



## A Filter-SQP strategy for training Support Vector Machine models

Uma estratégia Filtro-SQP para treinamento de modelos de Máquina de Vetores de Suporte

Una estrategia de Filtro-SQP para entrenar modelos de Máquinas de Vectores de Soporte



Tiago Lino Bello<sup>1</sup>

Federal University of Paraná (UFPR), Curitiba, PR, Brazil

 <https://orcid.org/0000-0002-1095-9587>,  <http://lattes.cnpq.br/1698921535580651>



Luiz Carlos Matioli<sup>2</sup>

Federal University of Paraná (UFPR), Curitiba, PR, Brazil

 <https://orcid.org/0000-0002-6506-3550>,  <http://lattes.cnpq.br/6084324625677114>



Lucas Garcia Pedroso<sup>3</sup>

Federal University of Paraná (UFPR), Curitiba, PR, Brazil

 <https://orcid.org/0009-0003-1516-119X>,  <http://lattes.cnpq.br/0906070603571189>

Daniela Miray Igarashi<sup>4</sup>

Federal University of Paraná (UFPR), Curitiba, PR, Brazil

 <https://orcid.org/0000-0002-7163-759X>,  <http://lattes.cnpq.br/8676771586561454>

**Abstract:** This paper introduces a filtering strategy for addressing optimization problems arising in binary Support Vector Machine classification. The training optimization problem aims to solve the dual formulation which involves a quadratic objective function subjected to a linear and box constraints. Our approach employs a Filter algorithm with Sequential Quadratic Programming iterations that minimize the quadratic Lagrangian approximations. Notably, we utilize the exact Hessian matrix in our numerical experiments to seek the desired classification function. Moreover, we present a Filter algorithm combined with the Augmented Lagrangian method aiming to accelerate the algorithm convergence. To substantiate our method's effectiveness, we conduct numerical experiments through MATLAB® comparing outcomes with alternative methodologies detailed in existing literature. Numerical experiments shows that the Filter–SQP combined with Augmented Lagrangian method is competitive and efficient method compared with an interior-point based solver and LIBSVM software

---

<sup>1</sup>**Brief curriculum:** Graduated in Mathematics from the State University of Paraná (UNESPAR), master and doctoral candidate in Numerical Methods in Engineering from the Federal University of Paraná (UFPR). **Authorship contribution:** Conceptualization, investigation, formal analysis, writing – original draft, writing – review and editing. **Contact:** [tiago.bello.30@gmail.com](mailto:tiago.bello.30@gmail.com).

<sup>2</sup>**Brief curriculum:** Graduated in Mathematics and master in Applied Mathematics from State University Paulista Júlio de Mesquita Filho (UNESP), PhD in Production Engineering from the Federal University of Santa Catarina (UFSC), post-doctorate from the Chinese Academy of Sciences. Professor at the Federal University of Paraná (UFPR). **Authorship contribution:** Conceptualization, investigation, formal analysis, writing – review and editing. **Contact:** [matioli@ufpr.br](mailto:matioli@ufpr.br).

<sup>3</sup>**Brief curriculum:** Bachelor in Applied and Computational Mathematics from the State University of Campinas (UNICAMP), master and PhD in Applied Mathematics from the State University of Campinas (UNICAMP). Professor at the Federal University of Paraná (UFPR). **Authorship contribution:** Conceptualization, investigation, formal analysis, writing – review and editing. **Contact:** [lucas.pedroso@ufpr.br](mailto:lucas.pedroso@ufpr.br).

<sup>4</sup>**Brief curriculum:** Graduated in Mathematics from the State University of Paraná (UNESPAR), master and doctoral candidate in Numerical Methods in Engineering from the Federal University of Paraná (UFPR). **Authorship contribution:** Conceptualization, investigation, formal analysis, writing – review and editing. **Contact:** [miray@ufpr.br](mailto:miray@ufpr.br).



in relation of classification metrics and CPU-time.

