Int. J. Nonlinear Anal. Appl. 14 (2023) 8, 197–215
ISSN: 2008-6822 (electronic)
http://dx.doi.org/10.22075/ijnaa.2021.23460.2543

Gradient projection algorithms for optimization problems on convex sets and application to SVM

Bessi Radhia, Harouna Soumare*

LAMSIN, ENIT, Université Tunis El Manar, Tunis, Tunisie

(Communicated by Madjid Eshaghi Gordji)

Abstract

In this paper, we present some gradient projection algorithms for solving optimization problems with a convex-constrained set. We derive the optimality condition when the convex set is a cone and under some mild assumptions, we prove the convergence of these algorithms. Finally, we apply them to quadratic problems arising in training support vector machines for the Wisconsin Diagnostic Breast Cancer (WDBC) classification problem.

Keywords: Optimization on convex cones, projection algorithm, generalized gradient projection algorithm, Euler inequation, quadratic optimization problem, Lipschitz continuous gradient, soft and hard dual SVM problem, classification of breast cancer
2020 MSC: 90C52, 47N10

1 Introduction

In this paper, we deal with a minimization problem of the form

$$\min_{\alpha\in\Omega} J(\alpha), \tag{1.1}$$

where Ω is a given non empty closed convex set of $\mathbb{R}^d$ and J is a continuous differentiable function. A necessary optimality condition for a solution $\bar{\alpha}$ to (1.1) is $-\nabla J(\bar{\alpha}) \in N_\Omega(\bar{\alpha})$, where $N_\Omega(\bar{\alpha})$ is the normal cone to Ω at $\bar{\alpha}$. This optimality condition, which becomes also sufficient if J is convex on Ω, is equivalent to the following Euler condition:

$$(\nabla J(\bar{\alpha}), \alpha - \bar{\alpha}) \geq 0, \forall\ \alpha\ \in \Omega, \tag{1.2}$$

where $(.,.)$ designates the inner product in $\mathbb{R}^n$ [15, 16]. We denote by P_Ω the projection operator on Ω, i.e. given $a \in \mathbb{R}^n$,

$$P_\Omega(a) = \arg\min_{\alpha\in\Omega} \|\alpha - a\|^2,$$

where $\|.\|$ is the Euclidean norm. Every optimal solution to the problem (1.1) satisfies also, for all $r > 0$,

$$\bar{\alpha} = P_\Omega(\alpha - r\nabla J(\bar{\alpha})). \tag{1.3}$$

*Corresponding author
Email addresses: radhia.bessi@enit.utm.tn (Bessi Radhia), soumare.harouna@enit.utm.tn (Harouna Soumare)

Received: March 2021 *Accepted:* May 2021

We recall that every point $\bar{\alpha}$ satisfying (1.2) and (1.3) is a stationary for problem (1.1). Many numerical methods can be adopted to solve problems of the form (1.1) such as penalization method, Uzawa algorithm or interior point method. The choice of each numerical method depends on the nature of Ω. The gradient-projection algorithm (GPA) is a powerful method for solving problems of type (1.1). This algorithm consists of:

- Choosing $\alpha^0 \in \Omega$.
- For $k \geq 0$, computing $\alpha^{k+1} = P_\Omega(\alpha^k - r_k \nabla J(\alpha^k))$

The sequence of step sizes (r_k) may be chosen in different ways and most of convergence results of gradient projection algorithm suppose that the gradient of J is $L-$ Lipschitz. The study of this algorithm for $r_k = r$ is constant was the subject of many works [5, 3, 15], where it was proven, for fixed $r_k = r \in]0, \frac{2}{L}[$, algorithm (GPA) converges to $\bar{\alpha}$, the unique solution to the problem (1.1) for every choice of initial vector α^0. More generally, if the sequence of step size (r_k) is such that:

$$0 < \liminf_{k\to+\infty} r_k \leq \limsup_{k\to+\infty} r_k < \frac{2}{L},$$

and if in addition, J is such that $\{\alpha \in \mathbb{R}^n \ / \ J(\alpha) \leq J(\alpha^0)\}$ is bounded, then, every cluster point of (α^k) is a stationary point of (1.1) and satisfies Euler condition [15, 16]. Convergence of (GPA) under different choices of the step size sequence (r_k) has been discussed in [18].

In the case of relatively simple structure of Ω, the authors in [13] studied (GPA) with line search strategy using exact minimization and $r_k = \arg\min_{r\in\mathbb{R}} J(P_\Omega(\alpha^k - r\nabla J(\alpha^k)))$. They proved that every cluster point of the sequence (α^k), generated by (GPA) is a stationary point of the problem (1.1). In the case of a general closed convex set, another step size rule has been proposed in [2] called generalized Armijo step-size rule to be used with (GPA). This rule is inspired by Armijo step-size proposed in [1] for unconstrained optimization problems. We propose in this work some new algorithms and then we apply them to Support Vector Machine classification problems.

Support Vector Machines (SVMs), or Vast Margin Separators stem from the work of [21] are a set of supervised learning methods which allow to solve classification or regression problems. An SVM is defined by a separating hyperplane. In the case of a binary classification, for a set of examples, each marked as belonging to one of two classes (1 and -1), the SVM constructs a hyperplane with optimal margin which classifies or separates the data better while maximizing the distance between the two classes.

Consider n training set $D = \{(x_i, y_i) \mid x_i \in \mathbb{R}^d,\ y_i \in \{-1, 1\}\}_{i=1}^n$, where x_i is the ith example of dimension d and y_i is the corresponding class. When the data is linearly separable, the goal is to find a linear separator

$$f(x_i) = w^T x_i + b, \text{ with } f(x_i) \geq 0 \text{ if } y_i = 1 \text{ and } f(x_i) \leq 0 \text{ if } y_i = -1, \quad \forall i \in \{1, ..., n\},$$

where w is the weight vector and b is the bias. The problem is rewritten

$$y_i(w^T x_i + b) \geq 0, \quad \forall i \in \{1, ..., n\}.$$

The separating hyperplane is defined by

$$f(x_i) = w^T x_i + b = 0.$$

If the data is linearly separable, we can find two parallel hyperplanes that separate the two classes, so that the distance between them is as large as possible. The region bounded by the two hyperplanes is called the margin, and the optimal (maximum margin) hyperplane is the hyperplane located midway between them. Hyperplanes are defined by the equation

$$w^T x_i + b = y_i \ (y_i = \pm 1), \quad \forall i \in \{1, ..., n\}.$$

The problem is written as an optimization problem with inequality constraints

$$\begin{array}{ll} \min\limits_{w,b} & \frac{1}{2}||w||_2^2 \\ \text{s.t.} & y_i(w^T x_i + b) \geq 1,\ \forall i \in \{1, ...n\}. \end{array} \tag{1.4}$$

In the non-linearly separable case, we use a nonlinear transformation

$$\begin{array}{rl} \phi: & \mathbb{R}^d \longrightarrow \mathbb{R}^p \\ & x \longrightarrow \phi(x) \end{array},$$

which transforms the data points into a larger space $\mathbb{R}^p$ for $p > d$, in which we have separability. The resolution of the (1.4) (primal problem) can be done directly, However, it is very difficult to solve the primal problem when d is much larger than n. One can however, pass to the dual formulation (1.5) of the problem, which is easy to solve. One of the main tasks in training SVMs is to solve its following dual quadratic optimization problem :

$$\begin{array}{ll}\min J(\alpha) & := \frac{1}{2}(A\alpha,\alpha) - (e,\alpha) \\ \alpha \in \Omega & \end{array}, \tag{1.5}$$

for $\Omega = \{\alpha \in \mathbb{R}^n,\ (y,\alpha) = 0 \text{ and } 0 \leq \alpha_i,\ \ i = 1,...,n\}$. Where $A = (a_{ij})$ is a symmetric n-square matrix defined by $a_{ij} = y_i y_j K(x_i,x_j)$, $i,j = 1,2,...,n$, where $K : \mathbb{R}^d \times \mathbb{R}^d \to \mathbb{R}$ is the Kernel function, and the vector $e \in \mathbb{R}^n$ is given by $e = \begin{pmatrix} 1 \\ \vdots \\ 1 \end{pmatrix}$. The construction of the kernel can be done using an implicit function ϕ which transforms the input space $\mathbb{R}^d$ into a Hilbert space H provided with a scalar product of higher dimension. Through this dot product, the kernel is defined by $K(x,x') = (\phi(x),\phi(x'))$, $(x,x') \in \mathbb{R}^d \times \mathbb{R}^d$.

The linear kernel corresponds to $K(x,x') = (x,x')$ and the Gaussian kernel, using o radial basis function (RBF) for which $K(x,x') = \exp(-\gamma||x-x'||^2)$, where $\gamma > 0$.

An advantage of using kernels is that they are typically tractable, even when ϕ is not. If $\bar{\alpha}$ is a solution to the quadratic minimizing problem (1.5), then the decision function, applied to a test vector x_{test}, is expressed by the kernel K and it is of the form:

$$f(x_{test}) = \sum_{i=0}^{n} \bar{\alpha}_i y_i K(x_i, x_{test}) + \bar{b},$$

where $\bar{b} = y_j - \sum_{i=1}^{n} \bar{\alpha}_i K(x_i,x_j)$, for a j for which $\bar{\alpha}_j > 0$.

