-the-art decomposition method for SVM is to cache the most recently computed Hessian matrix columns for later reuse, thus saving up in the number of kernel computations. Moreover it has been shown to be beneficial, especially for large data sets, to make use of the cached columns by augmenting the working set with additional variables of which the correspondent Hessian columns are currently cached [84]. This idea fits nicely in our framework, since we are able to deal with large working sets of variables. We thus enhance WSS-MIX by adding cached variables.

A refined version of the rule from [84] is proposed by [95]. By such a rule, the working set is filled as follows: $\bar{q}$ variables are selected according to extended-WSS1. Then they add to the working set $q - \bar{q}$ variables as described by Algorithm 11. Priority is given to free variables, then lower bound variables and last upper bound variables. The idea is that upper bound variables have probably reached their final value, while free variables are completing the optimization process. Also, indexes that have been in

Algorithm 11: Working Set Filling

Input: W^k, $|W^k| = \bar{q}$, W^{k-1}, $|W^{k-1}| = q$.

put

$$F = \{h \in W^{k-1} \mid 0 < \alpha_h^k < C\},$$
$$L = \{h \in W^{k-1} \mid \alpha_h^k = 0\},$$
$$U = \{h \in W^{k-1} \mid \alpha_h^k = C\}.$$

while $|W^k| \leq q$ *and* $F \setminus W^k \neq \varnothing$ **do**
    add to W^k the index $h \in F \setminus W^k$ that has been in the working set for the least number of iterations.
while $|W^k| \leq q$ *and* $L \setminus W^k \neq \varnothing$ **do**
    add to W^k the index $h \in L \setminus W^k$ that has been in the working set for the least number of iterations.
while $|W^k| \leq q$ *and* $U \setminus W^k \neq \varnothing$ **do**
    add to W^k the index $h \in U \setminus W^k$ that has been in the working set for the least number of iterations.

the working set for less iterations are preferred. For our method, we borrow Algorithm 11 to complete the working set after the first 4 variables are selected according to WSS-MIX.

[95] fixes arbitrarily the size of the working set at the beginning and then adaptively tune the proportion of variables that are selected with extended-WSS1 and with Algorithm 11. Here we follow a slightly different approach and keep both the number of variables chosen with WSS-MIX and of cached variables fixed from the beginning.

Identifying the ideal number of extra variables to add at each iteration for each problem is not an easy task. It seems unlikely that choosing more than 20 variables is useful; indeed, if the data set is very large, the cache will typically not be large enough to store more than $\sim$ 20 - 30 variables, while if the data set is not enough large, the time saved by not recomputing kernels is lost in performing unnecessary computation. Furthermore, the higher cost of solving sub-problems is another reason for not using large values of q. Indeed, in past non-SMO works q have usually been set around 10 or 20 [42, 45].

We found that a rule of thumb for choosing the number of additional variables with a problem from a given data set can be obtained starting

from the following formula. Let

$$S = \frac{\text{cachesize}}{8 \times n^2 \times m} \tag{5.21}$$

be the fraction of Hessian matrix elements that can be cached (assuming elements are double precision values) divided by the number of features (the cost of computing one element of the Hessian is proportional to this quantity). The rule is as follows:

- if $S > 10^{-3}$ do not select additional variables,
- if $10^{-5} < S < 10^{-3}$ select 6 additional variables to the working set by means of Algorithm 11,
- if $S < 10^{-5}$ select 14 additional variables to the working set by means of Algorithm 11.

The idea is that the addition of cached variables is useful if variables often get out of cache before being used again and if kernel computation is expensive; note the cost of computing one Hessian column is $\mathcal{O}(mn)$, where n is the number of examples in the data set and m the number of features.

5.3.3 Convergence properties of the proposed algorithm

To start the discussion we note that the algorithm is well defined i.e. the inner loop (Algorithm 10) stops in a finite number of steps. This trivially comes from the finite termination property of SMO-MVP algorithm [57, 59].

Finite termination of the whole algorithm is, instead, still an open issue. Indeed WSS-MIX (both with or without cached variables) does not satisfy the sufficient conditions for asymptotic global convergence and finite termination stated by [57, 59]. Under the assumption that the sub-problem are solved exactly, however, finite termination of the procedure is guaranteed by the result proven by [97]. Nonetheless, the proposed algorithm solves the sub-problems up to ϵ_{in} precision and, since $M(\alpha)$ and $m(\alpha)$ are not continuous, even letting $\epsilon_{in} \to 0$ does not guarantee that the result from [97] holds. We can however show that

1. the objective function is monotonically non increasing;
2. at each iteration the variables are updated;

3. the working set changes at every iteration, so that it cannot happen that the algorithm infinitely loops on the same sub-problem.

Point 1. is trivial, we prove points 2. and 3.

Let W_k be the working set at the (outer) iteration k. Let ϵ_{in} and ϵ be the stopping tolerances of the inner and outer loops respectively, with $\epsilon_{\text{in}} \leq \epsilon$.

Let us define

$$\bar{R}(\alpha^k_{W_k}) = W_k \cap R(\alpha^k) \qquad \bar{S}(\alpha^k_{W_k}) = W_k \cap S(\alpha^k).$$

From the above definition, we have that if $\{i,j\} \subset W_k$ and $\{i,j\} \subseteq R(\alpha^k) \times S(\alpha^k)$, then $\{i,j\} \subseteq \bar{R}(\alpha^k_{W_k}) \times \bar{S}(\alpha^k_{W_k})$.

From the instructions of the algorithm and the definition of WSS-MIX, we have that at the beginning of iteration k there exists $\{i,j\} \subseteq \bar{R}(\alpha^k_{W_k}) \times \bar{S}(\alpha^k_{W_k})$ such that

$$-y_i \nabla f(\alpha^k)_i + y_j \nabla f(\alpha^k) > \epsilon. \tag{5.22}$$

Moreover, from the instructions of Algorithm 10, at the end of the k-th outer iteration we get α^{k+1} such that, for all $\{i,j\} \subseteq \bar{R}(\alpha^{k+1}_{W_k}) \times \bar{S}(\alpha^{k+1}_{W_k})$, it holds

$$-y_i \nabla f(\alpha^{k+1})_i + y_j \nabla f(\alpha^{k+1}) \leq \epsilon_{\text{in}} \leq \epsilon. \tag{5.23}$$

If it was $\alpha^k = \alpha^{k+1}$, (5.22) and (5.23) would be in contradiction. Therefore have that $\alpha^{k+1} \neq \alpha^k$, i.e., point 2. holds.

From the definition of WSS-MIX, we also know that the working set W_{k+1} contains a pair $\{i_{k+1}, j_{k+1}\} \subseteq R(\alpha^{k+1}) \times S(\alpha^{k+1})$ such that

$$-y_{i_{k+1}} \nabla f(\alpha^{k+1})_{i_{k+1}} + y_{j_{k+1}} \nabla f(\alpha^{k+1})_{j_{k+1}} > \epsilon, \tag{5.24}$$

otherwise the outer stopping criterion would have been satisfied. We will prove that $\{i_{k+1}, j_{k+1}\} \nsubseteq W_k$, so $W_{k+1} \neq W_k$. Assume by contradiction that $\{i_{k+1}, j_{k+1}\} \subset W_k$. Then, since $\{i_{k+1}, j_{k+1}\} \subseteq R(\alpha^{k+1}) \times S(\alpha^{k+1})$, it has to be $\{i_{k+1}, j_{k+1}\} \subseteq \bar{R}(\alpha^{k+1}_{W_k}) \times \bar{S}(\alpha^{k+1}_{W_k})$. Thus, both (5.23) for $\{i,j\} = \{i_{k+1}, j_{k+1}\}$ and (5.24) hold, which is absurd. Point 3. is thus proved.

5.4 Numerical experiments

In this section we provide numerical evidence to back up each component of the proposed algorithm before comparing the efficiency of the method as a

Data set	Training Size	Features	Class
splice	1,000	60	small
a1a	1,605	123	small
leukemia	38	7,129	small
a9a	32,561	123	medium size
w8a	49,749	300	medium-size
ijcnn1	49,990	22	medium-size
rcv1.binary	20,242	47,236	large
real-sim	72,309	20,958	large
covtype.binary	581,012	54	huge

Table 5.1: Details of the 9 initial data sets.

whole. We use several data sets of different dimensions from the LIBSVM collection to obtain different instances of SVM training problems for several values of the hyper-parameters. We detail the common setup of all the experiments in the next Section.

5.4.1 Benchmark problems and experiments setup

We initially considered a collection of 9 data sets of different dimension for binary classification, taken from LIBSVM data sets collection webpage [1]. In Table 5.1 we list them, specifying their characteristics, i.e., number of examples and features.

To fairly measure the performance of the considered methods, we mimic a true application context. To do so, we built up our experimental benchmark as follows: for each data set, we generated a set of 25 RBF kernel SVM training problems, obtained varying the values of the hyper-parameters C and γ on a 5×5 grid. The grid takes values from $\{10^k \cdot \bar{C} \mid k = -2, -1, 0, 1, 2\}$ and $\{10^k \cdot \bar{\gamma} \mid k = -2, -1, 0, 1, 2\}$, where $\bar{C}$ and $\bar{\gamma}$ are suitable values found through a preliminary validation procedure conducted with LIBSVM. We recall that the RBF kernel is defined as

$$K(v, u) = \exp\left(-\gamma \|v - u\|^2\right).$$

Note that the classification of a data set in terms of dimensions (i.e., the data set scale) partially depends on the cache size. The diversity of

[1] https://www.csie.ntu.edu.tw/~cjlin/libsvmtools/datasets/binary.html

datasets we employ allows us to simulate contexts where either the whole kernel matrix can be stored or repeated and expensive kernel computations are needed. We roughly grouped the datasets into four classes: small (`a1a`, `splice`, `leukemia`), medium-size (`a9a`, `w8a`, `ijcnn1`), large (`rcv1`, `real-sim`) and huge (`covtype`).

In the experiment regarding the inner sub-problems solvers, we employed stopping condition (5.18) with tolerance $\epsilon_{\text{in}} = 10^{-5}$.

As for the various versions of Algorithm 7, the implementation exploits the structure of LIBSVM [24]. The considered methods share stopping criterion (5.10), with $\epsilon = 10^{-3}$. Different values of ϵ have also preliminarily been tried, leading to similar outcomes as $\epsilon = 10^{-3}$. The inner stopping tolerance for TLD-ISMO was again set to $\epsilon_{\text{in}} = 10^{-5}$.

5.4.2 Computational evidence on inner SMO efficiency

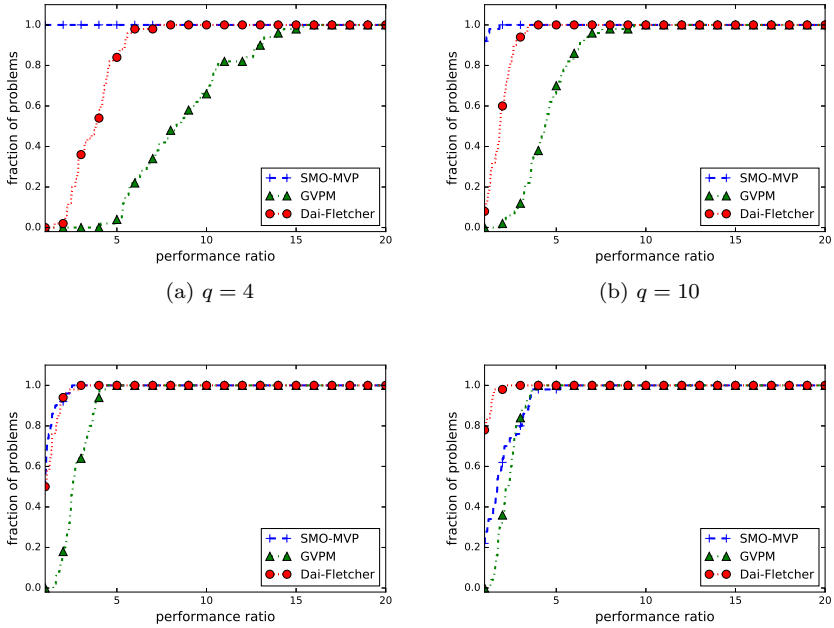
In order to evaluate the efficiency of the inner SMO scheme at solving sub-problems arising in Algorithm 7, we compared it against two state-of-art solvers for problems with bounds and a single linear constraint employed in decomposition schemes for SVM training. Namely we considered the Generalized Variable Projection Method (GVPM) proposed by [95] and the Dai-Fletcher method described in [30].

The inner SMO procedure have been implemented from scratch, while we used the implementations of GVPM and Dai-Fletcher methods available at `http://cdm.unimo.it/home/matematica/zanni.luca/`. All three algorithms were then employed in the decomposition scheme whose implementation is highly based on that of LIBSVM [24].

The comparison is carried out on 50 Gaussian kernel SVM training problems, generated from data sets `a1a` and `splice` as described in Section 5.4.1.

For each test problem, we computed the average time spent by each solver to solve one sub-problem of q variables during a training run conducted with Algorithm 7 with extended-WSS1. Different vales of q have been considered, in particular we repeated the experiment for $q \in \{4, 10, 20, 50\}$. The results are shown in Figure 5.1. Here, as well as in the rest of the chapter, comparisons are carried out by means of performance profiles (described in Appendix B.1).