**Keywords:** support vector machine; training; optimization; filter method; sequential quadratic programming.

**Resumo:** Neste artigo, introduzimos uma estratégia de Filtro para resolver o problema de otimização decorrente da do método de classificação Máquina de Vetores de Suporte. Este problema de treinamento visa resolver a formulação dual, que envolve uma função objetiva quadrática sujeita a uma restrição lineares de igualdade e caixa. Esta abordagem aplica um algoritmo de Filtro com iterações de Programação Quadrática Sequencial, que minimizam as aproximações Lagrangiano quadráticas, usando a matriz Hessiana exata nos experimentos numéricos, em busca da função de classificação desejada. Apresenta-se um algoritmo de Filtro combinado com o método do Lagrangiano Aumentado visando acelerar a convergência do algoritmo. Também apresentamos resultados numéricos obtidos ao implementar nosso algoritmo proposto no MATLAB® e comparar os resultados com outras metodologias da literatura. Os experimentos numéricos mostram que o método de Filtro-SQP combinado com o método do Lagrangiano Aumentado é um método competitivo e eficiente em relação às métricas de classificação e tempo de CPU, em comparação com um solucionador baseado em Pontos Interiores e o software LIBSVM.

**Palavras-chave:** máquina de vetores de suporte; treinamento; otimização; método de filtro; programação quadrática sequencial.

**Resumen:** En este trabajo, exponemos una estrategia de Filtro para resolver problemas de optimización que surgen en la clasificación binaria de Máquinas de Vectores de Soporte. El problema de optimización del entrenamiento tiene como objetivo resolver la formulación dual que implica una función objetiva cuadrática sujeta a restricciones lineales y de caja. Nuestro enfoque aplica un algoritmo de Filtro con iteraciones de Programación Cuadrática Secuencial que minimizan las aproximaciones Lagrangianas cuadráticas, utilizando la matriz Hessiana exacta en los experimentos numéricos, en busca de la función de clasificación deseada. Presentamos un algoritmo de Filtro combinado con el método del Lagrangiano Aumentado con el objetivo de acelerar la convergencia del algoritmo. También presentamos resultados numéricos obtenidos al implementar nuestro algoritmo propuesto en MATLAB® y comparamos los resultados con otras metodologías en la literatura. Los experimentos numéricos muestran que el método del Filtro-SQP combinado con el método del Lagrangiano Aumentado es un método competitivo y eficiente en comparación con un solucionador basado en puntos interiores y el software LIBSVM en términos de métricas de clasificación y tiempo de CPU.

**Palabras clave:** máquina de vectores de soporte; entrenamiento; optimización; método de filtro; programación cuadrática secuencial.

**Submitted:** August 26, 2022.

**Approved:** July 5, 2023.



## 1 Introduction

Support Vector Machine (SVM) has proved to be one of the most useful approaches for machine learning. Introduced by Boser, Guyon and Vapnik [3], and Cortes and Vapnik [7] for binary classification, SVM tries to find the trade-off between the training dataset error and a maximum-margin hyperplane, aiming to achieve the best generalization ability and avoid overfitting [6, 28]. The fitted classification function is only based on a set of support vectors, which are a subset of the training data. This is the reason that SVM is often called as memory efficient. Another advantage of SVM is the use of the kernel trick, which gives the versatility to classify non-linear separable data.

The SVM training problem can be equivalently formulated as a (linearly constrained) quadratic convex problem or, by Wolfe's duality theory, as a quadratic convex problem with one linear constraint and box constraints. The training problem consists of solving large-scale convex programming problems, whose difficulties are mainly related with the number training instances, that leads to a huge number of either variables or constraints. Due to structure of SVM training problem and the convexity of the constrained problem, optimization algorithms for SVM are required to reach a solution. This is possible because the properties of the SVM training problem are well-defined from an optimization point of view [5, 28].

Thus, several approaches has been specifically developed to solve the SVM training problem. Chauhan, Dahiya and Sharma [5] and Piccialli and Sciandrone [28] provides a literature survey for methods developed to solve the SVM training problem, discussing how the properties of these problems can be incorporated in designing useful algorithms. But, regardless of the formulation, the most known SVM binary classification package is LIBSVM [4]. LIBSVM utilize a decomposition strategy based on Sequential Minimal Optimization [29], dealing with the constrained dual formulation for solving iteratively two sets of variables: a working set with two determined variables and a second set with the rest of fixed variables.