Many specific algorithms were proposed to solve this quadratic problem. The Sequential Minimal Optimization (SMO) algorithm [14] is the most used one to solve numerically such problem. The main idea of (SMO) consists of restricting data to have only two elements in each iteration, it does not require any optimization software in order to solve a simple two-variable problem.

In [4] the authors combined projection algorithms with SMO to solve different forms of SVMs problem.

The rest of the paper is organized as follows: In Section 3, the set Ω is supposed to be a convex cone and two algorithms are proposed to solve the problem (1.1). In Section 4, Ω is extended to be a convex set where we study convergence of the generalized projection algorithm We establish convergence of different proposed algorithms under different assumptions of the cost function J. In order to apply the proposed algorithms on SVM problem, we give direct methods to calculate the projection on the corresponding feasible set. Proposed algorithms are implemented and applied on breast cancer diagnosis problem. We start first by reconsidering the descent algorithms for unconstrained minimizing problems.

2 A review of unconstrained descent algorithm

We consider here the case of an unconstrained problem where we have $\Omega = \mathbb{R}^d$. We begin this paper by revising gradient and Newton descent algorithms with exact line search

We suppose that J is continuously differentiable and we recall that a direction d is said to be a descent to J at the point α if it satisfies $(\nabla J(\alpha), d) < 0$.

Lemma 2.1. If J is convex, then, for all $\alpha \in \mathbb{R}^d$ and for every descent direction d of J on α satisfying $(\nabla J(\alpha), d) < 0$, we have

$$\arg\min_{r\in\mathbb{R}} J(\alpha + rd) = \arg\min_{r\in\mathbb{R}_+} J(\alpha + rd).$$

Proof . Let $\bar{r} \in \arg\min_{r\in\mathbb{R}} J(\alpha + rd)$. Then $J(\alpha + \bar{r}d) \leq J(\alpha)$. Using convexity characterization It follows that $\bar{r}(\nabla J(\alpha, d)) \leq 0$ and then $\bar{r} > 0$, since we have $(\nabla J(\alpha), d) < 0$. □

Let consider the general descent with exact line search algorithm.

Algorithm (DA)

```
• Choose $\alpha^0$
• For $k \geq 0$,
    - Choose a descent direction $d^k$ of $J$ at $\alpha^k$
    - Compute $r_k \in \arg\min_{r\in\mathbb{R}} J(\alpha^k + rd^k)$.
    - Compute $\alpha^{k+1} = \alpha^k + r_k d^k$,
- $k := k + 1$; go to step 2.
```

Results of the above lemma and the next proposition, which are valid for constrained or unconstrained optimization problems, will be useful for most proposed algorithms in this paper .

Proposition 2.2. Let J be a convex function of class C^1 admitting at least a minimum. We suppose that, for a same subsequence $n(k)$, (α^k) and (d^k) generated by algorithm (DG), converge respectively to $\bar{\alpha}$ and $\bar{d}$. Then, $(\nabla J(\bar{\alpha}), \bar{d}) = 0$

Proof . Since J is convex and continually differentiable, then, $\psi : r \in \mathbb{R}; \mapsto J(\alpha^k + rd^k)$ is convex and of class 1 on $\mathbb{R}$ and from lemma (2.1), $r_k \geq 0$. Moreover, $\psi'(r_k) = (\nabla J(\alpha^k + r_k d^k), d^k) = 0$. If for a subsequence $(r_{n(k)})$ converges to 0, from the last optimality condition and continuity of ∇J, and passing to the limit for this subsequence, we get easily $(\nabla J(\bar{\alpha}), \bar{d}) = 0$. Otherwise, there exists $\bar{r} > 0$ such that $r_{n(k)} > \bar{r}$.

We have, by construction, the sequence $J(\alpha^k)$ is decreasing and it is lower bounded, it is then convergent. For all $0 < r < \bar{r}$, using real value theorem, there exists $0 < \tilde{r}_k < r$ such that

$$\begin{array}{rcl} J(\alpha^{k+1}) & = & J(\alpha^k + r_k d^k) \\ & \leq & J(\alpha^k + rd^k) \\ & = & J(\alpha^k) + (\nabla J(\alpha^k + \tilde{r}_k d^k), d^k) \\ & = & J(\alpha^k) + \psi'(\tilde{r}_k) \leq J(\alpha^k) + \psi'(r_k) \\ & = & J(\alpha^k) \end{array} .$$

It follows:

$$J(\alpha^{k+1}) - J(\alpha^k) \leq J(\alpha^k + rd^k) - J(\alpha^k) \leq 0.$$

Passing to the limit when $n(k)$ converges to $+\infty$, we obtain

$$J(\bar{\alpha} + r\bar{d}) - J(\bar{\alpha}) = 0.$$

Since last equality is true for all $r > 0$, dividing by r and passing to the limit when r tends to 0^+ we obtain $(\nabla J(\bar{\alpha}), \bar{d}) = 0$. □

In particular, for the standard gradient descent algorithm with exact line search (DA), the descent direction is $d^k = -\nabla J(\alpha^k)$. If J is convex and if the sequence (α^k) generated by this algorithm is bounded, then for all $\bar{\alpha}$ a limit of a subsequence of (α^k), $\bar{\alpha}$ satisfies necessary and sufficient optimality condition for convex minimization problem, $\nabla f(\bar{\alpha}) = 0$, since we have $\|\nabla J(\bar{\alpha})\|^2 = 0$. Therefore, from convexity assumption, $\bar{\alpha} \in \arg\min_{\alpha\in\mathbb{R}^d} J(\alpha)$. We retrieve convergence result of gradient descent algorithm with line search line for convex problem.

Also note that if J is twice continuously differentiable, in newton method, we choose the descent direction $d^k = -\nabla^2 J(\alpha^k)\nabla J(\alpha^k)$, where we denote by $\nabla^2 J(\alpha)$ the Hessian of J at α. In the case of the constant step size $r_k = 1$, under some assumptions of J, we have local convergence of Newton method to a minimum of J on $\mathbb{R}^d$.

If J is convex and if a cluster point $\bar{\alpha}$ of the sequence (α^k), generated by Newton method with exact line search r_k, is such that $\nabla^2 J(\bar{\alpha})$ is a definite symmetric matrix, we have then $\bar{\alpha}$ is a minimum of J on $\mathbb{R}^d$.

Indeed, Newton method with exact line search r_k is algorithm (DA) with $d^k = -\nabla^2 J(\alpha^k)\nabla J(\alpha^k)$. Since J is convex and twice continuously differentiable, and for a subseqeunce (α^k) converging to $\bar{\alpha}$, then (d^k) converges to

$-\nabla^2 J(\bar{\alpha})\nabla J(\bar{\alpha})$ and from proposition (2.2), we have $(\nabla J(\bar{\alpha}), \nabla^2 J(\bar{\alpha})\nabla J(\bar{\alpha})) = 0$ and therefore $\nabla J(\bar{\alpha}) = 0$. Convexity of J implies that $\bar{\alpha}$ is a minimum of J.

Global convergence of Newton method with exact line search is guaranteed if for example J is strongly convex on $\mathbb{R}^d$ because:

- J is coercive and strictly convex, it has a unique minimum $\bar{\alpha}$
- The sequence $(J(\alpha^k))$ is decreasing by construction and lower bounded by $J(\bar{\alpha})$, as it will be clarified later, (α^k) is then bounded.
- Since, J is strongly convex, then $\nabla^2 J(\alpha)$ is definite positive for all $\alpha \in \mathbb{R}^d$
- (α^k) converges to $\bar{\alpha}$ the unique minimum of J, due to the fact that (α^k) is bounded, and every convergent subsequence of (α^k) converges to the unique minimum of J which is $\bar{\alpha}$.

3 Optimization problem on a convex cone

We suppose in this section that $\Omega \subset \mathbb{R}^n$ is a non empty closed convex cone.

3.1 Optimality condition and projection algorithms

If the feasible set is a closed and convex cone, Euler Inequality optimality condition for problem (1.1) is equivalent to the following optimality condition :

Proposition 3.1. We suppose that J is a differentiable function.

i) If $\bar{\alpha} \in \Omega$ is a solution to the problem (1.1), then

$$(\nabla J(\bar{\alpha}), \bar{\alpha}) = 0 \text{ and } P_\Omega(-\nabla J(\bar{\alpha})) = 0. \tag{3.1}$$

ii) If in addition J is convex, then $\bar{\alpha} \in \Omega$ is a solution to the problem (1.1) if and only if conditions in (3.1) are satisfied.