We can see that, for limited values of q, SMO-MVP is almost always the best algorithm, often by a large factor. Dai-Fletcher method appears to be overall superior to GVPM (in accordance with the observations of [95]). As



(a) $q = 4$ (b) $q = 10$

(c) $q = 20$ (d) $q = 50$

Figure 5.1: Performance profiles of mean sub-problem solution time required by different solvers (SMO-MVP, GVPM, Dai-Fletcher) in the decomposition method for SVM training with the extended-WSS1, run on 50 RBF kernel SVM training problems from `a1a` and `splice`, for different values of q.

the dimension of the working set grows, the gap in performance shrinks: for $q = 20$, the profiles of SMO-MVP and Dai-Fletcher nearly overlap. Finally, when $q = 50$, Dai-Fletcher substantially outperforms the other two methods, which behave similarly.

Now, this trend is in fact not surprising. The good scalability properties of GVPM and Dai-Fletcher algorithms are known from the literature [95].

The results of the preceding test make us confident that for General Decomposition Algorithms with $q > 2$ the Two-Level Decomposition scheme is the best one to adopt, at least for values of q not too large.

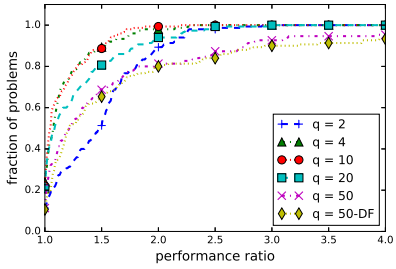
5.4.3 Evaluation of different working set sizes

It is now interesting to evaluate the benefits, if any, of using larger working sets, being an efficient sub-problems solver available. We thus compare the performance, in terms of runtime, number of iterations and kernel computations of Algorithm 7 equipped with the extended-WSS1 the inner SMO solver, for different values of q. For the case $q = 2$, the employed algorithm is nothing but the standard SMO-MVP on the whole problem. The test for $q = 50$ was repeated by using the Dai-Fletcher solver, as it appears to be, from the results in Figure 5.1d, the best solver for such working set size.

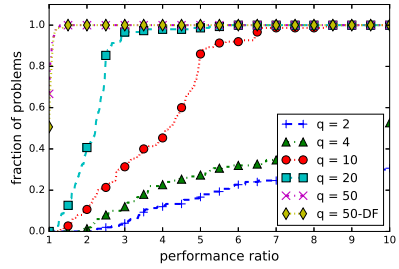
This experiment is performed on 150 SVM training problems generated from data sets `a1a`, `splice`, `a9a`, `ijcnn1`, `w8a` and `rcv1` as described in Section 5.4.1. This is a collection of problems from small, medium-sized and large (although not huge) data sets, constituting therefore a diverse enough test suite.

The results of the experiment are shown in Figure 5.2. Better performance, in terms of runtime, can be observed for the non-extreme working set sizes. As we could reasonably expect, the number of iterations generally decreases as the size of the working set increases. On the other hand, we can see that the number of kernel evaluations has the opposite behavior. We can interpret the result as follows: choosing more variables altogether has the advantage of reducing the number of iterations and consequently the number of times the selection procedure is performed; this is particularly beneficial with the extended-WSS1, since the cost of this selection rule is constant with respect to the WS size; however, the variables are selected in a less careful way: this is paid off by computing possibly unnecessary Hessian columns. Moreover, the computational effort required to solve the sub-problem becomes greater as the size of the working set grows. Therefore, intermediate values of q, such as $q = 4$, are not surprisingly good in terms of runtime. It is interesting to note that the performance of $q = 50$ does not change much when the internal solver is changed; we can explain this behavior highlighting that kernel evaluation and working set selection are much more relevant in determining the total runtime w.r.t. the sub-problem solution. We can conclude that the availability of an efficient solver for sub-problems with more than two variables allows to concretely consider the use of non-SMO decomposition strategy, at least for reasonable values of q. In particular, more sophisticated and combined variables selection rules can be employed, and also the caching mechanism can be exploited in a more systematic way.

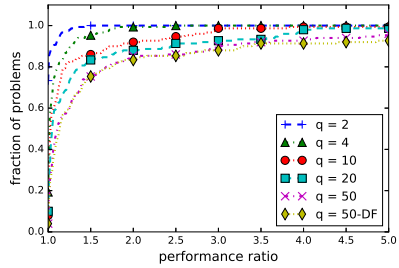
We now turn to evaluating the performance of the novel working set selection rule.



(a) Runtime



(b) Number of (outer) iterations

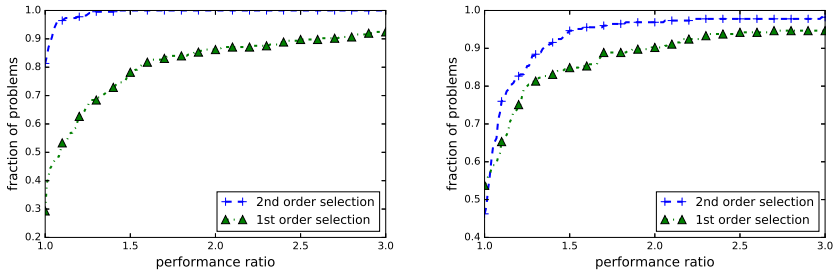


(c) Number of kernel evaluations.

Figure 5.2: Performance profiles of runtimes (a), iterations (b) and number of kernel evaluations (c) of the decomposition method with extended-WSS1, run on 150 RBF kernel SVM training problems with different values of q. The problems were generated from data sets `a1a`, `splice`, `a9a`, `w8a`, `ijcnn1` and `rcv1`. We employed SMO as inner solver. The test for $q = 50$ was repeated using Dai-Fletcher method since from the results in Figure 5.1d it appears to be the optimal choice for such working set size.

5.4.4 Experimental analysis of working set selection rules

We start the discussion on the working set selection rule with a preliminary comparison of WSS1 and WSS2, both in terms of iterations and CPU time. The results are given in Figure 5.3.



(a) Number of iterations. (b) Runtime.

Figure 5.3: Performance profiles comparing the performance, in terms of number of iterations and runtime, of SMO equipped with WSS1 and WSS2, run on 225 RBF kernel SVM training problems. The problems were generated as described in Section 5.4.1 from data sets `a1a`, `splice`, `leukemia`, `a9a`, `w8a`, `ijcnn1`, `rcv1`, `real-sim` and `covtype`.

We observe that, in terms of iterations, WSS2 is indeed clearly superior to WSS1. This confirms the results shown by [33], which make WSS2 more appealing especially with large datasets (less iterations are assumed to require less kernel evaluations). However, if we consider the computing time, the difference is less marked and in a good number of cases WSS1 seems a fairly good solution. In particular, in about a half of the situations, WSS1 led to a faster training.

An explanation to this inconsistency between iterations and time can be found in the cost of the working set selection rule itself. In fact, a deeper analysis of the results revealed that this behavior, although common to all problems, is particularly pronounced on problems generated from the small data sets. For such problems the time spent on kernel computations is not only less dominant (even if still relevant) in the composition of total runtime, but also it is almost equal for the two different algorithms, since the Hessian matrix can be entirely stored and thus approximately only Hessian columns associated with support vectors are computed, once and only once.

Therefore, the computational cost of the variables selection procedure gets much more weight in such cases.

Finally, let us consider the extensions to the general, non-SMO case of WSS1 and WSS2. The cost of WSS1 procedure does not depend on the number of selected variables; on the other hand, the cost of WSS2 linearly increases with the size of the working set. Therefore, we may expect that the behavior observed in Figure 5.3 is even more evident when dealing with General Decomposition Algorithms.

Next, we propose a comparison between all combinations of WSS1 and WSS2 for q equal to 2 or 4. In particular, we consider

- 4-WSS-MIX: Algorithm 7 with $q = 4$ and WSS-MIX;
- 2-WSS2: SMO with WSS2;
- 2-WSS1: SMO-MVP;
- 4-WSS1: Algorithm 7 with $q = 4$ and the extended-WSS1, i.e. the rule from [45];
- 4-WSS2: Algorithm 7 with $q = 4$ and an extension of WSS2; in particular, the generalization of WSS2 that we considered selects two pairs of variables with WSS2, but for the second pair the "roles" of sets $R(\alpha^k)$ and $S(\alpha^k)$ are inverted, so that it's the variable chosen from $S(\alpha^k)$ the one which is fixed; formally:

$$\begin{aligned} i_1 &= \operatorname*{argmax}_{h \in R(\alpha^k)} \left\{ -y_h \nabla_h f(\alpha^k) \right\}, \\ j_1 &= \operatorname*{argmin}_{h \in S(\alpha^k)} \left\{ -\frac{\delta_{i_1 h}^2}{p_{i_1 h}} \;\middle|\; -y_h \nabla_h f(\alpha^k) < -y_{i_1} \nabla_{i_1} f(\alpha^k) \right\}, \\ j_2 &= \operatorname*{argmin}_{h \in S(\alpha^k), h \neq j_1} \left\{ -y_h \nabla_h f(\alpha^k) \right\}, \\ i_2 &= \operatorname*{argmin}_{h \in R(\alpha^k), h \neq i_1} \left\{ -\frac{\delta_{j_2 h}^2}{p_{j_2 h}} \;\middle|\; -y_h \nabla_h f(\alpha^k) > -y_{j_2} \nabla_{j_2} f(\alpha^k) \right\}. \end{aligned} \tag{5.25}$$

We applied the above five algorithms to 175 Gaussian kernel SVM training problems, obtained from data sets `a1a`, `splice`, `a9a`, `w8a`, `ijcnn1`, `rcv1` and `real-sim` as described in Section 5.4.1. Concerning the methods with $q = 4$, we employed the TLD-ISMO scheme.

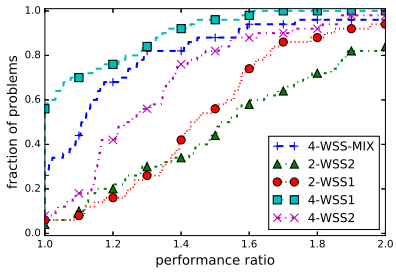
The comparison has been made on the basis of runtime, number of iterations and number of kernel evaluations. These three quantities should provide a sufficiently good insight about the behavior of the algorithms. We report in Figures 5.4 and 5.5 the results of the tests. Figures 5.4a, 5.4c and 5.4e illustrate the results on the problems from small data sets (`a1a`, `splice`); Figures 5.4b, 5.4d and 5.4f concern the medium-sized data sets (`a9a`, `w8a`, `ijcnn1`); Figures 5.5a, 5.5c and 5.5e show the results on the problems from large data sets (`rcv1`, `real-sim`); finally, Figures 5.5b, 5.5d and 5.5f summarize the results upon all 175 problems.

We can observe that with small data sets choosing larger working sets is convenient in terms of runtime, in particular by using WSS1. In fact, doubling the size of the working set on average reduces the number of iterations to slightly less than a half. As previously outlined, the number of kernel evaluations is almost constant w.r.t. the WSSRs. WSS2 and WSS-MIX are slightly more efficient, in terms of number of iterations, w.r.t. WSS1; however, when it comes to runtime, WSS1 seems to be the best choice; we can thus deduce that the cost of the selection procedure itself is particularly relevant with these problems.

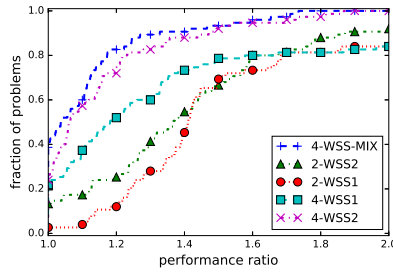
The situation is quite different with large data sets. The presence of at least one pair of variables selected by WSS2 is often useful to reduce the number of iterations. This quantity is indeed strictly related to the number of kernel evaluations in this case, since the cache is relatively small. Being the runtime almost entirely determined by the cost of kernel computations, using second order information is thus beneficial.

The case of medium-sized data sets appears to be the most complicated. The number of iterations, the cost of working set selection rule and the number of kernel evaluations are all relevant to the total runtime. WSS-MIX apparently balances these components of the cost to obtain the best performance overall. It is worth mentioning that for some of these problems, the absence of variables selected by second order information resulted in particularly critical cases in terms of iterations.