In addition to LIBSVM package, it is worth mentioning some others methods for SVM training problem. LIBLINEAR [11] is a SVM unconstrained formulation (primal or dual), using three users-choice hinge loss functions. Depending on the chosen hinge loss function, LIBLINEAR uses the coordinates algorithm [18] or Trust Region Newton Method [22]. PEGASOS [32] uses a stochastic sub-gradient method to solve primal constrained formulation with L1-hinge loss. SVM<sup>light</sup> [20] solves the dual constrained formulation using a generalized decomposition strategy. SVM<sup>perf</sup> [21] uses a cutting-plane algorithm for training a structural classification SVM, while some extensions



applies bundle methods for large scale datasets [31]. Lagrangian SVM method was proposed by Mangasarian and Musicant [23], which uses the Lagrangian function structure for linear and non-linear classification problems. Some methods are based in classical optimization approaches, such as interior-point methods [12, 36, 35] and Augmented Lagrangian [25, 38, 37], which produced good results with high dimensional optimization problems.

In this paper, we introduce and propose an algorithm to solve the dual constrained SVM problem such as [7, 29, 4]. The algorithm is based on Filter methods [13, 15, 27, 19]. The central idea is to find a point not prohibited by the filter from the current point. To find this point, the Filter method uses sequential quadratic programming (SQP) iterations, which minimize a quadratic Lagrangian approximations of the objective function. In this paper the Filter method used is based on [27] combined with the Augmented Lagrangian method to solve the feasibility step and then accelerate the algorithm convergence. In the numerical experiments, we use exact Hessian matrix, but a general matrix can also be used. The SQP iterations generates two optimization subproblems and we solve them by applying the Augmented Lagrangian and an interior-point method.

This paper is organized as follow. In Section 2, we describe the essential elements to present this paper, such as problem statement and preliminaries. In Section 3, we describes the Filter-SQP used for solving the problem. Numerical experiments are reported in Section 4. Finally, concluding remarks close our text in Section 5.

## 2 Problem statement and preliminaries

This section establishes notation and gives basic results that will be used throughout the paper. For further study about SVM, we suggest the following works [3, 7, 5, 8, 30, 33, 17, 9, 28].

Among all variations and formulations about SVM, in this paper we focus on the SVM binary classification approach, solving the dual form of SVM. Thus, the learning task of SVM is basically minimize a constrained quadratic programming problem. Giving the training instances/data  $(x_i, y_i)$ ,  $i = 1, \dots, n$ , where  $x_i \in \mathbb{R}^p$  are the observations and  $y_i \in \{-1, 1\}$  are the labels, the SVM classifica-



tion model requires the solution of the following optimization problem:

$$\begin{aligned} & \underset{\alpha}{\text{minimize}} && f(\alpha) = \sum_{i=1}^n \sum_{j=1}^n \alpha_i \alpha_j y_i y_j K(x_i, x_j) - \sum_{i=1}^n \alpha_i \\ & \text{subject to} && \sum_{i=1}^n y_i \alpha_i = 0 \\ & && 0 \leq \alpha_i \leq \mathcal{C}, \forall i \in \{1, \dots, n\}, \end{aligned} \tag{1}$$

where  $\alpha \in \mathbb{R}^n$ ,  $K(\cdot, \cdot)$  is a chosen kernel function and  $\mathcal{C} \in \mathbb{R}_+^*$  is the parameter upper bound of all variables.

Problem (1) can be reformulated equivalently in a matricial form as follows,

$$\begin{aligned} & \underset{\alpha}{\text{minimize}} && f(\alpha) = \frac{1}{2} \alpha^T P \alpha - e^T \alpha \\ & \text{subject to} && y^T \alpha = 0 \\ & && 0 \leq \alpha \leq \mathcal{C}, \end{aligned} \tag{2}$$

where  $e \in \mathbb{R}^n$  is a vector of ones,  $y \in \mathbb{R}^n, y = (y_1, \dots, y_n)^T$ , is the label vector and  $P \in \mathbb{R}^{n \times n}$  is a symmetric matrix with  $P_{i,j} = y_i y_j K(x_i, x_j)$  for  $i, j \in \{1, \dots, n\}$ .