Proof .

i) Let $\bar{\alpha}$ be an optimal solution to the problem (1.1). Then $\bar{\alpha}$ satisfies the Euler condition:

$$(\nabla J(\bar{\alpha}), \alpha - \bar{\alpha}) \geq 0, \quad \forall\, \alpha \in \Omega.$$

Since Ω is a cone, $\alpha = r\bar{\alpha} \in \Omega$, for all $\alpha \in \Omega$ and for all $r > 0$. We have then,

$$(r-1)(\nabla J(\bar{\alpha}), \bar{\alpha}) \geq 0, \quad \forall\, r \geq 0.$$

Choosing first $r > 1$ and then $0 \leq r < 1$, we deduce easily that $(\nabla J(\bar{\alpha}), \bar{\alpha}) = 0$.
The second property is satisfied since we have

$$(0 - (-\nabla J(\bar{\alpha})), 0 - \alpha) = -(\nabla J(\bar{\alpha}), \alpha - \bar{\alpha}) \leq 0, \quad \forall\, \alpha \in \Omega.$$

From projection operator characterization we deduce that $0 = P_\Omega(-\nabla J(\bar{\alpha}))$.

ii) If J is further convex on the convex set Ω, and if $\bar{\alpha}$ satisfies (3.1), then

$$(\nabla J(\bar{\alpha}), \alpha - \bar{\alpha}) = -(0 - \nabla J(\bar{\alpha}), 0 - \alpha) \geq 0, \ \forall\, \alpha \in \Omega.$$

Therefore, $\bar{\alpha}$ is a solution to the problem (1.1).

□

The next lemma proves that $P_\Omega(-\nabla J(\alpha))$ is a descent direction of J at α, for every point $\alpha \in \Omega$.

Lemma 3.2. For all $\alpha \in \mathbb{R}^n$, we have:

i)
$$(P_\Omega(\alpha) - \alpha, P_\Omega(\alpha)) = 0 \tag{3.2}$$

ii)
$$(P_\Omega(-\nabla J(\alpha)), \nabla J(\alpha)) = -\|P_\Omega(-\nabla J(\alpha))\|^2 \leq 0. \tag{3.3}$$

Proof . Let $\alpha \in \mathbb{R}^n$.

i) The projection $P_\Omega(\alpha)$ of $\alpha \in \mathbb{R}^d$ on Ω satisfies

$$(P_\Omega(\alpha) - \alpha, P_\Omega(\alpha) - \beta) \leq 0, \quad \forall\, \beta \in \Omega.$$

In particular by choosing $\beta = rP_\Omega(\alpha) \in \Omega$, for first $r > 1$, then for $0 < r < 1$, the result follows.

ii) In particular, we have

$$(P_\Omega(-\nabla J(\alpha)) + \nabla J(\alpha), P_\Omega(-\nabla J(\alpha)) - 0) = 0.$$

Therefore,

$$(P_\Omega(-\nabla J(\alpha)), \nabla J(\alpha))) = -\|P_\Omega(-\nabla J(\alpha))\|^2 \leq 0.$$

□

Remark 3.3. From property (3.2) we can deduce that projection on the closed convex cone P_Ω satisfies :

$$(P_\Omega(\alpha) - \alpha, \beta) \geq 0, \ \forall\, \beta \ \in \Omega,$$

And hence

$$P_\Omega(r\alpha) = rP_\Omega(\alpha),, \ \forall \alpha \in \mathbb{R}^d, \text{ and } \forall\, r \geq 0.$$

We now present an optimal descent projection algorithm where the descent direction is the projection of the opposite of the gradient on the feasible set Ω. Since $P_\Omega(-\nabla(\alpha))$ is a descent direction of J on α, we have first the idea to consider the following algorithm:

Algorithm (0)

```
• Choose $\alpha^0 \in \Omega$.
• For $k \geq 0$,
    - Compute $g^k = \nabla J(\alpha^k)$, $d^k = P_\Omega(-g^k)$ and $r_k \in \arg\min_{r \in \mathbb{R}} J(\alpha^k + rd^k)$.
    - Compute $\alpha^{k+1} = \alpha^k + r_k d^k$,
  $k := k + 1$; go to step 2.
```

Remark 3.4. From lemma (2.1), it follows that $r_k \geq 0$ and consequently, $\alpha_{k+1} = \alpha_k + r_k P_\Omega(-\nabla J(\alpha^k)) \in \Omega$ when f is convex differentiable function.

Proposition 3.5. We suppose that J is C^1 and convex on Ω. Then, any cluster point $\bar{\alpha}$ of the sequence (α^k) generated by the last algorithm satisfies $P_\Omega(-\nabla J(\bar{\alpha})) = 0$.

Proof . Let $\bar{\alpha}$ a cluster point of (α^k). Then $\bar{\alpha}$ is a limit of a subsequence of (α^k). If for this subsequence, $d^k = P_\Omega(-\nabla J(\alpha^k)) = 0$, then the sequence (α^k) becomes stationary and equals to $\bar{\alpha}$ satisfying obviously $P_\Omega(-\nabla J(\bar{\alpha})) = 0$. Otherwise, d^k is a descent direction of J at α^k. Using continuity of the two operators ∇J and P_Ω, we have, $(P_\Omega(-\nabla J(\alpha^k))$ converges to $(P_\Omega(-\nabla J(\bar{\alpha})$.

Since $r_k = \arg\min_{r \in \mathbb{R}} J(\alpha^k + rd^k)$, and thanks to proposition (2.2) and lemma (3.2), we have $(P_\Omega(-\nabla J(\bar{\alpha})), \nabla J(\bar{\alpha})) = -\|P_\Omega(-\nabla J(\bar{\alpha})\|^2 = 0$. Therefore, $P_\Omega(-\nabla J(\bar{\alpha})) = 0$ as was to be proved.

□

Remark 3.6. We have proved that $\bar{\alpha}$ the cluster point of (α^k) satisfies only one of the two optimality conditions: $P_\Omega(-\nabla J(\bar{\alpha})) = 0$ and not necessary $(\bar{\alpha}, \nabla(J(\alpha)) = 0$. As we have seen in proposition (3.1), this in general is not sufficient for $\bar{\alpha}$ to be a minimum of J as shown in the following example.

Example 3.7. Consider the following minimization problem

$$\min_{\alpha\in\Omega} J(\alpha)$$

where $J : \mathbb{R}^2 \to \mathbb{R}; \alpha \mapsto \frac{1}{2}(\alpha_1^2 + \alpha_2^2) - \alpha_1 + \alpha_2$ and the convex cone

$$\Omega = \{\alpha = (\alpha_1, \alpha_2) \in \mathbb{R}^2,\ \alpha_1 \geq 0 \text{ and } \alpha_2 \geq 0\}.$$

It is clear that that $(1, 0)$ is the unique minimum of J on Ω. For $\alpha^0 = (1.5, 0)^T$, we have $g^0 = \nabla J(\alpha^0) = (\frac{1}{2}, 1)^T$ and $d^0 = P_\Omega(-g^0) = (0, 0)$. Then the sequence generated by the last algorithm gives a stationary sequence $\alpha^k = \alpha^0$ which is not the minimum of J.

To ensure the second part of optimality condition, and in order to improve the last algorithm, since Ω is a cone, at iteration k, knowing α^k, we slightly modify the last algorithm by minimizing the cost function J from 0 in the direction α^k to obtain β^k. Then we compute the optimal step size r_k corresponding to the direction $d^k = P_\Omega(-\nabla J(\beta^k))$. We obtain the following steepest descent algorithm :

Algorithm (1)

```
• Choose $\alpha^0 \in \Omega$, such that $J(\alpha_0) < J(0)$
• For $k \geq 0$,
    - Compute $t_k = \arg\min_{t\in\mathbb{R}_+} J(t\alpha^k)$, $\beta^k = t_k\alpha^k$.
    - Compute $g^k = \nabla J(\beta^k)$, $d^k = P_\Omega(-g^k))$ and $r_k = \arg\min_{r\in\mathbb{R}} J(\beta^k + rd^k)$.
    - Compute $\alpha^{k+1} = \beta^k + r_k d^k$,
  $k := k + 1$; go to step 2.
```

Proposition 3.8. We suppose that J is C^1, μ- convex and that the operator $g = \nabla J$ is L-Lipschitz continuous. Then, the sequence (α^k) generated by the algorithm (1), converges to $\bar{\alpha}$, the unique solution to problem (1.1).

Proof . Since J is strongly convex, it is then coercive and strictly convex, it admits then a unique minimum $\bar{\alpha}$ on Ω. We have,

$$J(\alpha^{k+1}) \leq J(\beta^k) \leq J(\alpha^k) \leq J(\beta^{k-1}), \quad \forall\, k \in \mathbb{N}^*.$$

The decreasing real sequences $(J(\alpha^k))$ and $(J(\beta^k))$ are lower bounded by $J(\bar{\alpha})$, they are then convergent. Last inequalities prove that

$$\lim_{k\to+\infty} J(\alpha^k) = \lim_{k\to+\infty} J(\beta^k).$$

On the other hand, from μ-convexity assumption and using Cauchy-Schawrtz inequality, we deduce, $\forall\alpha \in \Omega$

$$J(\alpha) \geq J(0) + (\nabla J(0), \alpha) + \frac{\mu}{2}\|\alpha\|^2 \geq J(0) - \|\nabla J(0)\|.\|\alpha\| + \frac{\mu}{2}\|\alpha\|^2.$$

J is then coercive. Since J is coercive and the decreasing sequence satisfies $J(\beta^k) \leq J(\bar{\alpha})$, the sequence (β^k) is bounded. For a sub-sequence if necessary, it is then convergent to $\bar{\beta}$.