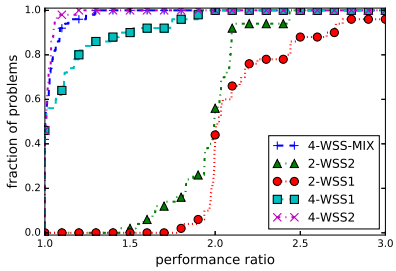
The aggregate results from all 175 problems show a behavior, not surprisingly, somewhere in between the cases of large data sets and small data sets and similar to the case of medium-sized data sets. One variable selected using second order information is necessary in order to avoid disasters in terms of iterations; at the same time, selecting more variables at once, especially if most of them are chosen by the cheap WSS1, can reduce the time spent on



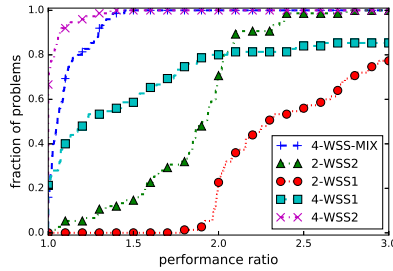
(a) small data sets - runtime



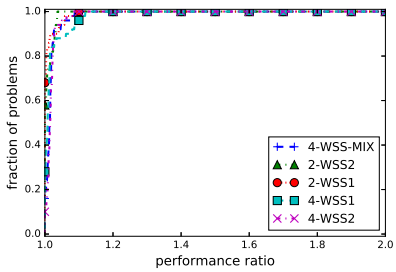
(b) medium-sized data sets - runtime



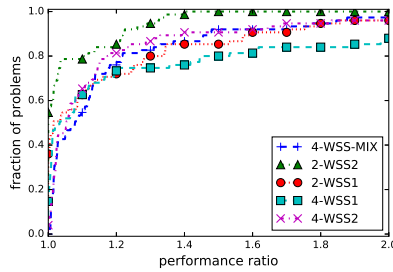
(c) small data sets - number of (outer) iterations



(d) medium data sets - number of (outer) iterations

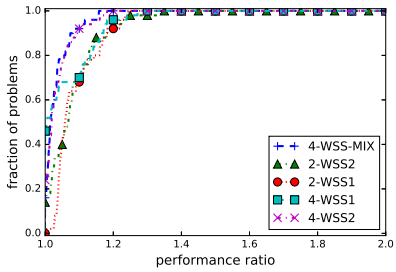


(e) small data sets - number of kernel evaluations

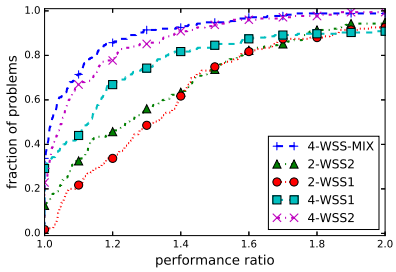


(f) medium data sets - number of kernel evaluations

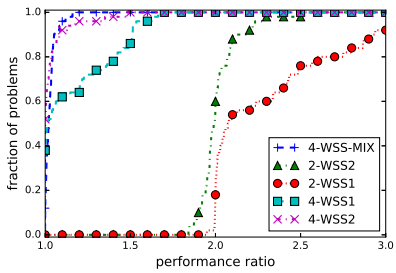
Figure 5.4: Performance profiles comparing the performance, in terms of runtime (first row), number of (outer) iterations (second row) and number of kernel columns evaluations (third row), of Algorithm 7 equipped with combinations of WSS1 and WSS2, run on problems generated from the small data sets `a1a`, `splice` (first column) and from the medium-sized data sets `a9a`, `w8a`, `ijcnn1` (second column).



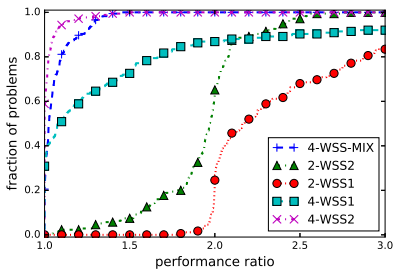
(a) large data sets - runtime



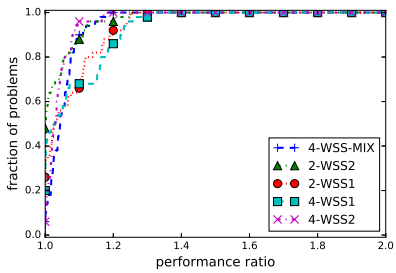
(b) all data sets - runtime



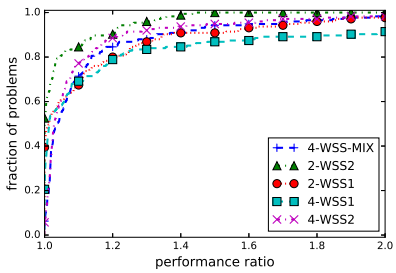
(c) large data sets - number of (outer) iterations



(d) all data sets - number of (outer) iterations



(e) large data sets - number of kernel evaluations



(f) all data sets - number of kernel evaluations

Figure 5.5: Performance profiles comparing the performance, in terms of runtime (first row), number of (outer) iterations (second row) and number of kernel columns evaluations (third row), of Algorithm 7 equipped with combinations of WSS1 and WSS2, run on problems generated from the large data sets `rcv1`, `real-sim` (first column) and from all the data sets `a1a`, `splice`, `a9a`, `w8a`, `ijcnn1`, `rcv1`, `real-sim`.

the selection procedures. WSS-MIX thus appears as the most suited option to be applied on a generic problem.

5.4.5 Analysis of the effects of adding cached variables

We conclude the analysis on the working set selection rule, before evaluating the efficiency of the full method, by analyzing the effect of adding additional cached variables to the working set.

We therefore analyze how the selection of additional "cached" variables actually affects the total number of kernel evaluations required by Algorithm 7 on both cases of small and large problems.

With small data sets, the Hessian matrix can entirely be stored; therefore, Hessian columns corresponding to support vectors need to be computed only once, while, with reasonable WSS rules, few other columns are computed throughout the optimization process. This consideration is supported by empirical evidence, as shown in Figure 5.6a, concerning the number of kernels evaluated by Algorithm 7 with different WSS rules.

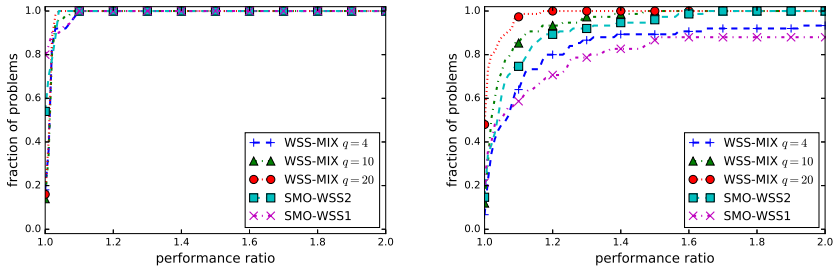
On the contrary, with large problems, the selection of the extra variables corresponding to cached Hessian columns helps much at speeding up the entire process. In fact, as we see in Figure 5.6b, base WSS-MIX, being an hybrid rule combining WSS1 and WSS2, has an intermediate behavior w.r.t. these two basic rules. When the working set is enlarged with the addition of the stored variables, however, the performance greatly improves.

5.4.6 Overall evaluation of TLD-ISMO with WSS-MIX

In this section, the overall efficiency performance of Algorithm 9 is finally evaluated. We compare the TLD-ISMO to the most widely used decomposition schemes for SVM, i.e. SMO methods equipped with WSS1 and WSS2.

Concerning the working set selection rule of TLD-ISMO, rule (5.20) was integrated with the selection of a number of variables from the preceding working set according to Algorithm 11. The final dimension of the working set for each problem was decided by means of the heuristic rule (5.21).

Since we solve a convex problem and for all algorithms we used the same stopping condition, we obtain from the different algorithms equivalent solutions, except for numerical imprecision issues. This is mirrored also in the test accuracy of the models obtained by the different algorithms. Therefore, it makes sense comparing the algorithms in terms of efficiency, i.e. runtime.



(a) 50 small problems

(b) 75 large problems

Figure 5.6: Performance profiles in number of Hessian columns computed by Algorithm 7 with different WSS rules. WSS-MIX with $q = 10$ and $q = 20$ include the addition of 6 and 16 "cached" variables respectively. Figure 5.6a concerns 50 test RBF kernel SVM problems generated from small data sets `a1a` and `splice` as described in Section 5.4.1. Figure 5.6b describes the results of the experiments on 75 problems from large data sets `rcv1`, `real-sim` and the very large `covtype`.

We first ran the three algorithms on the 225 RBF kernel SVM training problems generated from data sets `a1a`, `splice`, `leukemia`, `a9a`, `w8a`, `ijcnn1`, `rcv1`, `real-sim` and `covtype` as described in Section 5.4.1.

A summary of the obtained results can be found in Figures 5.7 and 5.8. In Figure 5.7 the performance profile of the efficiency of the three algorithms on all 225 problems is shown. We can see that not only TLD-ISMO in general performs better than SMO algorithms, but also it often provides a significant speed-up.

In Figure 5.8 we show the same comparison, but we focus the analysis on the different data sets scales. We can observe that the dominance of TLD-ISMO spreads to any situation.

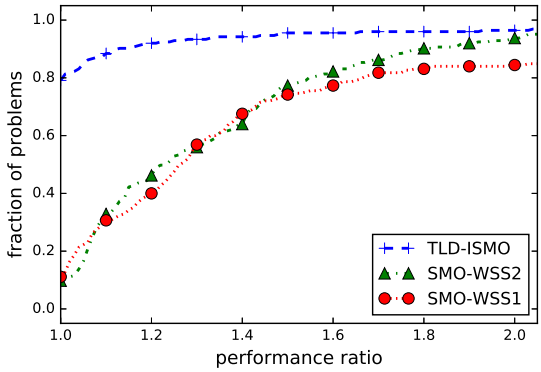
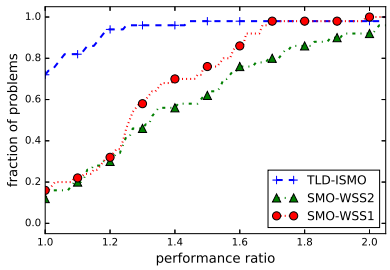
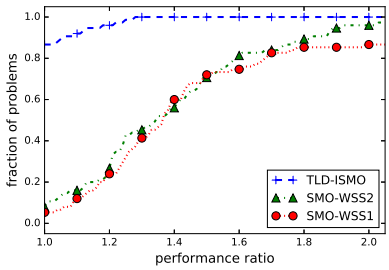


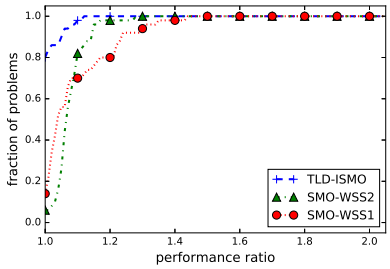
Figure 5.7: Comparison of the efficiency of TLD-ISMO and SMO algorithms (equipped with WSS1 and WSS2) on all 225 test problems.



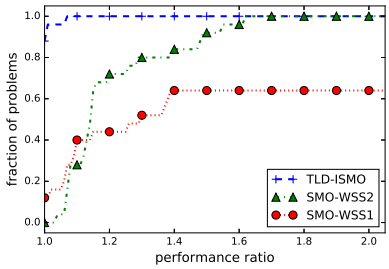
(a) 75 problems from the three small data sets.



(b) 75 problems from the three medium-sized data sets.



(c) 50 problems from the two large data sets.



(d) 25 problems from the huge data set.

Figure 5.8: Performance profiles comparing the efficiency, in terms of runtime, of TLD-ISMO and SMO (equipped with WSS1 and WSS2), on different classes of RBF kernel SVM problems. The problems are obtained, by varying the hyperparameters, from data sets `a1a`, `splice`, `leukemia` (small), `a9a`, `w8a`, `ijcnn1` (medium-sized), `rcv1`, `real-sim` (large) and `covtype` (huge).

5.5 Conclusions

Decomposition schemes are a popular and effective way to solve the optimization problem underlying the training stage of kernel SVMs. A decomposition algorithm reduces the training problem to the solution of several smaller sub-problems of 2 or more variables. However, solving sub-problems of more than two variables is often significantly harder and, thus, not computationally convenient.

In this chapter we proposed a novel way to deal with sub-problems of more than two variables. Namely, since the sub-problems share the same structure and formulation of the original SVM training problem, we solved the sub-problems with the same SMO technique that is used in state-of-art solvers like LIBSVM to solve the original problem.

Numerical evidence is provided to show that employing SMO to solve the sub-problems compares favorably, for up to 50 variables, with other state-of-the-art solvers designed to solve sub-problems of more than two variables, like GVPM and the Dai-Fletcher method.

We then equipped the decomposition algorithm with a novel working set selection rule which exploits both first and second order information, with the addition of "cached variables" for large data sets. The rule yields sub-problems of a minimum of 4 variables for small data sets to a few tens for larger ones.

Finally we compared the whole decomposition scheme, where the sub-problems generated by the novel working set selection rule are solved with SMO, against state-of-art solvers like LIBSVM. The numerical evidence shows that the proposed approach significantly outperforms LIBSVM on a large and diverse range of data sets.

Future work will be devoted to investigate the use of sub-problems of more than 50 variables, where the benefits of the current approach start to wear off. A potentially effective way to deal with larger sub-problems could be that of further decomposing them into sub-sub-problem and use, again, SMO to solve them in a recursive way.

Chapter 6

Bilevel decomposition of nonsmooth problems with convex constraints

6.1 Introduction

In this chapter we consider the optimization of a nonsmooth function $f : \mathbb{R}^n \to \mathbb{R}$ over a closed convex set, namely

$$\begin{array}{ll} \min & f(x) \\ s.t. & x \in X. \end{array} \tag{6.1}$$

We assume that f is locally Lipschitz continuous and that first order information is unavailable or impractical to obtain.

The aim of the optimization, for non smooth problems, is to find a Clarke stationary point. For unconstrained problems Clarke stationarity is defined as in the following.

Definition 4 (Clarke stationarity)**.** A point $\bar{x} \in X$ is Clarke stationary w.r.t. the problem $\min_{x \in \mathbb{R}^n} f(x)$ if

$$f^\circ(\bar{x}; d) \geq 0 \quad \forall d \in \mathbb{R}^n,$$

where

$$f^\circ(x; d) = \limsup_{y \to x,\ t \to 0} \frac{f(y + td) - f(y)}{t} \geq 0$$

is the Clarke generalized directional derivative

For constrained problems of the form (6.1) a possibility is modifying the latter using tangent cones.