It should be pointed some facts about the chosen kernel function in SVM problem. The dual formulation of SVM (1) gives the opportunity to apply the kernel trick [7], which measures the similarity of two points in a high-dimensional space without building it. Another interesting fact about kernel functions satisfying the Mercer’s theorem, we can guarantee that the fully dense symmetric  $P$  matrix in problem (2) is semidefinite positive [33]. This guarantees the convexity of Problem (2).

The classification function is estimated using the optimal solution,  $\alpha^*$ , of Problem (2) as follows (see [7, 8, 30]).

$$f(x) = \text{sign} \left( \sum_{i \in \mathcal{S}} \alpha_i^* y_i K(x_i, x) + b^* \right), \tag{3}$$

where  $\mathcal{S} = \{i : \alpha_i^* > 0\}$  is the index subset of support vectors of the fitted model. It should be noted that the bias value  $b^*$  in classification function (3) can be estimated by Karush-Kuhn-Tucker (KKT) conditions of Problem (2) [8, 2, 4].

In case that exists  $0 < \alpha_j^* < \mathcal{C}$ , we have a closed formula for the  $j$ -bias value,

$$b_j^* = \left( y_j - \sum_{i \in \mathcal{S}} \alpha_i^* y_i K(x_i, x_j) \right).$$



For numerical stability, we average them

$$b^* = \frac{1}{|\mathcal{I}|} \sum_{j \in \mathcal{I}} \left( y_j - \sum_{i \in \mathcal{S}} \alpha_i^* y_i K(x_i, x_j) \right), \tag{4}$$

where  $\mathcal{I} = \{j : 0 < \alpha_j^* < \mathcal{C}\}$  and  $|\mathcal{I}|$  is the number of elements in  $\mathcal{I}$ . For the case where no  $\alpha_j^*$  is between 0 and  $\mathcal{C}$ , we take  $b^*$  as the midpoint of the range generated by the KKT conditions of Problem (2),

$$\begin{aligned} & \max\{y_j \nabla f(\alpha)_j : \alpha_j^* = 0, y_j = 1 \text{ or } \alpha_j^* = \mathcal{C}, y_j = -1\} \\ & \leq b^* \\ & \leq \min\{y_j \nabla f(\alpha)_j : \alpha_j^* = 0, y_j = -1 \text{ or } \alpha_j^* = \mathcal{C}, y_j = 1\}. \end{aligned} \tag{5}$$

We emphasize that the estimate of  $b^*$  is the same as computed in LIBSVM [4].

### 3 Filter-SQP algorithm

In this section, we present the main contribution of the paper which consists in the development of a framework to solve the SVM problem, given in (2), based on the Filter method. The Filter algorithm was proposed by Fletcher and Leyffer [13] and extended by several works, but in this paper we use a general Filter algorithm as depicted in [27].

Before getting started, we will highlight some considerations adopted. Let  $c_E : \mathbb{R}^n \rightarrow \mathbb{R}$  be the function related to the equality constraint of Problem (2), defined by

$$c_E(\alpha) = y^T \alpha - \epsilon, \tag{6}$$

where  $\epsilon \in \mathbb{R}_+^*$  is as small as you want. This was assumed to guarantee numerical stability of the proposed algorithm. Also, consider  $c_I : \mathbb{R}^n \rightarrow \mathbb{R}^{2n}$  function related to the box constraints,

$$c_I(\alpha) = \begin{pmatrix} \alpha - \mathcal{C} \\ -\alpha \end{pmatrix}, \tag{7}$$

and let  $A_E \in \mathbb{R}^{(1 \times n)}$  and  $A_I \in \mathbb{R}^{(2n \times n)}$  be the Jacobian matrices associated with the functions  $c_E$  and  $c_I$ , respectively, as well the Lagrangian multipliers of these constraints functions as  $\lambda_E \in \mathbb{R}$  and  $\lambda_I \in \mathbb{R}^{2n}$ .