Clearly, $\bar{\beta} \in \Omega$. The function J is convex and $r_k = \arg\min_{r\in\mathbb{R}} J(\beta^k + rd^k)$, using lemma (2.1), it follows that $r_k \geq 0$. $\alpha^{k+1} = \beta^k + r_k d^k$ and $\lim_{k\to+\infty} J(\alpha^{k+1}) = \lim_{k\to+\infty} J(\beta^k)$, we check easily like in proposition (2.2) that $\bar{\beta}$ is such that: $P_\Omega(-\nabla J(\bar{\beta})) = 0$. First part of optimality condition is well satisfied.

On the other hand, for all k, using μ- convexity of J we get

$$J(\alpha^k) - J(\beta^k) \geq (1-t_k)(\nabla J(\beta^k), \alpha^k) + \frac{\mu(1-t_k)^2}{2}\|\alpha^k\|^2.$$

By construction, we have : $t_k = \arg\min_{t\geq 0} \phi(t)$, for $\phi(t) = J(t\alpha^k)$. Applying optimality condition, $\phi'(t_k)(t-t_k) \geq 0$ for $t = 1 \geq 0$, we deduce that $(\nabla J(\beta^k), \alpha^k)(1-t_k) \geq 0$. Moreover, we have proved that $\lim_{k\to+\infty} J(\alpha^k) - J(\beta^k) = 0$, it follows that

$$\lim_{k\to+\infty}(1-t_k)^2\|\alpha^k\|^2 = 0.$$

Obviously, either $(1-t^k)^2$ converges to 0, or (α^k) converges to null vector.

If the last one is satisfied, continuity of ∇J yields $P_\Omega(-\nabla J(0)) = 0$. In this case 0 is the solution to problem (1.1), since it satisfies the two necessary optimality conditions $P_\Omega(-\nabla J(0)) = 0$ and $(\nabla J(0), 0) = 0$. Otherwise, (t_k) converges to 1 and $(\alpha^k) = (\frac{\beta^k}{t_k})$ converges to $\bar{\beta}$. Since we have $(\nabla J(\beta^k), \alpha^k) = 0$, passing to the limit to deduce that $(\nabla J(\bar{\beta}), \bar{\beta}) = 0$ and the necessary optimality condition (3.1) is satisfied. Uniqueness of optimal solution of problem (PG) implies that $\bar{\beta} = \bar{\alpha}$. Convergence of the sequence (α^k) to the unique solution to the minimizing problem (1.1) is then proved. □

We initialize the algorithm by choosing α^0 such that $J(\alpha^0) < J(0)$ in order to grantee that $\alpha^k \neq 0$, since we have $J(\alpha^k) < J(0)$, for all iteration k. This is due to the fact that t_k can not be defined if $\alpha^k = 0$. We can choose α^0 arbitrary in Ω. In such case, we just take $t_k = 1$ and $\beta^k = \alpha^k = 0$ if $\alpha^k = 0$.

If we take back example (3.7) and we apply algorithm (1) for the same initialisation $\alpha^0 = (1.5, 0)^T$, we obtain $\beta^0 = (1, 0)$ which is the optimal solution of problem. Solution is obtained here with just one iteration.

We can accelerate convergence of sequences (α^k) and (β^k) in algorithm (1) by combining algorithm (1) and classical gradient projection algorithm (GPA) to obtain the following algorithm (2).

Algorithm (2)

```
• Choose $\alpha^0 \in \Omega$, such that $J(\alpha_0) < J(0)$
• For $k \geq 0$,
    - Compute $t_k = \arg\min_{t\in\mathbb{R}_+} J(t\alpha^k)$, $\beta^k = t_k\alpha^k$.
    - Compute $g^k = \nabla J(\beta^k)$, $d^k = P_\Omega(-g^k))$ and $r_k = \arg\min_{r\in\mathbb{R}_+} J(\beta^k + rd^k)$.
    - Compute $\alpha^{k+1} = \begin{cases} \beta^k + r_k d^k & \text{if } J(\beta^k + r_k d^k) < J(P_\Omega(\beta^k + r_k d^k)) \\ P_\Omega(\beta^k + r_k d^k) & \text{otherwise} \end{cases}$
  $k := k+1$; go to step 2.
```

Under the same assumptions of algorithm (1) we have exactly the same steps to prove convergence of algorithm (2) to the unique solution of problem (1.1). In order to prove that $P_\Omega(-\nabla J(\bar{\alpha})) = 0$, we use in algorithm (2) this inequality $J(\alpha^{k+1}) \leq J(\beta^k + r_k d^k)$ which was an equality in proposition (3.8).

3.2 General applications

Projection on the closed convex constrained set should be easy to implement in order to be able to apply algorithms (1) or (2). Let consider the minimizing problem of the form:

$$\min_{C_e\alpha = b} J_0(\alpha) \tag{3.4}$$

with $C_e \in \mathbb{R}^m \times \mathbb{R}^n$ is a rectangular matrix of size (m, n) and $b \in \mathbb{R}^m$. With the change of variable $\alpha \to \alpha - \alpha_f$, for a feasible point α_f, the problem takes the form:

$$\min_{C_e\alpha = 0} J(\alpha), \tag{3.5}$$

where $J(\alpha) = J_0(\alpha - \alpha_f)$.

The projection on the feasible convex set $\Omega = \ker(C_e)$ is not difficult to compute and we have: $P_\Omega(a) = a + C_e^T\lambda$, for all vector λ solution to the system $C_e^T C_e\lambda = -C_e a$. In particular, if $C_e \in \mathbb{R}^n$ is a line vector, then

$$P_\Omega(a) = a - \frac{C_e.a}{\|C_e\|^2} C_e^T.$$

For inequality constraints of the form $C_I\alpha \geq 0$, projection becomes less obvious. As examples of problem (3.4) we can find problems related to active set or to interior point methods [6]. In particular, for quadratic problem:

$$\begin{array}{ll} \min J(\alpha) & = \frac{1}{2}(A\alpha, \alpha) - (e, \alpha) \\ C_e\alpha = b & \\ D_I\alpha \geq c & \end{array} \tag{3.6}$$

at each iteration k, we have to solve the constrained problem

$$\begin{array}{c} \min J(\beta) = \frac{1}{2}(A\beta, \beta) - (e, \beta) \\ C_e\beta = b \\ (d_i, \beta) = c_i, \qquad \text{for } i \in \mathcal{A}_k \end{array}, \tag{3.7}$$

where $\mathcal{A}_k = \{i \;/\; (D_I\beta)_i = (d_i, \beta) = c_i\}$ is the index set of active inequality constraints. In the case where the matrix $D_I = -I$ and the inequality constraint is $\alpha \geq 0$, interior point methods consists to solve

$$\begin{array}{c} \min \frac{1}{2}(A\beta, \beta) - (e, \beta) - \mu \sum_{j=1}^{p} \log(\beta_i) \\ C_e\beta = b \end{array}, \tag{3.8}$$

We can also apply algorithms (1) and (2) to solve quadratic sub problems in Sequential Quadratic Programming (SQP) method.

3.3 Application to hard-margin SVM problem

Here, we are interested to the quadratic programming problem arising in training hard-margin Support Vectors Machines (4.4) :

$$\begin{array}{c} \min J(\alpha) \\ \alpha \in \Omega \end{array} \tag{3.9}$$

where $J(\alpha) = \frac{1}{2}(A\alpha, \alpha) - (e, \alpha)$ and

$$\Omega = \left\{\alpha \in \mathbb{R}_+^n \;/\; (y, \alpha) = 0, \;\; \alpha_i \geq 0, \; \forall\, i = 1, ..., n\right\}.$$

It is clear that Ω is non empty closed convex cone. We propose first a direct algorithm to calculate the projection on the feasible set Ω.

3.3.1 Projection algorithm on Ω

Computing $P_\Omega(a)$, the projection of a on the closed convex set Ω, for a given vector $a \in \mathbb{R}^n$, is based on optimality conditions of the operator P_Ω.