Definition 5 (Hyper-Tangent Cone)**.** A vector $d \in \mathbb{R}^n$ is said to be a *hyper-tangent* vector to the set X at $\bar{x} \in X$ if there exists $\epsilon > 0$ such that

$$y + tw \in X \quad \forall y \in X \cap B(x, \epsilon), \quad w \in B(d, \epsilon), \quad t \in (0, \epsilon).$$

The set of all hyper-tangent vector is called the hyper-tangent cone to X at $\bar{x}$ and is denoted $T_X^H(\bar{x})$.

Then from Theorem of [9]) we can state the following.

Definition 6 (Clarke stationarity)**.** A point $\bar{x} \in X$ is Clarke stationary w.r.t. problem (6.1) if

$$f^\circ(\bar{x}; d) \geq 0 \quad \forall d \in T_X^H(\bar{x}),$$

where

$$f^\circ(x; d) = \limsup_{\substack{y \to x,\ y \in X \\ t \to 0,\ y + td \in X}} \frac{f(y + td) - f(y)}{t} \geq 0$$

is the Clarke generalized directional derivative and $T_X^H(\bar{x})$ the hyper-tangent cone at $\bar{x}$.

Literature on derivative-free methods for smooth constrained optimization (i.e. when f is differentiable even tough derivatives are not available) is wide. Several approaches, based on the *pattern search* methods dating back to [41], have been developed for bound and linearly constrained problems in [51] and [52] and for more general type of constraints in [53]. Other works stem from the seminal works of [36, 64] whose approach has been extended for box and linearly constrained problems in [69] and [65] while more general constraints are covered in [63].

The nonsmooth case has seen less development. One of the two major approaches that have emerged is represented by *Mesh adaptive direct search* (MADS) that dates back to [6, 7] and that has been later modified in [1, 8]. This method combines a dense search with an extreme barrier to deal with the constraints. The second main approach [34] is instead based on an exact penalty function.

In the latter the feasible set is expressed by a possibly nonsmooth set of inequalities $g : \mathbb{R}^n \to \mathbb{R}^m$ and the original problem is replaced with the penalized version

$$\min_x f(x) + \frac{1}{\epsilon} \sum_{i=1}^{m} \max\{0, g_i(x)\}. \tag{6.2}$$

It can be shown (see Proposition 3.6 in [34]) that, under suitable assumptions, a value ϵ^* exists such that $\forall \epsilon \in (0, \epsilon^*]$ every Clarke-stationary point $\bar{x}$ of (6.2) is also a stationary point of the original problem. Thus, any algorithm for nonsmooth uncontrained optimization can be applied. In [34], however, a linesearch based algorithm which employs a dense set of directions alongside the $2n$ coordinate directions (CS-DFN) is proposed. This latter reformulation combined with CS-DFN is shown to compare favorably against state-of-art MADS based software like NOMAD.

The value of ϵ^* is, however in general unknown and choosing a proper value of ϵ can be a difficult task. Setting a wrong value of ϵ can be extremely harmful to the performance of the algorithm. For example if f is unbounded outside the feasible set setting too high a value of ϵ can drive the algorithm towards minus infinity. On the other hand too small a value ($< \epsilon^*$), even if theoretical convergence is assured, can yield extremely poor performances because the algorithms may be forced to take really small steps near the boundary of the feasible set.

In this work we propose a novel way of treating "difficult" constraints that is not based on penalty functions. We assume that the feasible set X is a closed convex set and that a projection operator onto the feasible set is available. We do not require X to have an analytical expression nor make any other regularity assumptions. We do, however, suppose that the computational effort needed to compute the projection is negligible compared to the evaluation of the objective function. Indeed, the only computational cost we consider is the number of evaluation of the objective function.

This chapter is organized as follows. In Section 6.2 we introduce the proposed reformulation. In Section 6.3 we prove the equivalence between the novel formulation and the original problem in terms of local and global minima. Then in Section 6.4 we present some numerical results. In particular we propose a comparison between the exact penalty approach of [34] and the proposed reformulation. We make use of the CS-DFN algorithm, used in [34], for both formulations as to make a fair comparison. Finally we give some concluding remarks in Section 6.5.

6.2 A novel formulation

Generalizing a bit, all the approaches that have been proposed in literature to deal with general constraints, try to steer the search towards the feasible set by adding (maybe in a sequential manner) to the objective function some kind of penalty ϕ, which, in its most general form, can be described by

$$\phi(x) = \begin{cases} 0 & \text{if } x \in X \\ > 0 & \textit{otherwise.} \end{cases} \tag{6.3}$$

Such function can be a smooth quadratic or an exact nonsmooth penalty, or also an hard barrier that take $+\infty$ outside the feasible set. The problem is thus rewritten as

$$\min_x P(x;\epsilon) = f(x) + \epsilon\phi(x),$$

where $\epsilon > 0$ is a parameter that must be set.

Consider a local minimum of the original problem x^*. We have that, for some neighborhood $B(x^*, \delta)$,

$$f(x^*) \leq f(x) \ \forall x \in B(x^*, \delta) \cap X.$$

The idea behind penalty-based approaches is that by making the penalty large enough (either by making ϵ large of by using an hard barrier) we have

$$f(x) + p(x) \geq f(x^*) + p(x^*) = f(x^*), \quad \forall x \in B(x^*, \delta) \tag{6.4}$$

so that x^* is also a local minima of the penalized problem.

The idea behind the proposed reformulation is, instead, to avoid penalties by "assigning" to a point x outside the feasible set the value of the objective function computed at its projection $\pi(x)$. In this way we do not ever compute f outside the feasible set and we do not need to "correct" f with a penalty for points outside the feasible set.

Let π be the projection operator over X defined as

$$\pi(x) = \operatorname*{argmin}_{z \in X} \|z - x\| .$$

Notice that since X is convex the projection has a unique solution. We can thus define the problem

$$\min_x f(\pi(x))$$

where each point outside the feasible set assumes the values of its projection. In the latter formulation it is guaranteed that no point outside the feasible set can take a value lower than some point in X. We have however that all the points that share the some projection (consider a ray perpendicular to the constraints) share the same function value. To overcome this issue is sufficient to add to the previous formulation a term that penalizes the distance of a point from its projection. We thus propose to replace the original problem with

$$\min_x \tilde{f}(x) = f(\pi(x)) + \|x - \pi(x)\| . \tag{6.5}$$

We note also that since the projection operator is continuous we have that $\tilde{f}$ is continuous. Moreover if f is bounded from below on the feasible set X then $\tilde{f}$ is bounded on $\mathbb{R}^n$ since $\tilde{f}(x) \geq f(\pi(x)) \geq \inf_{x \in X} f(x)$. On the contrary in the penalty approach $f(x) + \frac{1}{\epsilon} \sum_{i=1}^{m} \max\{0, g_i(x)\}$ can be unbounded.

Consider, as a simple example, the problem HS224 from the test suite [82].

$$\begin{array}{ll} \min\limits_x & 2x_1^2 + x_2^2 - 48x_1 - 40x_2 \\ s.t. & x_1 + 3x_2 \geq 0 \\ & 18 - x_1 - 3x_2 \geq 0 \\ & x_1 + x_2 \geq 0 \\ & 8 - x_1 - x_2 \geq 0 \\ & 0 \leq x \leq 6. \end{array}$$

The level curves of the original objective function f and of the modified problem $\tilde{f}$ are shown in Figure 6.1. Notice how the solution to the problem $x^* = (4, 4)$ becomes an unconstrained global minimum in the proposed formulation.

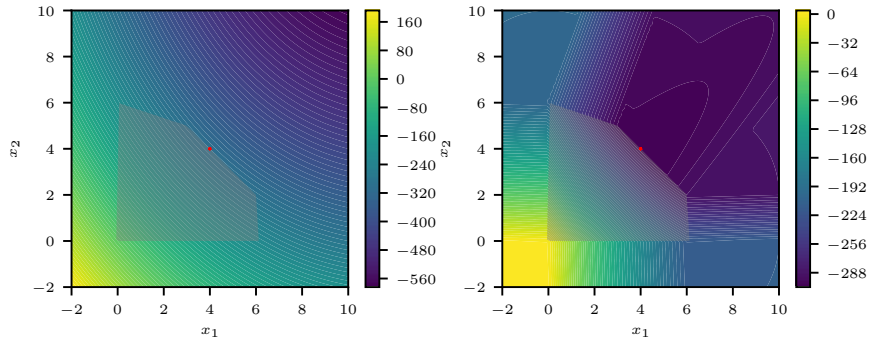


Figure 6.1: Level curves of f (left) and $\tilde{f}$ (right) for problem HS224. The feasible set is in gray. The solution is in red.

6.3 Equivalence of the formulations

In this section we prove the equivalence between the original constrained problem (6.1) and the proposed formulation (6.5) in terms of local and global minima. We also show that modifying the objective function we do not lose Lipschitz continuity so that if f is (locally) Lipschitz $\tilde{f}$ is (locally) Lipschitz too. We start with the latter.

Lemma 3. Let f be locally Lipschitz continuous. Then the modified function $\tilde{f} = f(\pi(x)) + \|x - \pi(x)\|$ is also locally Lipschitz.

Proof. Let $x_0 \in \mathbb{R}^n$. Since f is locally Lipschitz there exists L_0 and δ_0 so that

$$|f(x) - f(x_0)| \leq L_0 \|x - x_0\| \quad \forall x \in B(x_0, \delta_0).$$

Now consider $\tilde{f}$. For every $x \in B(x_0, \delta_0)$ we have

$$\begin{aligned} |\tilde{f}(x) - \tilde{f}(x_0)| &= |f(\pi(x)) + \|x - \pi(x)\| - f(\pi(x_0)) - \|x_0 - \pi(x_0)\|\,| \\ &\leq |f(p(x)) - f(\pi(x_0))| + |\,\|x - \pi(x)\| - \|x_0 - p(x_0)\|\,| \\ &\leq L_0 \|\pi(x) - \pi(x_0)\| + |\,\|x - \pi(x)\| - \|x_0 - \pi(x_0)\|\,| \\ &\leq L_0 \|x - x_0\| + |\,\|x - \pi(x)\| - \|x_0 - \pi(x_0)\|\,|, \end{aligned}$$

where we have used the local Lipschitz continuity of f and the non expansiveness property of the projection operation. Now, by the triangular inequality

we have $\|x_0 - \pi(x_0)\| \leq \|x_0 - x\| + \|x - \pi(x)\| + \|\pi(x) - \pi(x_0)\|$ so that

$$\begin{aligned}\|x_0 - \pi(x_0)\| - \|x - \pi(x)\| &\leq \|x_0 - x\| + \|\pi(x) - \pi(x_0)\| \\ &\leq 2\,\|x_0 - x\|\,.\end{aligned}$$

The same reasoning applies to the opposite sign $-\|x_0 - \pi(x_0)\| + \|x - \pi(x)\|$ so that $|\,\|x - \pi(x)\| - \|x - \pi(x_0)\|\,| \leq 2\,\|x_0 - x\|$ and we can conclude that

$$|\tilde{f}(x) - \tilde{f}(x_0)| \leq (2 + L_0)\,\|x - x_0\| \quad \forall x \in B(x_0, \delta_0).$$

□

We now consider the relationship between the global and local minimum of the two formulations. We first prove that each global (local) minimum of the original problem is also a global (local) minimum of the proposed formulation in Lemma 4 and 5.

Lemma 4. Every global minimum of problem (6.1) is also a global minimum for problem (6.5).

Proof. Let $x^* \in X$ be a global minimum for problem (6.1). Suppose by contradiction that $\exists \bar{x} \in \mathbb{R}$ s.t. $\tilde{f}(\bar{x}) < \tilde{f}(x^*)$ then

$$\tilde{f}(\bar{x}) = f(\pi(\bar{x})) + \|\bar{x} - \pi(\bar{x})\| < \tilde{f}(x^*) = f(\pi(x^*)) + \|x^* - \pi(x^*)\| = f(x^*).$$

Then we have found a point $y = \pi(\bar{x}) \in X$ s.t.

$$f(y) < f(x^*) - \|\bar{x} - y\| < f(x^*),$$

which is a contradiction. □

Lemma 5. Every local minimum of problem (6.1) is also a local minimum for problem (6.5).

Proof. Let $x^* \in X$ be a local minimum for problem (6.1). Then there exists a ball $\mathcal{B}(x^*, \rho)$ with $\rho > 0$ s.t.

$$f(x^*) \leq f(x) \quad \forall x \in \mathcal{B}(x^*, \rho) \cap X.$$

Let $\bar{x} \in \mathcal{B}(x^*, \rho)$ and suppose by contradiction that $\tilde{f}(\bar{x}) < \tilde{f}(x^*)$. Thus we have

$$\tilde{f}(\bar{x}) = f(\pi(\bar{x})) + \|\bar{x} - \pi(\bar{x})\| < \tilde{f}(x^*) = f(\pi(x^*)) + \|x^* - \pi(x^*)\| = f(x^*).$$

Let $y = \pi(\bar{x}) \in X$. We have, by the properties of the projection operator that

$$\|y - \pi(x^*)\| = \|y - x^*\| \leq \|\bar{x} - x^*\| \leq \rho,$$

so that $y \in \mathcal{B}(x^*, \rho) \cap X$. Moreover it holds that

$$f(y) < f(x^*) - \|\bar{x} - y\| < f(x^*),$$

which is a contradiction. □

Moreover since f and $\tilde{f}$ take the same values on X it holds that any global (local) minimum of the modified problem which belongs to the feasible set is also o global (local) minimum of the original problem.

Lemma 6. Every global (local) minimum $x \in X$ for problem (6.5) is also a global (local) minimum of problem (6.1).