Let $\bar{a} = P_\Omega(a) = \arg\min_{\alpha\in\Omega} \|x - a\|^2$. From Karush- Kunh and Tuker condition, there exist $\lambda \in \mathbb{R}$, $\mu \in \mathbb{R}_+^n$ such that:

$$\begin{array}{ll} \bar{a} - a + \lambda y - \mu = 0 & i = 1, ..., n \\ \mu_i\bar{a}_i = 0, & i = 1, ..., n \\ \mu_i \geq 0, & i = 1, ..., n \\ (y, \bar{a}) = 0 & \end{array} \tag{3.10}$$

In order to calculate $\bar{a} = P_\Omega(a)$, we have to establish first some of its properties. We denote the set indices of inactive inequality constraints and its length by :

$$I_+ = \{i \in [1, ..., n] \;/\bar{a}_i > 0.\} \text{ and } np = |I_+|.$$

Lemma 3.9. If $\bar{a} = P_\Omega(a)$ and λ its Lagrange multiplier associated to the equality constraint, then we have :

i)

$$a_i > \lambda y_i \text{ iff } \bar{a}_i > 0, \ \forall\, 1 \le i \le n$$

ii)

$$a_i \le a_j \text{ iff } \bar{a}_i \le \bar{a}_j, \forall i, j \ / \ yiyj = 1$$

iii)

$$\lambda = \frac{1}{np} \sum_{i \in I_+} a_i yi.$$

Proof .

i) For $i \in [1, ..., n]$ such that $\bar{a}_i > 0$, due to condition (3.10) , we get $\mu_i = 0$ and $\bar{a}_i = a_i - \lambda y_i > 0$ which yields $a_i > \lambda y_i$. Inversely, if $a_i > \lambda y_i$, then $\bar{a}_i = a_i - \lambda y_i + \mu_i > \mu_i$. If $\mu_i > 0$, necessarily $\bar{a}_i > 0$ which is impossible since $\bar{a}_i \mu_i = 0$. It follows that $\mu_i = 0$ and $\bar{a}_i = a_i - \lambda y_i > 0$. The first property is proved.

ii) The condition $y_i y_j = 1$ means that $y_i = y_j = 1$ or $y_i = y_j = -1$. In both cases, let $a_i \le a_j$. If $\bar{a}_i > 0$, then $a_i > \lambda y_i = \lambda y_j$. Therefore, $a_j > \lambda y_j$ and $\bar{a}_j > 0$. Then

$$\bar{a}_i = a_i - \lambda y_i = a_i - \lambda y_j \le a_j - \lambda y_j = \bar{a}_j.$$

If $\bar{a}_i = 0$, clearly $\bar{a}_j \ge 0 = \bar{a}_i$.

iii) Since we have, $\forall i$, $\bar{a}_i = a_i - \lambda y_i + \mu_i$, it is enough to multiply by $\bar{a}_i$ and next to sum to get easily the expression of λ.

□

According to the previous lemma, we can deduce that it is sufficient to compute λ to determine $\bar{a} = P_\Omega(a)$ The algorithm (3) is a direct method to compute $\bar{a}$: Without loss of generality we can suppose that

$$\max\{a_i;\ y_i = 1\} \ge max\{a_i;\ y_i = -1\}.$$

Otherwise we change y by its opposite.

Algorithm (3)

- Fix $a_{i_1} = \max\{a_i;\ y_i = 1\}$ and $a_{j_1} = \max\{a_j;\ y_j = -1\}$.
- Determine

$$\begin{array}{ll} I_+ = \{i \ / \ a_i > -a_{i_1} \text{ and } y_i = 1\}, & n_+ = |I_+| \\ I_- = \{j \ / \ a_{j_1} > a_j \text{ and } y_j = -1\}, & n_- = |I_-| \end{array}$$

- Compute

$$n_s = n_+ + n_- \text{ and } \lambda = \frac{\sum_{i \in I_+} a_i y_i + \sum_{i \in I_-} a_i y_i}{n_s}.$$

```
if $\min_{i \in I_+, i \le n_+} a_i \le \lambda$, take $k_1 = n_+$, $a_{i_k} = \min_{i \in I_+} a_i$
    while $k_1 > 0$ and $a_{i_{k_1}} \le \lambda$, take
        $\lambda = \frac{ns\lambda - a_{i_{k_1}}}{n_s - 1}$, $k_1 = k_1 - 1$ and $ns = ns - 1$
else:
    $k_2 = n_-$, $a_{j_{k_2}} = \min_{j \in I_-} a_j$
        while $k_2 > 0$ and $a_{j_{k_2}} \le \lambda$, take
        $\lambda = \frac{ns\lambda + a_{j_k}}{n_s - 1}$, $ns = ns - 1$, $k_2 = k_2 - 1$.
```

• Compute $\bar{a}$:

```
for  $i = 1, ..., k_1$,
    $\bar{a}_{i_k} = a_{i_k} - \lambda$.
for  $j = 1, ..., k_2$,
    $\bar{a}_{j_k} = a_{j_k} + \lambda$.
```

Knowing how to compute projection on the constrained set Ω, we can use algorithms (1) or (2) to solve the quadratic problem (4.4).

3.3.2 Experiment results

Practically, to train the SVM problem we test algorithm (1) and (2) for linear kernel, then we chose radial basis function (RBF) for proposed method in our experiments where the kernel:

$$K(x, x') = \exp(-\gamma \|x - x'\|^2).$$

The dataset used here was taken from the website of Wisconsin Prognostic Breast Cancer (WPBC) which was used in several publications in the medical literature. SVM algorithms have been applied many times on (WPBC) datasets [20, 19, 7, 12, 22, 8]. Data consists of 569 observations with 357 negative (benign) and 212 positive (malignant) observations. Each observation has 30 attributes. The data is split into a training set (80%) and testing set (20%) and it is invariant for different kernel.

Python programming language is used to code and apply different algorithms in this paper. Our objective is to compare different algorithms with K.K.T method which is in particular used and implemented in the function 'solvers.qp' of the open source software package CVXOPT of python environment for solving convex optimization problems.

For linear kernel, we tested the two algorithms comparing their results with one of numerical solution obtained with python. Using the python function 'solvers.qp' for minimizing quadratic problems, objective function value obtained is not decreasing and it reaches -1249457.382 after number of iterations $N.I = 20$, -1249440.5701 for $N.I = 100$ and -1249411.396 after 500 iterations. But obtained solutions do not satisfy the fist part of optimality condition cited in (3.1) for different tested iterations. For example, if we denote by β the obtained numerical solution, we get $(\beta, \nabla J(\beta)) = -47$ for $N.I = 100$ and 72.09 for $N.I = 2.10^4$. For both cases, cost value function $\simeq -1249.10^3$.

This can be explained by the fact that the hard SVM problem does not have an optimal solution and the data are not linearly separable or by the fact that the matrix A of our data is ill-conditioned and it satisfies $\lambda_{min}(A) << 1 << \lambda_{max}(A)$, where $\lambda_{min}(A)$ and $\lambda_{max}(A)$ are respectively the smallest and largest eigenvalues of A. More precisely, for this example, calculated with python, cond(A) $= 3.84.20 \times 10^{20}$.

However, we obtain good accuracies for training and testing sets. For example, for number iterations equal to 20 or 500, we obtain same results with training accuracy equals to 1 and testing accuracy is 0.9298.

Applying algorithms (1) and (2), experiment results give a decreasing sequence $(J(\alpha^k))$ whose speed of variation depends on the initialisation of the corresponding algorithm.

Table (1) and figure (1) illustrate these results. In table (1) we present 'qp.solver' optimal value using the K.K.T method, numerical optimal value by algorithm (1) or (2), number of iterations and maximum accuracy for training (train acc) and testing set (test acc) registered for different iterations. In figures (1) and (2) we plot accuracy variation and loss function with respect to the number of iterations for respectively algorithm (1) and (2). In the case of a number of iterations larger then 100, we present accuracy and cost value function after each 10 iterations.

	Initial value	Numerical optimal value	Iterations 's' number	train accuracy	test accuracy
'qp.solver'	-2.499128	-1249457.382865	100	1	0.93
	-2.10^{-6}	-1249457.382865	100	1	0.93
Alg (1)	-2.499128	-1246189.161718	100	0.94	0.96
	-2.10^{-6}	-4.118085	1000	0.94	0.95
Alg (2)	-2.499128	-1246662.90	100	0.93	0.96
	-2.10^{-6}	-4.538499	2000	0.92	0.965

Table 1: Results of linear kernel

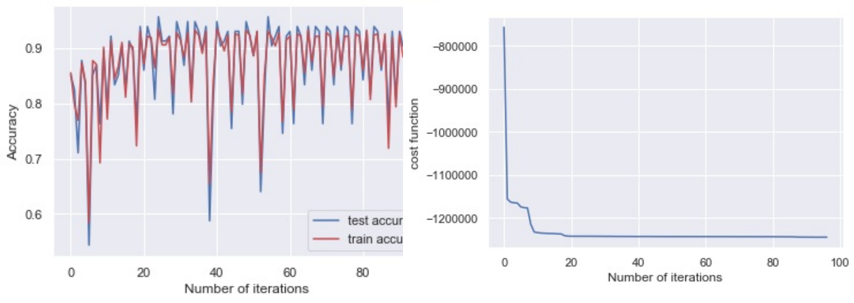

Figure 1: Algorithm (1): cost function and accuracy for linear kernel

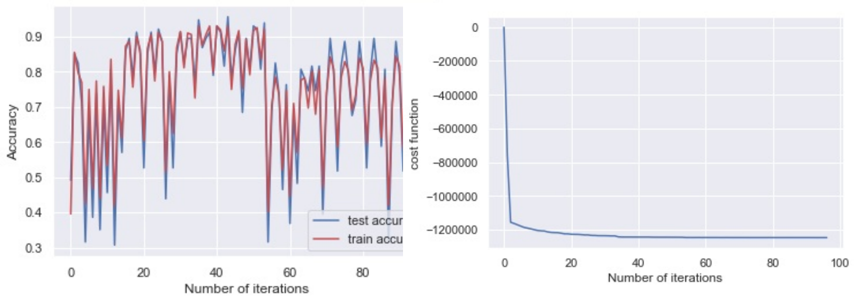

Figure 2: Algorithm (2): Cost function and accuracy for linear kernel

With RBF kernel, and for the same split of data set we apply algorithms (1) and (2) to measure the accuracy of the classification for different values of the parameter $\gamma \in \{0.1, 0.01, 0.0001\}$. For $\gamma \neq 0.0001$, we obtained 100% classification accuracy for training set. Accuracy for testing set depends on γ as follows table (2) and figures (3) to (6). Iteration number depends on matrix condition number of the matrix A.

	γ	Initial value	Numerical optimal value	iterations 's' number	train accuracy	test accuracy
'qp.solver'	0.1		-26354.728	15	1	0.956
	0.01		-870282.990	20	1	0.947
	0.0001		-476414440.850	26	1	0.91
Alg (1)	0.1	-0.5297	-23894.962	12	1	0.956
	0.01	-17.628	-517786.968	40	0.997	0.964
	0.0001	-117.1588	-6735414.960	40	0.971	0.947
Alg (2)	0.1	-0.5297	-26352.0235	12	1	0.973
	0.01	-17.628	-860188.953	20	0.996	0.964
	0.0001	-117.1588	-11245752.089	40	0.985	0.96

Table 2: Results for $\gamma \in \{0.1, 0.01, 0.0001\}$

γ	0.1	0.01	0.005	0.0001
Cond(A)	1933719617	1137659878224	7320547409175	$2.908664907747441 10^{17}$

Table 3: Condition number of A

The two algorithms gave a good classification for both testing and training set where the accuracy is better for $\gamma = 0.01$. Notice also that algorithm (2) is faster than algorithm (1).

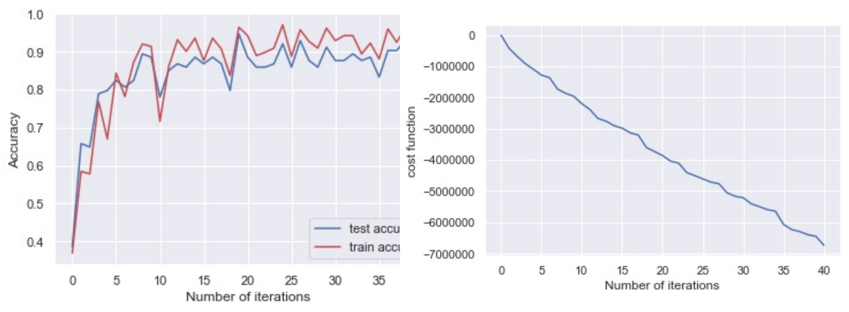

Figure 3: Algorithm (1): Cost function and accuracy for $\gamma = 0.0001$

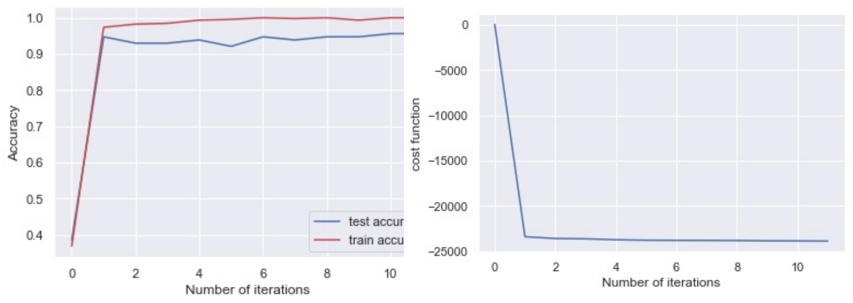

Figure 4: Algorithm (1): Cost function and accuracy for $\gamma = 0.1$

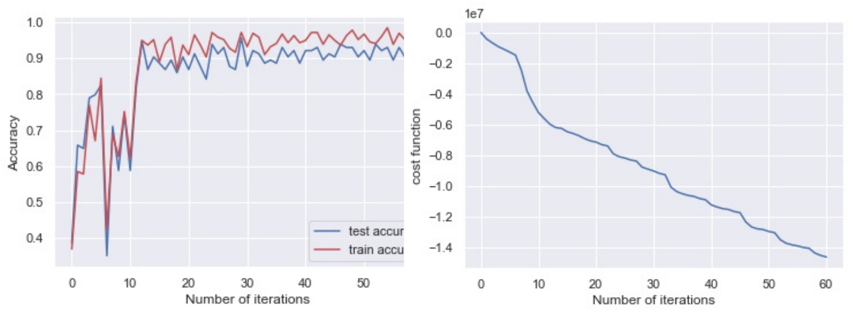

Figure 5: Alg (2):Cost function and accuracy for $\gamma = 0.0001$

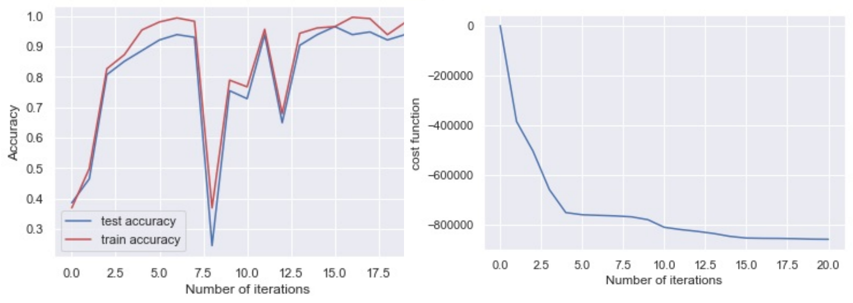

Figure 6: Alg (2): Cost function and accuracy for $\gamma = 0.01$

4 Optimization problem on a convex set

We suppose here that Ω is only a closed convex set, not necessary a cone.

4.1 Generalized projection algorithm

In what we call generalized gradient projection method, the descent direction is $d^k = \delta^k - \alpha^k$, for $\delta^k = P_\Omega(\alpha^k - \nabla J(\alpha^k))$. Let's first check that for all $\alpha \in \Omega$, $d = P_\Omega(\alpha - \nabla J(\alpha)) - \alpha$ is a descent direction.

Lemma 4.1. For all $\alpha \in \mathbb{R}^n$, we have

$$(P_\Omega(\alpha - \nabla J(\alpha)) - \alpha, \nabla J(\alpha)) \leq -\|P_\Omega(\alpha - \nabla J(\alpha)) - \alpha\|^2. \tag{4.1}$$

Proof . Let $\alpha \in \Omega$ and let $d = P_\Omega(\alpha - \nabla J(\alpha)) - \alpha$. From projection operator 's' characterization, and since $\alpha \in \Omega$, we get

$$(P_\Omega(\alpha - \nabla J(\alpha)) - (\alpha - \nabla J(\alpha)), P_\Omega(\alpha - \nabla J(\alpha)) - \alpha) = (d + \nabla J(\alpha), d) \leq 0.$$

Then, if $d \neq 0$,

$$(\nabla J(\alpha), d) \leq -(d, d) = -\|d\|^2 < 0.$$

□

We propose the following gradient projection algorithm of type minimization rule :

Algorithm (4)

- Choose $\alpha^0 \in \Omega$.
- for $k \geq 0$,
    - Compute $\delta^k = P_\Omega(\alpha^k - \nabla J(\alpha^k))$ and $d^k = \delta^k - \alpha^k$,
    - search for r_k solution of $\min\limits_{r \in [0,1]} J(\alpha^k + rd^k)$,
    - Compute $\alpha^{k+1} = \alpha^k + r_k d^k$

We study convergence of this algorithm for a function J with Lipschitz gradient. If for an iteration k we have $d^k = 0$, then the sequence (α^k) becomes stationary and converges to $\bar{\alpha}$ a stationary point of problem (PG). Otherwise:

Proposition 4.2. We suppose that J is C^1, lower bounded on Ω and with gradient $g = \nabla J$ is L-Lipschitz. we suppose that $d^k \neq 0$ for all k. Then,

i) if $L < 1$, $r_k = 1$ for all $k \in \mathbb{N}$.

ii) If $1 \leq L$, then $r_k \geq \frac{1}{L}$

iii) Every cluster point $\bar{\alpha}$ of the sequence (α^k) is a stationary point of problem (PG).