We now show, in Lemma 7, that no stationary point does exist outside the feasible region X so that we have a perfect equivalence between global and local minima in the two formulations as remarked in Corollary 2

Lemma 7. Suppose $\hat{x} \in \mathbb{R}^n \setminus X$ then $\hat{x}$ is not a stationary point for the modified problem $\min_x \tilde{f}(x)$.

Proof. We will prove the thesis by showing that there exists a descent direction at $\hat{x}$ and hence $\hat{x}$ cannot be a stationary point.

Consider the projection of $\hat{x}$ onto the feasible set $\bar{x} = \pi(\hat{x})$. Let $d = \bar{x} - \hat{x}$ and consider a point $\hat{x} + \alpha d$ with $\alpha \in (0, 1]$. For every y we have

$$\begin{aligned}(\hat{x} + \alpha d - \bar{x})^T (y - \bar{x}) &= (\hat{x} + \alpha(\bar{x} - \hat{x}) - \bar{x})^T (y - x) \\ &= (1 - \alpha)(\hat{x} - x)^T (y - x) \leq 0,\end{aligned} \tag{6.6}$$

where the latter holds because $\bar{x} = \pi(\hat{x})$. Since the projection has a unique solution, from (6.6) we conclude that $\bar{x} = \pi(\hat{x}) = \pi(\hat{x} + \alpha d)\ \forall \alpha \in [0, 1]$. Thus we have that

$$\begin{aligned}\tilde{f}(\hat{x}) - \tilde{f}(\hat{x} + \alpha d) &= f(\bar{x}) + \|\hat{x} - \bar{x}\| - f(\pi(\hat{x} + \alpha d)) - \|\hat{x} + \alpha d - \pi(\hat{x} + \alpha d)\| \\ &= f(\bar{x}) + \|\hat{x} - \bar{x}\| - f(\bar{x}) - \|\hat{x} + \alpha(\bar{x} - \hat{x}) - \bar{x}\| \\ &= \|\hat{x} - \bar{x}\| - (1 - \alpha)\|\hat{x} - \bar{x}\| \\ &= \alpha \|\hat{x} - \bar{x}\| \\ &> 0,\end{aligned}$$

so that d is a descent direction at $\hat{x}$. □

By putting together Lemma 4, 5, 6 and 7 we establish the perfect equivalence of the two formulations in terms of local and global minima as expressed by the following.

Corollary 2. Every global (local) minimum of problem (6.1) is also a global (local) minimum of problem (6.5) and viceversa.

6.4 Numerical experiments

In the following we propose some numerical experiments to investigate on the possible advantages of the proposed formulation. As a comparison we consider the exact penalty approach of [34]. To make a fair comparison we used the same algorithm to solve both formulations. In particular we used the CS-DFN algorithm proposed in [34]. In the following we call *solver* an algorithm applied to a particular formulation of a given problem. So we compare the *exact penalty* solver, i.e. the CS-DFN algorithm applied to the penalized formulation, and *our* solver, i.e. the CS-DFN applied to the proposed formulation.

Test problems. We set up a benchmark composed of 28 problems belonging to different classes: general nonlinear functions subjected to 1) non degenerate linear constraints from the collection [39,82]; 2) degenerate linear constraints from [2]; 3) general convex constraints again from [39, 82]; and minmax programs under linear constraints from [70]. The problem are listed in Table 6.1.

Performance metric. To compare the results we employ *data profiles*. Data profiles for unconstrained optimization have been proposed in [73] for benchmarking derivative free algorithms. We describe data profiles in Appendix B.

In the constrained case, however, the proposed scheme is not readily applicable. We propose to modify the convergence test (B.1) by considering also the constraint violation as follows:

$$(1-\beta)(f(x)-f_L)+\beta\,\|g_+(x)\| \leq \tau(1-\beta)(f(x_0)-f_L)+\tau\beta\,\|g_+(x_0)\|\,,$$

where $\beta \in (0,1)$ is new parameter which balances function value and constraint violation. We set $f_L = f^*$ since f^* is available for all the problems in the test suite.

problem	n	m	type
HS24	2	5	smooth with linear constraints from [39, 82]
HS36	3	7	
HS37	3	8	
HS44	4	10	
HS86	5	15	
HS224	2	5	
HS231	2	2	
HS232	2	5	
HS250	3	8	
HS331	2	4	
AS6(n=6)	6	12	smooth with linear degenerate constraints from [2]
AS6(n=7)	7	14	
AS6(n=8)	8	16	
AS7(n=6)	6	12	
AS7(n=7)	7	14	
AS7(n=8)	8	16	
HS22	2	2	smooth with convex non linear from [39, 82]
HS29	3	1	
HS43	4	3	
HS65	3	7	
HS66	3	8	
HS270	5	5	
MAD1	2	1	minmax with linear constraints [70]
MAD2	2	1	
MAD4	2	1	
MAD5	2	1	
PENTAGON	6	15	
WONG2	10	3	

Table 6.1: Benchmark problems details.

Note that by violating the constraints the function values can be lower than the f^*. Naturally by choosing a high value for β this situation can be arbitrarily penalized as long as f does not get to $-\infty$. This cases must be removed before computing the profiles.

We extract different data profiles for different values of both τ and β. In particular we extract the curves for τ^{-k} with $k \in \{1, 3, 5, 7\}$ and $\beta \in \{0.9, 0.99\}$.

We set a budget of 1000 function evaluations for all the solvers.

Solvers details As already mentioned for both solver we employ the CS-DFN proposed in [34]. We recall the scheme of the algorithm in Algorithm 12.

Algorithm 12: CS-DF

Input: $x_0 \in \mathbb{R}^n,\ \theta \in (0,1),\ \gamma > 0,\ \eta > 0,$
$\tilde{\alpha}_0^i > 0,\ d_i^i = e^i$ for $i = 1, \ldots, n.$
$\tilde{\alpha}_0$, a sequence $\{d_k\}$ of search directions s.t. $\|d_k\| = 1\ \forall k$

for $k = 0, 1, \ldots$ **do**
  $z_{k+1}^0 = x_k$
  for $i = 1, \ldots, n$ **do**
    $\alpha = \texttt{linesearch}\ (z_{k+1}^{i-1}, d_k^i, \tilde{\alpha}_k^i,\ \gamma,\ \delta)$
    if $\alpha > 0$ **then**
      $\alpha_k^i = \alpha,\ \tilde{\alpha}_{k+1}^i = \alpha_k^i,\ d_{k+1}^i = d_k^i$
    else
      $\alpha = \texttt{linesearch}\ (z_{k+1}^{i-1}, -d_k^i, \tilde{\alpha}_k^i, \gamma, \delta)$
      if $\alpha > 0$ **then**
        $\alpha_k^i = \alpha, \tilde{\alpha}_{k+1}^i = \alpha_k^i,\ d_{k+1}^i = -d_k^i$
      else
        $\alpha_k^i = 0,\ \tilde{\alpha}_{k+1}^i = \theta \tilde{\alpha}_k^i$
    $z_{k+1}^i = z_k^{i-1} + \alpha_k^i d_k^i$
  if $\max_{i=1,\ldots,n}\{\alpha_{k+1}^i,\ \tilde{\alpha}_{k+1}^i\} \leq \eta$ **then**
    $\alpha = \texttt{linesearch}\ (z_{k+1}^n, d_k, \tilde{\alpha}_k, \gamma, \delta)$
    if $\alpha > 0$ **then**
      $\tilde{\alpha}_{k+1} = \alpha_k$
    else
      $\alpha_k = 0,\ \tilde{\alpha}_{k+1} = \theta \tilde{\alpha}_k$
    $z_{k+1}^{n+1} = z_k^n + \alpha_k d_k$
  find x_{k+1} s.t. $f(x_{k+1}) \leq f(z_{k+1}^{n+1})$

def *linesearch*$(x, d, \tilde{\alpha}, \gamma, \delta)$**:**
  if $f(x + \tilde{\alpha} d) > f(x) - \gamma \tilde{\alpha}^2$ **then**
    return 0
  else
    $\alpha = \tilde{\alpha}$
    while $\frac{\alpha}{\delta} \leq \alpha_{max}$ *and* $f(x + \frac{\alpha}{\delta} d) \leq f(x) - \gamma(\frac{\alpha}{\delta})^2$ **do**
      $\alpha = \frac{\alpha}{\delta}$
    return α

We set $\delta = 0.5,\ \gamma = 10^{-6}, \eta = 10^{-6}$ for Algorithm 12. The dense sequence of direction $\{d_k\}$ is obtained by Sobol quasi-random generation [87].

We used the implementation available at [74]

For the *exact penalty* solver we employ the adaptive strategy for tuning ϵ which is proposed in [34] and is deemed to be the better choice.

6.4.1 A first comparison

We start by comparing *ours* solver against the *exact penalty*. The data profiles are reported in Figure 6.2. From the plots we can see that *ours* solver enjoy generally better performance both in term of speed and robustness (or number of problems eventually solved). We note however that neither solver manage to solve more than the 60% of the test problems within the budget of function evaluations when a relatively high precision ($\tau = 10^{-7}$) is required.

6.4.2 A parametrization

In this section we introduce a scale factor ϵ that controls how much to penalize points outside the feasible region. Namely we modify our formulation as

$$\min_x f(\pi(x)) + \epsilon \|x - \pi(x)\| .$$

To understand the effect of ϵ we start with a qualitative analysis. In Figure 6.3 we show the iterates of the algorithm when run on same problem with different values of ϵ.

From the figure we can see that the iterates for greater values of ϵ stay closer to the feasible set while for small values of ϵ the algorithm is allowed to stray far from the feasible set.

To understand whether staying closer to the feasible set has a good or bad effect on the overall optimization process we measure the performance on the solver for different values of ϵ. Namely we try $\epsilon \in \{0.1, 1, 10, 100\}$. We use the same setup of the previous experiments. In Figure 6.4 we give data profiles for the different solvers (we include, for later reference, another configuration ($\epsilon = 10,\ \sigma = 2$) which is explained later in the manuscript).

From Figure 6.4 we can see that choosing an high value of ϵ yields high performances when low precision is required but quickly lessen for higher values of τ. For instance for $\tau = 10^{-7},\ \beta = 0.99$ the algorithm with $\epsilon = 100$ manages to satisfy the convergence criterion only in roughly 30% of the test problems. The opposite if true for small values of ϵ. The algorithm is

generally slower but can achieve very good solutions if a greater number of function evaluation is allowed.

After this analysis it is natural to ask if employing an adaptive strategy for ϵ may be advantageous. For example one could start with ϵ set to a large number and gradually decreasing it to get to accurate solutions.

Coming up with a good schedule for ϵ that works for all problems can be an hard task. However, the CS-DFN algorithm offers a good way to understand in what regime the algorithm is working by looking at the length of the steps that the algorithm takes at each iteration. We thus propose to set the value of ϵ as a function of the set length α_k. More precisely we set

$$\epsilon_{k+1} = \sigma \cdot \alpha_k. \tag{6.7}$$

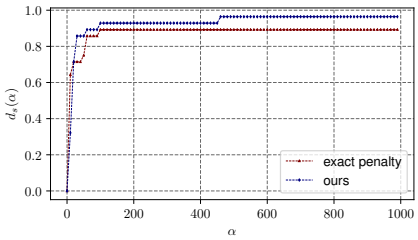
In this way at the beginning of the algorithm we can start with a large value of ϵ_0 and then let it decreasing as the algorithm takes smaller steps.

We found, by manually tuning, that good performances can be obtained by setting $\epsilon_0 = 10, \sigma = 2$ although other configuration perform similarly well. As we can see, again in Figure 6.4, this configuration performs almost equally well when low or high precision is required. Moreover notice that when high precision is required we go from less than 60% of solved problems to 80%.

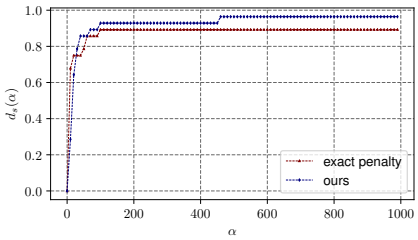
6.4.3 Final comparison

To end the discussion we propose a final comparison of *ours* solver, in its parameterized version, equipped with the adaptive strategy for tuning ϵ against the *exact penalty* approach. The results are reported in Figure 6.5.

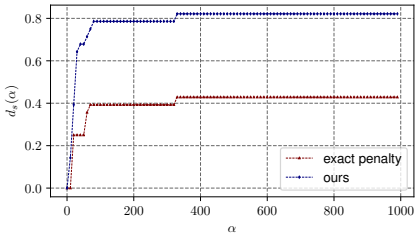
We can see that *ours* solver enjoys better performance for every threshold of accuracy although the *exact penalty* can be faster for some problems when a relatively low precision is required. We note furthermore that the exact penalty fails to reach accurate solution for a large portion of the test problems whether, as already noticed, *ours* solvers manage to reach more than the 80% of solved problems even when high precision is required. We also report for completeness the details of function values and constraint violations after the total budget of function evaluations have been spent in Table 6.2 for both solvers.