Proof . By construction, $0 \leq r_k \leq 1$. If $r_k < 1$, since $r_k = \arg\min\limits_{r \in [0,1]} J(\alpha^k + rd^k)$, then

$$(\nabla J(\alpha^k + r_k g^k), (r - r_k)d^k) \geq 0, \ \forall\, r \in \ [0,1].$$

In particular, for $r = 1$, we get

$$(\nabla J(\alpha^k + r_k d^k), d^k) \geq 0. \tag{4.2}$$

On the other hand, from property (4.1) we have :

$$\|d^k\|^2 \leq -(\nabla J(\alpha^k), d^k) \leq (\nabla J(\alpha^k + r_k d^k) - (\nabla J(\alpha^k), d^k) \leq r_k L \|d^k\|^2. \tag{4.3}$$

Since $d^k \neq 0$, simplifying by $\|d^k\| \neq 0$, we deduce that $1 \leq r_k L$.

i) If $L < 1$, then necessary $r_k = 1$. Otherwise, $1 \leq r_k L < r_k$. Contradiction, since we have $0 \leq r_k \leq 1$. .

ii) In the case where $1 \leq L$, if $0 < r_k < 1$, we have already $\frac{1}{L} \leq r_k$. If $r_k = 1$, obviously $r_k = 1 \geq \frac{1}{L}$.

iii) If $L < 1$, we retrieve the classical gradient algorithm with fixed step size $r_k = 1 < \frac{2}{L}$ and convergence is then satisfied. If $1 \geq L$, then we prove that (d^k) converges to 0.

Since $L \geq 1$, choosing $r = \frac{1}{2L} \leq 1$, and from real value theorem and inequality (4.2), there exists $0 \leq r_k \leq r = \frac{2}{L}$, such that :

$$\begin{array}{ll} J(\alpha^{k+1}) \leq J(\alpha^k + rd^k) & = J(\alpha^k) + r(\nabla J(\alpha^k + \tilde{r}_k d^k), d^k) \\ & = J(\alpha^k) + r(\nabla J(\alpha^k), d^k) + r(\nabla J(\beta^k + rd^k) - \nabla J(\alpha^k), d^k) \\ & \leq J(\alpha^k) - r\|d^k\|^2 + r^2 L \|d^k\|^2 \\ & \leq J(\alpha^k) - \frac{1}{4L}\|d^k\|^2. \end{array}.$$

The function J is lower bounded, then, the sequence $(J(\alpha^k))$ which is decreasing by construction and lower bounded is convergent. It follows that

$$0 \leq \frac{1}{4L}\|d^k\|^2 \leq J(\alpha^k) - J(\alpha^{k+1}) \quad \begin{array}{l} \to 0 \\ k \to +\infty \end{array}.$$

Finally, if $\bar{\alpha}$ a limit of a subsequence of (α^k), then $\bar{\alpha} \in \Omega$ since, Ω is a closed set. Passing to the limit and using continuity of ∇J and of the operator P_Ω we obtain $\bar{\alpha} = P_\Omega(\bar{\alpha} - \nabla J(\bar{\alpha}))$ and $\bar{\alpha}$ is then a stationary point of problem (PG).

□

Remark 4.3. If further J is convex on Ω, then, every cluster point of (α^k) is a minimum of J. If moreover J is strongly convex, then the sequence (α^k) converges to the unique minimum $\bar{\alpha}$ of J.

4.2 Soft SVM problem

For soft margin SVM problem, the cost function $J(\alpha) = \frac{1}{2}(A\alpha, \alpha) - (e, \alpha)$ is exactly the same for hard SVM problem. The constrained set Ω is slightly modified:

$$\Omega = \{\alpha \in \mathbb{R}^n, (e, \alpha) = d,\ 0 \leq \alpha_i \leq C, \forall i = 1, ..., n\},$$

where $C > 0$ is fixed non negative real. The corresponding dual quadratic problem is:

$$\begin{array}{c} \min J(\alpha) \\ \alpha \in \Omega \end{array} \tag{4.4}$$

We propose first the projection algorithm over the convex of feasible solutions set Ω.

4.2.1 Projection on Ω

For a given $a \in \mathbb{R}^n$, if we denote $\bar{a} = P_\Omega(a)$, and

$$I_0 = \{i \text{ such that } \bar{a}_i = 0\}, I_c = \{i \text{ such that } \bar{a}_i = C\} \text{ and } q = |I_c|,$$

$$I_{in} = \{i \text{ such that } 0 < \bar{a}_i < C\} \text{ and } p = |I_{in}|.$$

Writing Karush- Kunh-Tuker optimality condition for this convex problem, we can prove the following lemma.

Lemma 4.4. There exists a unique $\lambda \in \mathbb{R}$ such that

1.
$$\lambda = \frac{\displaystyle\sum_{i \in I_{in}} a_i - d + qC}{p} \quad \text{and} \quad \left\{ \begin{array}{ccc} a_i \leq \lambda & \text{iff} & \bar{a}_i = 0 \\ \lambda < a_i < C + \lambda & \text{iff} & 0 < \bar{a}_i < C \\ C + \lambda \leq a_i & \text{iff} & \bar{a}_i = C \end{array} \right. \tag{4.5}$$

2. If $a_i \leq a_j$, then $\bar{a}_i \leq \bar{a}_j$.

Proof . Since the cost function is differentiable, and the minimized problem is with linear equality and inequality constraints, Karush- Kunh-Tuker condition is satisfied and it can be written as:

$$\begin{array}{ccc} \sum_{i=1}^{n} \bar{a}_i = d & & \\ \bar{a}_i - a_i + \lambda - u_i + v_i = 0 & & i = 1, ..., n, \\ u_i \bar{a}_i = 0, & v_i(C - \bar{a}_i) = 0 & i = 1, ..., n, \\ u_i \geq 0, & v_i \geq 0 & i = 1, ..., n. \end{array}$$

1. If $i \in I_{in}$, then $0 < \bar{a}_i < C$ and therefore $u_i = v_i = 0$ and $\bar{a}_i = a_i - \lambda$.

   By summing these equalities for $i \in I_{in}$ and using the fact that $\sum_{i=1}^{n} \bar{a}_i = d$, we deduce that $\lambda = \dfrac{\sum_{i \in I_{in}} a_i - d + qC}{p}$.

   In this case $\lambda < a_i < C + \lambda$.
   If $i \in I_0$, then $\delta_i = 0$ and $a_i - \lambda = u_i \geq 0$.
   If $i \in I_c$, then $\gamma_i = 0$ and $-C + a_i - \lambda = v_i \geq 0$ and (4.5) follows.
2. Let $a_i \leq a_j$, if $i \in I_0$, then $\bar{a}_i = 0 \leq \bar{a}_j$. If $i \notin I_0$, then $\lambda \leq a_i \leq a_j$. $\bar{a}_i - \bar{a}_j = a_i - a_j$ if $i, j \in I_{in}$ and therefore $\bar{a}_i \leq \bar{a}_j$. But if $j \in I_c$, $\bar{a}_j = C$ and $\bar{a}_i \leq \bar{a}_j$.

□

It is sufficient to compute λ in order to determine $\bar{a}$. The next algorithm allows to determine λ. For this end, to simplify, we begin by ordering components of a as follows $a_{i_1} \leq a_{i_2} \leq ... \leq a_{i_{n-1}} \leq a_{i_n}$.

Algorithm(5)

1. Compute $\lambda_0 = \dfrac{\sum_{i=1}^{n} a_i - d}{N}$; $k := 1$, $p_0 = 1$, $q_0 = N$. $N_0 = N$

2. For $k \geq 0$,

- if $\lambda_k < a_{i_j} < \lambda_k + C, \;\; \forall \, p_k \leq j \leq q_k$, stop,
- else, compute

$$\beta_k = \frac{N\lambda_k - a_{i_{p_k}}}{N_k - 1}, \eta_1 = \frac{N\lambda_k - (C - a_{i_{q_k}})}{N - 1}.$$

if $a_{i_{q_k}} < \beta_k + C$, take $\lambda_{k+1} = \beta_k$ and $N_{k+1} = N_k - 1$,

else : if $a_{i_{p_k}} > \eta_1$, take $\lambda_{k+1} = \eta_k$ and $N_{k+1} = N_k - 1$.

Otherwise, take $\lambda_{k+1} = \dfrac{N_k\lambda_k - a_{i_{p_k}} + C - a_{i_{q_k}}}{N_k - 2}$ and $N_{k+1} = N_k - 2$

Remark 4.5. It is obvious that this direct method stops at most in $k_s \leq n$ iterations. In the case where $k_s < n$, then, at the iteration k_s, the multiplier $\lambda = \lambda_{k_s}$ with $I_{in} = \{i_j / p_{k_s} \leq j \leq q_{k_s}\}$ and the projection $\bar{a}$ of a on Ω is given by

$$\bar{a}_{i_j} = \left\{ \begin{array}{ccc} 0 & \text{if} & j < p_{k_s}, \\ a_{i_j} - \lambda & \text{if} & p_{k_s} \leq j \leq q_{k_s}, \\ C & \text{if} & j > q_{k_s} \end{array} \right. .$$

Otherwise for $k = n - 1$, $a_{p_{k-1}} \leq \lambda_{k-1}$ or $a_{p_{k-1}} \geq C + \lambda_{k-1}$, necessary in this case $I_{in} = \emptyset$ and $\frac{d}{C} = q \in \{1, ..., n\}$. Clearly

$$\bar{a}_{i_j} = \left\{ \begin{array}{ccc} 0 & \text{if} & j < n - p, \\ C & \text{if} & j \geq q \end{array} \right. .$$

4.2.2 Experimental results

We consider the same data set of breast cancer problem where the data is split into a training set (80%) and testing set (20%). With linear kernel we have the same ill -conditioned matrix A. On the contrary of the case of hard SVM, numerical solution obtained using the function 'solvers.qp' of python is the same after a few number of iterations (12 for $C = 1$, 13 for $C = 10$ and 18 for $C = 18$).

We tested algorithm (4) for different values of $C \in \{1, 10, 200\}$.. The initial vector α^0 is taken equal to null vector. For $C = 100$, we present accuracy and cost value function after each 5 iterations.

Classification results are provided in table (4) and in figures below.

	C	Numerical optimal value	iterations 's' number	train accuracy	test accuracy
'qp.solver'	1	-57.219	12	0.417	0.447
	10	-311.543	13	0.5	0.526
	100	-2062.733	18	0.575	0.587
Alg (4)	1	-57.219	20	0.909	0.929
	10	-311.543	100	0.91	0.93
	100	-2052.307	500	0.909	0.92

Table 4: Results of linear kernel for $C \in \{1, 10, 100\}$

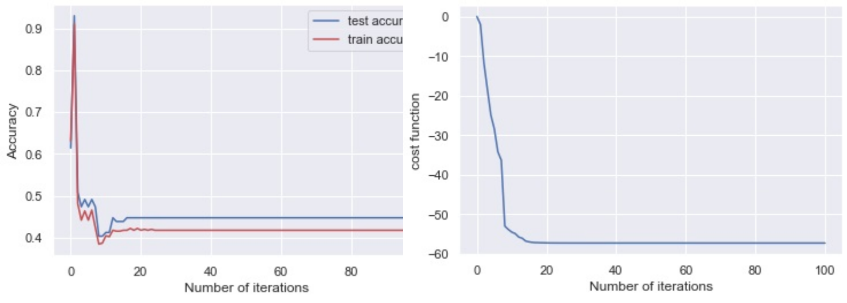

Figure 7: Algorithm (4): Cost function and accuracy for $C = 1$

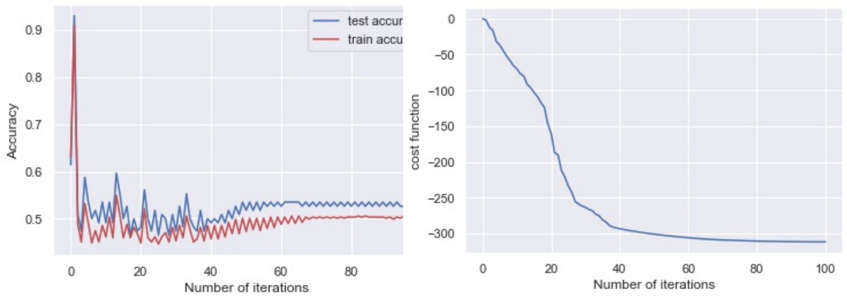

Figure 8: Alg (4): Cost function and accuracy for $C = 10$

We can remark from experimental results that for considered values of C, algorithm (4) reaches the optimal solution in less then 18 iterations for $C = 1$. However, it is very slow for $C = 10$ and $C = 100$. But classification is not perfect for testing and training sets for the number of iterations assuring convergence of the algorithm. A technique of early stopping may be necessary here to choose the α which can perform the test classification.

Conclusion

In this work, three descent projection optimization algorithms are proposed to solve a constrained minimization problem on a convex set. These algorithms were applied for classification of breast cancer using SVM method. Both linear and RBF kernel yield very promising results, in term of performance algorithm (2) for hard SVM is better and converge faster than algorithm (1). We tested algorithm (4) only for linear kernel. Despite that classification is better for algorithm (1) and (2), convergence is fater for algorithm (4).

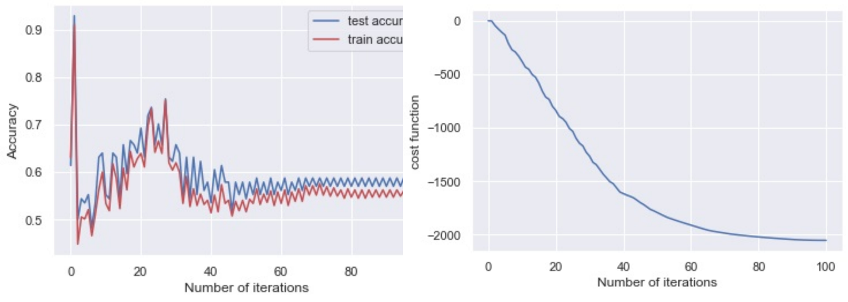

Figure 9: Algorithm (4): Cost function and accuracy for $C = 100$

References

[1] L. Armijo, *Minimization of functions having Lipschitz continuous first partial derivatives*, Pacific J. Math. **16** (1966), no. 1, 1–3.

[2] D.P. Bertsekas, *On the Goldstein-Levitin-Polyak gradient projection method*, IEEE Trans. Automatic Control **21** (1976), no. 2, 174–184.

[3] P.H. Calamai and J.J. More, *Projected gradient methods for linearly constrained problems*, Math. Program. **39** (1987), 93–116.

[4] R.E. Fan, P.H. Chen, C.J. Lin, and T. Joachims, *Working set selection using second order information for training support vector machines*, J. Machine Learn. Res. 6 (2005), no. 12, 1889–1918.

[5] E.M. Gafni and D.P. Bertsekas, *Two-metric projection methods for constrained optimization*, SIAM J. Control Optim. **22** (1984), 936–964,

[6] J. Gondzio, *Interior point methods 25 years later*, Eur. J. Oper. Res. **218** (2012), 587–601.

[7] M.W. Huang, C.W. Chen, W.C. Lin, S.W. Ke, and C.F. Tsai, *SVM and SVM ensembles in breast cancer prediction*, PloS one **12** (2017), no. 1, e0161501.

[8] S.G. Jacob and R.G. Ramani, *Efficient classifier for classification of prognostic breast cancer data through data mining techniques*, Proc. World Cong. Engin. Comput. Sci., 2012, pp. 24–26.

[9] H. Karimi, J. Nutini, and M. Schmidt, *Linear convergence of gradient and proximal-gradient methods under the polyak-łojasiewicz condition*, Machine Learn. Knowledge Discov. Databases: Eur. Conf. ECML PKDD 2016, Riva del Garda, Italy, September 19-23, 2016, Proc. Part I 16, Springer International Publishing, 2016, pp. 795–811.

[10] S.S. Keerthi, D. DeCoste, and T. Joachims, *A modified finite Newton method for fast solution of large scale linear SVMs*, J. Mach. Learn. Res. **6** (2005), no. 3, 341–361.

[11] E.S. Levitin and B.T. Polyak, *Constrained minimization problems*, USSR Comput Math. Phys. **6** (1966), 1–50.

[12] N. Liu, E.S. Qi, M. Xu, B. Gao, and G.Q. Liu, *A novel intelligent classification model for breast cancer diagnosis*, Inf. Process. Manag. **56** (2019), no. 3, 609–623.

[13] G.P. Mccormick, R.A. Tapia, *The gradient projection method under mild differentiability conditions*, SIAM J. Control **10** (1972), no. 1, 93–98.

[14] J.C. Platt, *Sequential minimal optimization: A fast algorithm for training support vector machines*, Published by Microsoft, MSR-TR-98-14, 1998.

[15] B.T. Polya, *Introduction to Optimization*, Optimization Software, New York, 1987.

[16] A. Ruszczynski, *Nonlinear Optimization*, Princeton University Press, New Jersey, 2006.

[17] F. Steinke, B. Schölkopf, and V. Blanz, *Support vector machines for 3D shape processing*, Comput. Graphics Forum **24** (1982-2005), no. 3, 285-294.

[18] M. Su and H.-K. Xu, *Remarks on the gradient-projection algorithm*, J. Nonlinear Anal. Optim. **1** (2010), 35–43.

[19] P. Taylor, J. Fox, and A. Todd-Pokropek, *Evaluation of a decision Aid for the classification of micro calcifications*, Digital Mammog.: Nijmegen **1998** (1998), 237–244.

[20] G.D. Tourassi, M.K. Markey, J.Y. Lo, and C.E. Floyd Jr, *A neural network approach to breast cancer diagnosis as a constraint satisfaction problem*, Med. Phys. **28** (2001), 804–811.

[21] V. Vapnik, I. Guyon, and T. Hastie, *Support vector machines*, Mach. Learn. **20** (1995), no. 3, 273–297.

[22] B.P. Vrigazova, *Detection of malignant and benign breast cancer using the Anova-Bootstrap-SVM*, J. Data Inf. Sci. **5** (2020), no. 2, 62–75.