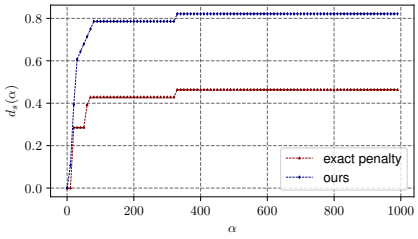
(a) $\tau = 10^{-1},\ \beta = 0.9$



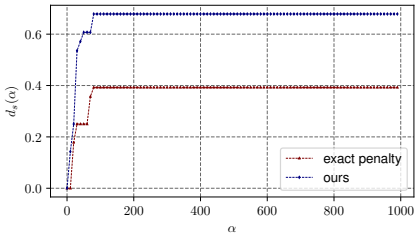
(b) $\tau = 10^{-1},\ \beta = 0.99$



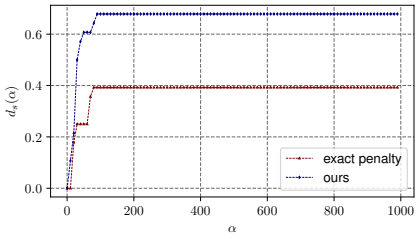
(c) $\tau = 10^{-3},\ \beta = 0.9$



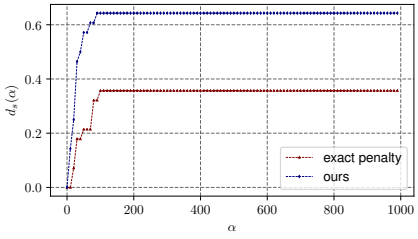
(d) $\tau = 10^{-3},\ \beta = 0.99$



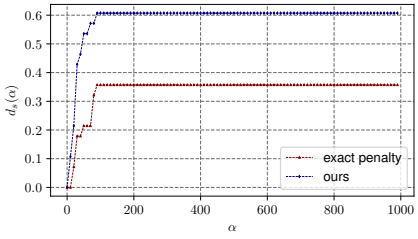
(e) $\tau = 10^{-5},\ \beta = 0.9$



(f) $\tau = 10^{-5},\ \beta = 0.99$



(g) $\tau = 10^{-7},\ \beta = 0.9$



(h) $\tau = 10^{-7},\ \beta = 0.99$

Figure 6.2: Data profiles for *ours* solver and the *exact penalty*.

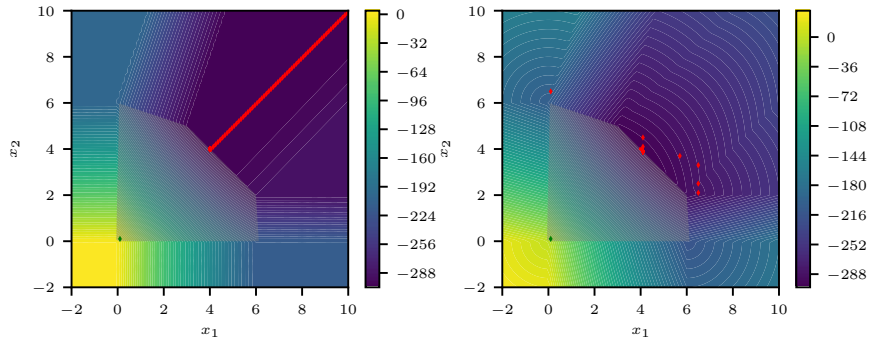
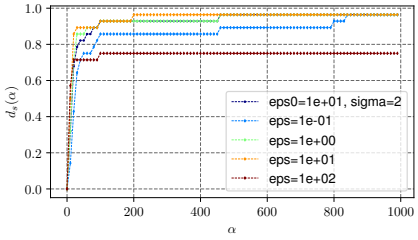
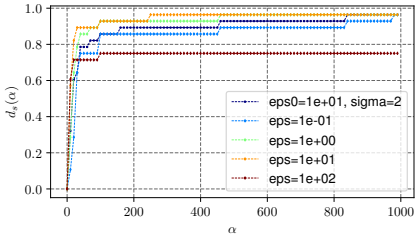


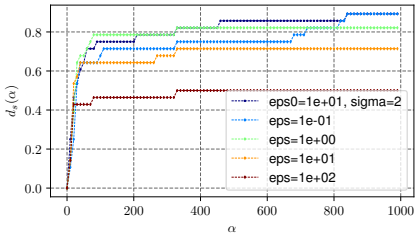
Figure 6.3: Iterates (in red) of CS-DFN for $\epsilon = 0.1$ (left) $\epsilon = 10$ (right) for problem HS224. The feasible set is in grey, the initial point in green. Each dot is obtained after a search along the $2n$ coordinate directions and possibly a direction from the dense sequence.



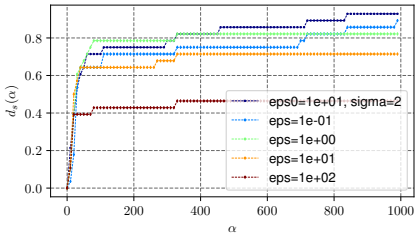
(a) $\tau = 10^{-1},\ \beta = 0.9$



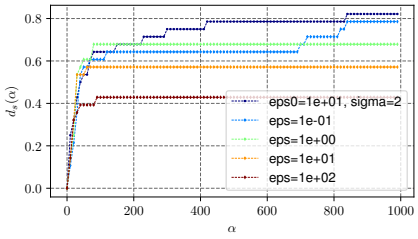
(b) $\tau = 10^{-1},\ \beta = 0.99$



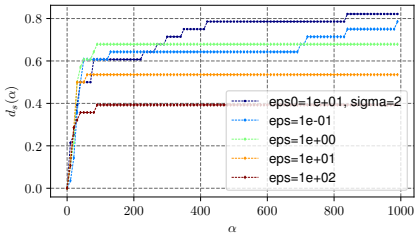
(c) $\tau = 10^{-3},\ \beta = 0.9$



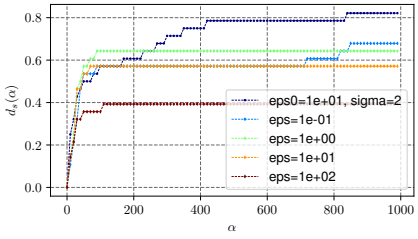
(d) $\tau = 10^{-3},\ \beta = 0.99$



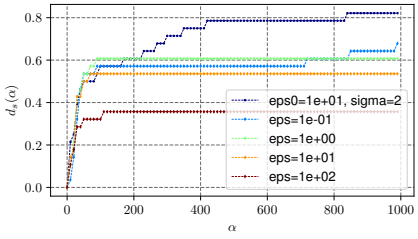
(e) $\tau = 10^{-5},\ \beta = 0.9$



(f) $\tau = 10^{-5},\ \beta = 0.99$

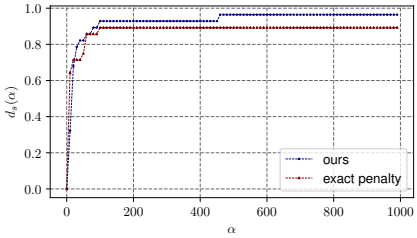


(g) $\tau = 10^{-7},\ \beta = 0.9$

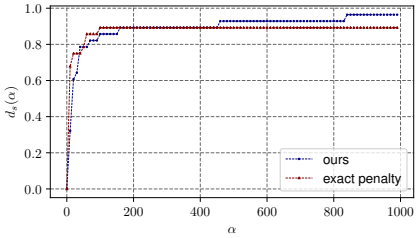


(h) $\tau = 10^{-7},\ \beta = 0.99$

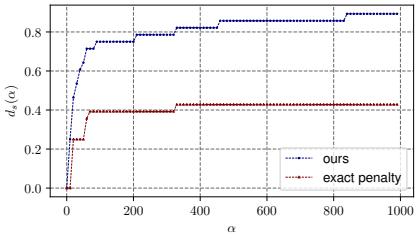
Figure 6.4: Data profiles for different value of the parameter ϵ.



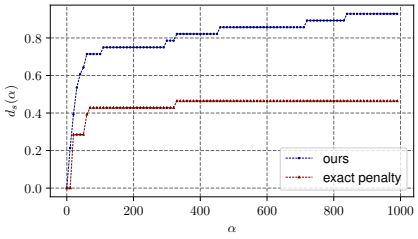
(a) $\tau = 10^{-1},\ \beta = 0.9$



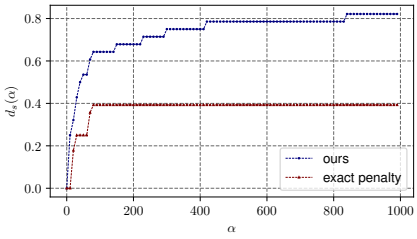
(b) $\tau = 10^{-1},\ \beta = 0.99$



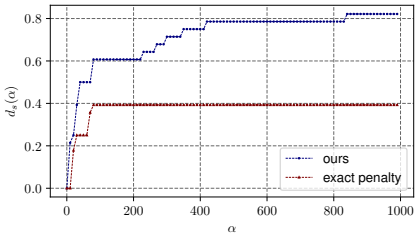
(c) $\tau = 10^{-3},\ \beta = 0.9$



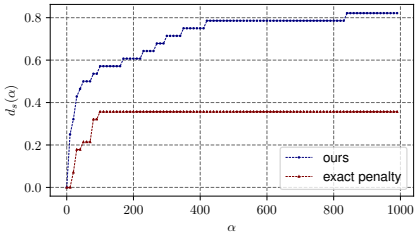
(d) $\tau = 10^{-3},\ \beta = 0.99$



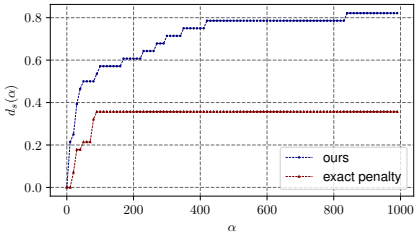
(e) $\tau = 10^{-5},\ \beta = 0.9$



(f) $\tau = 10^{-5},\ \beta = 0.99$



(g) $\tau = 10^{-7},\ \beta = 0.9$



(h) $\tau = 10^{-7},\ \beta = 0.99$

Figure 6.5: Data profiles for the bilevel and the exact penalty formulation.

problem	ours ($\epsilon_0 = 10$, $\sigma = 2$)		exact penalty	
	Δf	$\|g_+\|$	Δf	$\|g_+\|$
HS36	4.03e-06	0.00e+00	1.32e+02	0.00e+00
HS37	1.02e-08	0.00e+00	2.40e+01	0.00e+00
HS44	2.88e-12	0.00e+00	0.00e+00	0.00e+00
HS86	8.27e-08	0.00e+00	3.58e+00	0.00e+00
HS224	7.87e-10	0.00e+00	1.47e+00	0.00e+00
HS231	3.74e-06	0.00e+00	3.74e-06	0.00e+00
HS232	9.23e-13	0.00e+00	4.44e-16	0.00e+00
HS250	4.00e-10	0.00e+00	1.32e+02	0.00e+00
HS331	-1.44e-05	0.00e+00	-1.44e-05	0.00e+00
AS6(n=6)	2.12e-28	0.00e+00	3.02e-28	0.00e+00
AS6(n=7)	7.18e-29	0.00e+00	3.40e-28	0.00e+00
AS6(n=8)	1.51e-28	0.00e+00	2.16e-28	0.00e+00
AS7(n=6)	2.59e-20	0.00e+00	1.30e-28	0.00e+00
AS7(n=7)	5.17e-20	0.00e+00	6.17e-28	0.00e+00
AS7(n=8)	5.43e-20	0.00e+00	1.32e-28	0.00e+00
HS22	4.44e-14	0.00e+00	7.06e-09	0.00e+00
HS29	6.22e-13	0.00e+00	5.84e+00	0.00e+00
HS43	1.08e-12	1.22e-15	2.13e+01	0.00e+00
HS65	-3.53e-07	0.00e+00	2.77e-01	0.00e+00
HS66	1.04e-10	0.00e+00	1.43e-02	0.00e+00
HS270	-3.55e-15	0.00e+00	1.00e+00	0.00e+00
MAD1	3.90e-09	6.66e-16	1.38e-01	0.00e+00
MAD2	-2.99e-09	2.41e-10	6.51e-02	0.00e+00
MAD4	3.89e-09	0.00e+00	7.30e-02	0.00e+00
MAD5	-4.62e-09	0.00e+00	2.27e-02	0.00e+00
PENTAGON	4.53e-02	0.00e+00	1.88e-01	0.00e+00
WONG2	6.27e-01	3.55e-15	4.00e+00	0.00e+00

Table 6.2: Details for *ours* and *exact penalty* solvers after the maximum budget of function evaluations have been spent.

6.5 Conclusion

In this work we proposed to rewrite a nonsmooth optimization problem subjected to convex constraints as an unconstrained parameter-free problem. Such formulation is proven to be equivalent to the original problem, in terms of global and local minima. The formulation can be solved by any optimization algorithm for nonsmooth optimization.

We compared the proposed formulation against to state-of-art approaches for constrained nonsmooth optimization. In particular we compared it against the exact penalty approach. We used the same solver that is shown to deliver state-of-art performances for the penalized problem to solve the proposed formulation. The results clearly shows the advantages of the proposed formulation.

Further work is needed to establish if a prefect equivalence holds also in terms of Clarke stationary points. Indeed, although it is evident that no stationary points can exist outside the feasible set and that in the interior any Clarke stationary point of the reformulation is also a stationary point of the original problem, we still need to understand if there can be stationary points on the boundary of the feasible set for the reformulation that are not stationary points for the original problem.

Appendix A

Linesearch properties

In this Appendix we recall some well known properties for the Armijo-type line search algorithm used in the proof of Proposition 6. All the results and their proofs can be easily found, for instance in [11].

Consider the function $q_{\tau,\lambda}(x,y) : \mathbb{R}^n \times \mathbb{R}^n \to \mathbb{R}$. Let $\{(x^t, y^t)\}$ be a sequence of points belonging to $X \times Y$. Let $d^t \in R^n$ be a direction such that $\nabla_x q_{\tau,\lambda}(x^t, y^t)^T d^t < 0$ and $x^t + d^t \in X$. The line search procedure along the feasible descent direction d^t is shown in Algorithm 13.

Algorithm 13: Line search algorithm (LS)

Data: $\gamma \in (0,1),\ \delta \in (0,1)$.

1 compute

$$\alpha_t = \max_{j=0,1,\ldots} \{\delta^j : q_{\tau,\lambda}(x^t + \delta^j d^t, y^t) \leq q_{\tau,\lambda}(x^t + \delta^j d^t, y^t) + \gamma \delta^j \nabla_x q_{\tau,\lambda}(x^t, y^t)^T d^t\} \tag{A.1}$$

It can be easily seen that the algorithm is well-defined, i.e., there exists a finite integer j such that α_t satisfies the acceptability condition (A.1). Moreover the following result holds.

Proposition 12. Let $T \subseteq \{0, 1, \ldots, \}$ be an infinite subset such that $(x^t, y^t) \to (\bar{x}, \bar{y})$ for $t \in T$ and $t \to +\infty$. Assume that $\|d^t\| \leq M$ for some $M > 0$ and for all $t \in T$. If

$$\lim_{t \in T, t \to +\infty} q_{\tau,\lambda}(x^t, y^t) - q_{\tau,\lambda}(x^t + \alpha_t d^t, y^t) = 0,$$

then we have

$$\lim_{t\in T, t\to+\infty} \nabla_x q_{\tau,\lambda}(x^t, y^t)^T d^t = 0.$$

Appendix B

Performance metrics

B.1 Performance profiles

Performance profiles have been proposed by [31] as a mean to compare the performance of different solvers on a suite of test problems. Performance profiles provide a unified view of relative performance of the solvers that overcome the limitations of previous approaches: they dot not involve interpreting long tables of values, nor do they suffer from the shortcomings of comparing, for instance, the number of wins, the average performance, quantiles, or other cumulative statistics (as argued by [31]). For these reasons they have been adopted by an ever growing number of researches over the last decade.

Formally, consider a test suit of $\mathcal{P}$ problem and a set of solvers $\mathcal{S}$. For each solver $s \in \mathcal{S}$ and problem $p \in \mathcal{P}$ define

$$t_{p,s} = \text{the cost for solver } s \text{ to solve problem } p,$$

where cost is the performance metric we are interested in. For example the cost can be the number of function evaluations, the number of iterations or the CPU time. Define also the ratio

$$r_{p,s} = \frac{t_{p,s}}{\min_{s \in \mathcal{S}}\{t_{p,s}\}},$$

which expresses a relative measure of the performance on problem p of solver s against the performance of the best solver for this problem. If (and only if) a solver fails to solve a problem we put $r_{p,s} = r_M$ with $r_M \geq r_{p,s}\ \forall s, p$.

The performance profile for a solver s is thus the function

$$\rho_s(\tau) = \frac{1}{|\mathcal{P}|} \cdot |\{p \in \mathcal{P} \mid r_{p,s} \leq \tau\}|,$$

which represents the probability for solver s that a performance ratio $r_{p,s}$ is within a factor $\tau \in R$ of the best possible ratio. The function $\rho_s(\tau) : [1, +\infty] \to [0, 1]$ is, in fact, the cumulative distribution function for the performance ratio.

The value of $\rho_s(1)$ is the probability that the solver will win over the rest of the solvers while $\lim_{\tau \to r_M^-} \rho_s(\tau)$ is is the probability that the solver solves a problem.

B.2 Data profiles

Data profiles have been proposed in [73] as a mean to benchmarking derivative-free optimization algorithms. In this context, in fact, there is usually a limited computational budget and hence performance profiles are not well suited tot the task.

Consider a test set of $\mathcal{P}$ problems. We fix a tolerance parameter τ and we say that a problem has been solved by if

$$f(x) - f_L \leq \tau \left(f(x_0) - f_L\right), \tag{B.1}$$

where f_L is an accurate estimate of the minimum of the problem (we can set $f_L = f^*$ if f^* is available). Note that here function values are referred to the original problem even if the solver employs a reformulation.

Then for each solver s we define its data profile as

$$d_s(\alpha) = \frac{1}{|\mathcal{P}|} \left| \left\{ p \in \mathcal{P} \ s.t. \ \frac{nf(\tau)}{n_p + 1} \leq \alpha \right\} \right|, \tag{B.2}$$

where n_p is the number of variables of problem p and $nf(\tau)$ is the number of iterations needed to satisfy the convergence criterion (B.1).

Data profiles are extracted for different values of τ to compare the solvers against different balances of speed versus accuracy.

Appendix C

Publications

This research activity has led to several publications in international journals. These are summarized below.[1]

1. **G. Galvan** and M. Lapucci. "On the convergence of inexact augmented Lagrangian methods for problems with convex constraints", *Operations Research Letters*, vol. 47(3), 2019.
2. **G. Galvan**, M. Lapucci, T. Levato and M. Sciandrone. "An Alternating Augmented Lagrangian method for constrained nonconvex optimization", *Optimization Methods and Software*, 2019.
3. F. Ceccarelli, M. Sciandrone, C. Perricone, **G. Galvan**, E. Cipriano, A. Galligari, T. Levato, T. Colasanti, L. Massaro, F. Natalucci, F.R. Spinelli, C. Alessandri. G. Valesini and F. Conti. "Biomarkers of erosive arthritis in systemic lupus erythematosus: Application of machine learning models", *PLOS ONE*, vol. 13(12), 2018.
4. G. Cocchi, L. Galli, **G. Galvan** and M. Sciandrone. "Machine learning methods for short-term bid forecasting in the renewable energy market: A case study in Italy", *Wind Energy*, vol 21(5), 2018.
5. F. Ceccarelli, M. Sciandrone, C. Perricone, **G. Galvan**, F. Morelli, L.N. Vicente, I. Leccese, L. Massaro, E. Cipriano, F.R. Spinelli, C. Alessandri, G. Valesini and F. Conti. "Prediction of chronic damage in systemic lupus erythematosus by using machine-learning models", *PLOS ONE*, vol 12(3), 2017.
6. Luca Parca, Gerardo Pepe, Marco Pietrosanto, **Giulio Galvan**, Leonardo Galli, Antonio Palmeri, Marco Sciandrone, Fabrizio Ferrè, Gabriele Ausiello, and Manuela Helmer Citterich "Modeling cancer drug response through drug-specific informative genes." Scientific Reports, vol 9, 2019.

Submitted

1. Matteo Lapucci, Marco Sciandrone, **Giulio Galvan**, and Chih-Jen Lin. "A Two-Level Decomposition Framework Exploiting First and Second Order Information for SVM Training Problems", *Journal of Machine Learning Research*, 2019. (Submitted)

[1]The author's bibliometric indices are the following: H-index = 1, total number of citations = 3 (source: Scopus on Month September, 2019).

Bibliography

[1] M. Abramson, C. Audet, J. Dennis, and S. Digabel, "Orthomads: A deterministic mads instance with orthogonal directions," *SIAM Journal on Optimization*, vol. 20, no. 2, pp. 948–966, 2009.

[2] M. A. Abramson, O. A. Brezhneva, J. E. D. Jr, and R. L. Pingel, "Pattern search in the presence of degenerate linear constraints," *Optimization Methods and Software*, vol. 23, no. 3, pp. 297–319, 2008.

[3] R. Andreani, E. Birgin, J. Martínez, and M. Schuverdt, "Augmented lagrangian methods under the constant positive linear dependence constraint qualification," *Mathematical Programming*, vol. 111, no. 1-2, pp. 5–32, 2008.

[4] R. Andreani, E. G. Birgin, J. M. Martínez, and M. L. Schuverdt, "On augmented lagrangian methods with general lower-level constraints," *SIAM Journal on Optimization*, vol. 18, no. 4, pp. 1286–1309, 2007.

[5] R. Andreani, G. Haeser, and J. M. Martínez, "On sequential optimality conditions for smooth constrained optimization," *Optimization*, vol. 60, no. 5, pp. 627–641, 2011.

[6] C. Audet, A. Custódio, and J. Dennis, "Erratum: Mesh adaptive direct search algorithms for constrained optimization," *SIAM Journal on Optimization*, vol. 18, no. 4, pp. 1501–1503, 2008.

[7] C. Audet and J. Dennis, "Mesh adaptive direct search algorithms for constrained optimization," *SIAM Journal on Optimization*, vol. 17, no. 1, pp. 188–217, 2006.

[8] ——, "A progressive barrier for derivative-free nonlinear programming," *SIAM Journal on Optimization*, vol. 20, no. 1, pp. 445–472, 2009.

[9] C. Audet and W. Hare, *Derivative-Free and Blackbox Optimization*, 01 2017.

[10] D. P. Bertsekas, "Multiplier methods: A survey," *Automatica*, vol. 12, no. 2, pp. 133–145, 1976.

[11] ——, *Nonlinear Programming*. Athena Scientific, 1995.

[12] ——, *Constrained Optimization and Lagrange Multiplier Methods (Optimization and Neural Computation Series)*. Athena Scientific, 1996.

[13] D. P. Bertsekas and J. N. Tsitsiklis, *Parallel and Distributed Computation: Numerical Methods*. Upper Saddle River, NJ, USA: Prentice-Hall, Inc., 1989.

[14] E. Birgin and J. Martínez, *Practical Augmented Lagrangian Methods for Constrained Optimization*. Philadelphia, PA: Society for Industrial and Applied Mathematics, 2014.

[15] ——, *Practical Augmented Lagrangian Methods for Constrained Optimization*. Philadelphia, PA: Society for Industrial and Applied Mathematics, 2014.

[16] E. Birgin, C. Floudas, and J. Martínez, "Global minimization using an augmented lagrangian method with variable lower-level constraints," *Mathematical Programming*, vol. 125, no. 1, pp. 139–162, 2010.

[17] E. Birgin, J. Martínez, and L. Prudente, "Optimality properties of an augmented lagrangian method on infeasible problems," *Computational Optimization and Applications*, vol. 60, no. 3, pp. 609–631, 2015.

[18] S. Bochkanov, "ALGLIB." [Online]. Available: http:/alglib.net/

[19] B. E. Boser, I. M. Guyon, and V. N. Vapnik, "A training algorithm for optimal margin classifiers," in *Proceedings of the Fifth Annual Workshop on Computational Learning Theory*. ACM, 1992, pp. 144–152.

[20] L. Bottou, O. Chapelle, D. DeCoste, and J. Weston, *Support Vector Machine Solvers*. MIT Press, 2007.

[21] S. Boyd, N. Parikh, E. Chu, B. Peleato, and J. Eckstein, "Distributed optimization and statistical learning via the alternating direction method of multipliers," *Foundations and Trends in Machine Learning*, vol. 3, no. 1, pp. 1–122, 2010.

[22] R. Chan, M. Tao, and X. Yuan, "Constrained total variation deblurring models and fast algorithms based on alternating direction method of multipliers," *SIAM Journal on Imaging Sciences*, vol. 6, no. 1, pp. 680–697, 2013.

[23] C.-C. Chang, C.-W. Hsu, and C.-J. Lin, "The analysis of decomposition methods for support vector machines," *IEEE Transactions on Neural Networks*, vol. 11, no. 4, pp. 1003–1008, 2000.

[24] C.-C. Chang and C.-J. Lin, "LIBSVM: a library for support vector machines," *ACM Transactions on Intelligent Systems and Technology (TIST)*, vol. 2, no. 3, p. 27, 2011.

[25] R. Chartrand and B. Wohlberg, "A nonconvex admm algorithm for group sparsity with sparse groups," in *2013 IEEE International Conference on Acoustics, Speech and Signal Processing*. IEEE, May 2013, pp. 6009–6013.

[26] P. Chen, R. Fan, and C. Lin, "A study on smo-type decomposition methods for support vector machines," *IEEE Trans. Neural Networks*, vol. 17, no. 4, pp. 893–908, 2006.

[27] P.-H. Chen, R.-E. Fan, and C.-J. Lin, "A study on SMO-type decomposition methods for support vector machines," *IEEE Transactions on Neural Networks*, vol. 17, no. 4, pp. 893–908, 2006.

[28] A. R. Conn, N. I. Gould, and P. L. Toint, "Globally convergent augmented lagrangian algorithm for optimization with general constraints and simple bounds," *SIAM Journal on Numerical Analysis*, vol. 28, no. 2, pp. 545–572, 1991.

[29] C. Cortes and V. Vapnik, "Support-vector networks," *Machine Learning*, vol. 20, no. 3, pp. 273–297, 1995.

[30] Y.-H. Dai and R. Fletcher, "New algorithms for singly linearly constrained quadratic programs subject to lower and upper bounds," *Mathematical Programming*, vol. 106, no. 3, pp. 403–421, 2006.

[31] E. D. Dolan and J. J. Moré, "Benchmarking optimization software with performance profiles," *Mathematical programming*, vol. 91, no. 2, pp. 201–213, 2002.

[32] J. Eckstein and W. Yao, "Understanding the convergence of the alternating direction method of multipliers: Theoretical and computational perspectives," *Pacific Journal of Optimization*, vol. 11, no. 4, pp. 619–644, 2015.

[33] R.-E. Fan, P.-H. Chen, and C.-J. Lin, "Working set selection using second order information for training support vector machines," *Journal of Machine Learning Research*, vol. 6, no. Dec, pp. 1889–1918, 2005.

[34] G. Fasano, G. Liuzzi, S. Lucidi, and F. Rinaldi, "A linesearch-based derivative-free approach for nonsmooth constrained optimization," *SIAM Journal on Optimization*, vol. 24, no. 3, pp. 959–992, 2014.

[35] T. Glasmachers and C. Igel, "Maximum-gain working set selection for SVMs," *Journal of Machine Learning Research*, vol. 7, no. Jul, pp. 1437–1466, 2006.

[36] L. Grippo, F. Lampariello, and S. Lucidi, "Global convergence and stabilization of unconstrained minimization methods without derivatives," *Journal of Optimization Theory and Applications*, vol. 56, no. 3, pp. 385–406, Mar 1988.

[37] D. Hajinezhad, T.-H. Chang, X. Wang, Q. Shi, and M. Hong, "Nonnegative matrix factorization using admm: Algorithm and convergence analysis," vol. 2016-May. IEEE, 2016, pp. 4742–4746.

[38] B. He, H. Yang, and S. Wang, "Alternating direction method with self-adaptive penalty parameters for monotone variational inequalities," *Journal of Optimization Theory and Applications*, vol. 106, no. 2, pp. 337–356, 2000.

[39] W. Hock and K. Schittkowski, *Test Examples for Nonlinear Programming Codes*. Berlin, Heidelberg: Springer-Verlag, 1981.

[40] M. Hong, Z.-Q. Luo, and M. Razaviyayn, "Convergence analysis of alternating direction method of multipliers for a family of nonconvex problems," *SIAM Journal on Optimization*, vol. 26, no. 1, pp. 337–364, 2016.

[41] R. Hooke and T. A. Jeeves, ""direct search" solution of numerical and statistical problems." *J. ACM*, vol. 8, no. 2, pp. 212–229, 1961.

[42] C.-W. Hsu and C.-J. Lin, "A simple decomposition method for support vector machines," *Machine Learning*, vol. 46, no. 1-3, pp. 291–314, 2002.

[43] Z. Jia, X. Cai, and D. Han, "Comparison of several fast algorithms for projection onto an ellipsoid," *Journal of Computational and Applied Mathematics*, vol. 319, pp. 320–337, 2017.

[44] B. Jiang, S. Ma, and S. Zhang, "Alternating direction method of multipliers for real and complex polynomial optimization models," *Optimization*, vol. 63, no. 6, pp. 883–898, 2014.

[45] T. Joachims, "Making large-scale support vector machine learning practical," in *Advances in Kernel Methods*, B. Schölkopf, C. J. C. Burges, and A. J. Smola, Eds. Cambridge, MA, USA: MIT Press, 1999, pp. 169–184.

[46] C. Kanzow and D. Steck, "An example comparing the standard and safeguarded augmented lagrangian methods," *Operations Research Letters*, vol. 45, no. 6, pp. 598 – 603, 2017.

[47] ——, "Quasi-variational inequalities in banach spaces: theory and augmented lagrangian methods," *arXiv preprint arXiv:1810.00406*, 2018.

[48] S. S. Keerthi, S. K. Shevade, C. Bhattacharyya, and K. R. K. Murthy, "Improvements to Platt's SMO algorithm for SVM classifier design," *Neural Computation*, vol. 13, no. 3, pp. 637–649, 2001.

[49] R. Lai and S. Osher, "A splitting method for orthogonality constrained problems," *Journal of Scientific Computing*, vol. 58, no. 2, pp. 431–449, Feb 2014.

[50] M. Leinonen, M. Codreanu, and M. Juntti, "Distributed joint resource and routing optimization in wireless sensor networks via alternating direction method of multipliers," *IEEE Transactions on Wireless Communications*, vol. 12, no. 11, pp. 5454–5467, 2013.

[51] R. Lewis and V. Torczon, "Pattern search algorithms for bound constrained minimization," *SIAM Journal on Optimization*, vol. 9, no. 4, pp. 1082–1099, 1999.

[52] ——, "Pattern search methods for linearly constrained minimization," *SIAM Journal on Optimization*, vol. 10, no. 3, pp. 917–941, 2000.

[53] ——, "A globally convergent augmented lagrangian pattern search algorithm for optimization with general constraints and simple bounds," *SIAM Journal on Optimization*, vol. 12, no. 4, pp. 1075–1089, 2002.

[54] A. P. Liavas and N. D. Sidiropoulos, "Parallel algorithms for constrained tensor factorization via alternating direction method of multipliers," *IEEE Transactions on Signal Processing*, vol. 63, no. 20, pp. 5450–5463, Oct 2015.

[55] M. Lichman, "UCI machine learning repository," 2013. [Online]. Available: http://archive.ics.uci.edu/ml

[56] C.-J. Lin, "On the convergence of the decomposition method for support vector machines," *IEEE Transactions on Neural Networks*, vol. 12, no. 6, pp. 1288–1298, 2001.

[57] ——, "Asymptotic convergence of an SMO algorithm without any assumptions," *IEEE Transactions on Neural networks*, vol. 13, no. 1, pp. 248–250, 2002.

[58] C. Lin, "A formal analysis of stopping criteria of decomposition methods for support vector machines," *IEEE Trans. Neural Networks*, vol. 13, no. 5, pp. 1045–1052, 2002.

[59] C.-J. Lin, "A formal analysis of stopping criteria of decomposition methods for support vector machines," *IEEE Transactions on Neural Networks*, vol. 13, no. 5, pp. 1045–1052, 2002.

[60] C.-J. Lin, S. Lucidi, L. Palagi, A. Risi, and M. Sciandrone, "Decomposition algorithm model for singly linearly-constrained problems subject to lower and upper bounds," *Journal of Optimization Theory and Applications*, vol. 141, no. 1, pp. 107–126, 2009.

[61] C.-J. Lin and J. J. Moré, "Newton's method for large bound-constrained optimization problems," *SIAM Journal on Optimization*, vol. 9, no. 4, pp. 1100–1127, 1999.

[62] Z. Lin, R. Liu, and Z. Su, "Linearized alternating direction method with adaptive penalty for low-rank representation," in *Advances in Neural Information Processing Systems 24*, J. Shawe-Taylor, R. S. Zemel, P. L. Bartlett, F. Pereira, and K. Q. Weinberger, Eds. Curran Associates, Inc., 2011, pp. 612–620.

[63] G. Liuzzi, S. Lucidi, and M. Sciandrone, "Sequential penalty derivative-free methods for nonlinear constrained optimization," *SIAM Journal on Optimization*, vol. 20, no. 5, pp. 2614–2635, 2010.

[64] S. Lucidi and M. Sciandrone, "On the global convergence of derivative-free methods for unconstrained optimization," *SIAM Journal on Optimization*, vol. 13, no. 1, pp. 97–116, 2002.

[65] S. Lucidi, M. Sciandrone, and P. Tseng, "Objective-derivative-free methods for constrained optimization," *Mathematical Programming*, vol. 92, no. 1, pp. 37–59, Mar 2002.

[66] S. Lucidi, L. Palagi, A. Risi, and M. Sciandrone, "A convergent decomposition algorithm for support vector machines," *Computational Optimization and Applications*, vol. 38, no. 2, pp. 217–234, 2007.

[67] ——, "A convergent hybrid decomposition algorithm model for SVM training," *IEEE Transactions on Neural Networks*, vol. 20, no. 6, pp. 1055–1060, 2009.

[68] S. Lucidi and M. Sciandrone, "A derivative-free algorithm for bound constrained optimization," *Computational Optimization and Applications*, vol. 21, no. 2, pp. 119–142, Feb 2002.

[69] ——, "A derivative-free algorithm for bound constrained optimization," *Computational Optimization and Applications*, vol. 21, no. 2, pp. 119–142, Feb 2002.

[70] L. Lukšan and J. Vlcek, "Test problems for nonsmooth unconstrained and linearly constrained optimization," *Technická zpráva*, vol. 798, 2000.

[71] O. L. Mangasarian and D. R. Musicant, "Successive overrelaxation for support vector machines," *IEEE Transactions on Neural Networks*, vol. 10, no. 5, pp. 1032–1037, 1999.

[72] J. Martínez and B. Svaiter, "A practical optimality condition without constraint qualifications for nonlinear programming," *Journal of Optimization Theory and Applications*, vol. 118, no. 1, pp. 117–133, Jul 2003.

[73] J. Moré and S. Wild, "Benchmarking derivative-free optimization algorithms," *SIAM Journal on Optimization*, vol. 20, no. 1, pp. 172–191, 2009.

[74] naught101, "Sobol random generation code," https://github.com/naught101/sobol_seq, 2017.

[75] J. Nocedal and S. Wright, *Numerical optimization*, 2nd ed., ser. Springer series in operations research and financial engineering. New York, NY: Springer, 2006.

[76] C. S. Ong, S. Canu, and A. J. Smola, "Learning with non-positive kernels," in *In Proc. of the 21st International Conference on Machine Learning (ICML)*. ACM, 2004, pp. 639–646.

[77] E. Osuna, R. Freund, and F. Girosi, "An improved training algorithm for support vector machines," in *Neural Networks for Signal Processing VII. Proceedings of the 1997 IEEE Signal Processing Society Workshop*. IEEE, 1997, pp. 276–285.

[78] V. Piccialli and M. Sciandrone, "Nonlinear optimization and support vector machines," *4OR*, vol. 16, no. 2, pp. 111–149, 2018.

[79] J. C. Platt, "Advances in kernel methods," B. Schölkopf, C. J. C. Burges, and A. J. Smola, Eds. Cambridge, MA, USA: MIT Press, 1999, ch. Fast Training of Support Vector Machines Using Sequential Minimal Optimization, pp. 185–208.

[80] R. Rockafellar, "Augmented lagrange multiplier functions and duality in nonconvex programming." *SIAM J Control*, vol. 12, no. 2, pp. 268–285, 1974.

[81] C. Saunders, M. O. Stitson, J. Weston, L. Bottou, A. Smola *et al.*, "Support vector machine reference manual," *Technical Report CSD-TR-98-03*, 1998.

[82] K. Schittkowski, Ed., *More Test Examples for Nonlinear Programming Codes*. Berlin, Heidelberg: Springer-Verlag, 1987.

[83] B. Schölkopf and A. J. Smola, *Learning with Kernels: Support Vector Machines, Regularization, Optimization, and Beyond*. Cambridge, MA, USA: MIT Press, 2001.

[84] T. Serafini and L. Zanni, "On the working set selection in gradient projection-based decomposition techniques for support vector machines," *Optimization Methods and Software*, vol. 20, no. 4-5, pp. 583–596, 2005.

[85] Y. Shen, Z. Wen, and Y. Zhang, "Augmented lagrangian alternating direction method for matrix separation based on low-rank factorization," *Optimization Methods and Software*, vol. 29, no. 2, pp. 239–263, 2014.

[86] Y. Shi, J. Zhang, B. O'Donoghue, and K. Letaief, "Large-scale convex optimization for dense wireless cooperative networks," *IEEE Transactions on Signal Processing*, vol. 63, no. 18, pp. 4729–4743, 2015.

[87] I. Sobol, "Uniformly distributed sequences with an additional uniform property," *USSR Computational Mathematics and Mathematical Physics*, vol. 16, no. 5, pp. 236 – 242, 1976.

[88] D. Steck, "Lagrange multiplier methods for constrained optimization and variational problems in banach spaces," Ph.D. dissertation, PhD thesis (submitted), University of Würzburg, 2018.

[89] N. Takahashi and T. Nishi, "Global convergence of decomposition learning methods for support vector machines," *IEEE Transactions on Neural Networks*, vol. 17, no. 6, pp. 1362–1369, 2006.

[90] R. J. Vanderbei, "Loqo: An interior point code for quadratic programming," *Optimization Methods and Software*, vol. 11, no. 1-4, pp. 451–484, 1999.

[91] S. Wang and L. Liao, "Decomposition method with a variable parameter for a class of monotone variational inequality problems," *Journal of Optimization Theory and Applications*, vol. 109, no. 2, pp. 415–429, 2001.

[92] Y. Wang, W. Yin, and J. Zeng, "Global convergence of ADMM in nonconvex nonsmooth optimization," *CoRR*, vol. abs/1511.06324, 2015.

[93] L. Xu, B. Yu, and Y. Zhang, "An alternating direction and projection algorithm for structure-enforced matrix factorization," *Computational Optimization and Applications*, pp. 1–30, 2017.

[94] Y. Xu, W. Yin, Z. Wen, and Y. Zhang, "An alternating direction algorithm for matrix completion with nonnegative factors," *Frontiers of Mathematics in China*, vol. 7, no. 2, pp. 365–384, Apr 2012.

[95] L. Zanni, "An improved gradient projection-based decomposition technique for support vector machines," *Computational Management Science*, vol. 3, no. 2, pp. 131–145, 2006.

[96] J. Zhang and B. Morini, "Solving regularized linear least-squares problems by the alternating direction method with applications to image restoration," *Electronic Transactions on Numerical Analysis*, vol. 40, pp. 356–372, 2013.

[97] Q. Zhang, D. Wang, and Y. Wang, "Convergence of decomposition methods for support vector machines," *Neurocomputing*, vol. 317, pp. 179–187, 2018.

[98] M. Šorel and M. Bartoš, "Efficient jpeg decompression by the alternating direction method of multipliers." IEEE, 2017, pp. 271–276.