First, we will comment about the Filter algorithm. It is an algorithm that aims to minimize both the objective function  $f$  and a measure of infeasibility  $h : \mathbb{R}^n \rightarrow \mathbb{R}_+$ , defined here as

$$h(\alpha) = |y^T \alpha - \epsilon|, \tag{8}$$



where  $|\cdot|$  is the absolute value. At each iteration, the Filter algorithm will try to improve these criteria under a domination rule. These two criteria, objective values and infeasibility measure, together define a *forbidden region* composed by points  $(f(\alpha^k), h(\alpha^k))$ , suitably chosen from former iterations. So, we want to avoid points in the regions

$$\mathcal{R}_k = \{\alpha \in \mathbb{R}^n : f(\alpha) \geq f(\alpha^k) \text{ and } h(\alpha) \geq (1 - \beta)h(\alpha^k)\}, \tag{9}$$

where  $\beta \in (0, 1)$  is constant. The points  $\alpha^k$  which belongs the aforementioned forbidden region  $\mathcal{R}$  also compose a permanently forbidden region in  $\mathbb{R}^n$ , denoted here by  $\mathcal{F}_k$ . As the Filter algorithm is an iterative method, we also consider  $\bar{\mathcal{F}}_k = \mathcal{F}_k \cup \mathcal{R}_k$  as the temporarily forbidden region.

From now on, in order to simplify the notation, we write  $(f^k, h^k)$  to represent the pair  $(f(\alpha^k), h(\alpha^k))$ . The core idea of the Filter algorithm is to construct a sequence of filter sets  $F_0, F_1, \dots, F_k$  composed of pairs  $(f^k, h^k) \in \mathbb{R}^2$ . The temporarily forbidden region  $\bar{\mathcal{F}}_k$  is used to provide a new iterated point  $\alpha^k \notin \bar{\mathcal{F}}_k$ , which region is later updated at the end of the iteration. Thus, in the interaction  $k$  iteration, it is expected that  $\alpha^{k+1}$  must give a better results than  $\alpha^k$  for at least one criteria. In case that  $(f^{k+1}, h^{k+1})$  does not produce a reduction in  $f$ , this pair will become permanent in the filter, updating the sets by

$$F_{k+1} = \bar{F}_k \setminus \{(f^l, h^l) \in F_k, \mathcal{R}_l \subset \mathcal{R}_k\} \text{ and } \mathcal{F}_{k+1} = \bar{\mathcal{F}}_k. \tag{10}$$

As Perićaro, Ribeiro and Karas [27] mentions, the Rule (10) is applied because it removes all pairs  $(f^l, h^l) \in F_k$  which define a forbidden region that is a subset of the region related to the pair  $(f^{k+1}, h^{k+1})$ . On the other hand, if  $(f^{k+1}, h^{k+1})$  does produce a reduction in objective value, the pair  $(f^{k+1}, h^{k+1})$  is simply discarded and the filter is not updated for the next iteration.

The Filter algorithm generates a subproblem, which consists of computing a new point that not belongs to the temporarily forbidden region, that is,  $\alpha^{k+1} \notin \bar{\mathcal{F}}_k$ . Here we use the SQP algorithm, adjusting the steps to comply the filter rule. In the Filter-SQP algorithm, each iteration starts with a point  $\alpha^k$ , an estimate of the objective function  $f^k = f(\alpha^k)$ , gradient of the objective function  $\nabla f^k = \nabla f(\alpha^k)$  and  $B_k \in \mathbb{R}^{n \times n}$  a symmetric matrix, and then it computes a second order Lagrangian approximation of objective function as

$$m_k(\alpha^k + d) = f^k + (\nabla f^k)^T d + \frac{1}{2}d^T B_k d, \tag{11}$$

and a linear approximation around  $\alpha^k$  of the equality constraint, defined as

$$\mathcal{L}_k = \{\alpha^k + d \in \mathbb{R}^n : y^T \alpha^k + y^T d = 0\}. \tag{12}$$





We point out that any symmetric matrix can be used in the model  $m_k$ , but if it is possible to compute and storage the original Hessian matrix of objective function  $f$ , namely matrix  $P$  in Problem (2), the proposed algorithm will give a more accurate classifier.

The main idea of the Filter-SQP is to minimize the model  $m_k$  in the intersections of a trust region with the set  $\mathcal{L}_k$  given in (12). We will describe the essence of this idea, but for more details on SQP algorithms and trust-region methods, see [26]. Thereby, the step is computed by solving the quadratic subproblem

$$\begin{aligned}
 & \underset{d}{\text{minimize}} && m_k(\alpha^k + d) \\
 & \text{subject to} && \alpha^k + d \in \mathcal{L}_k \\
 & && \alpha^k + d \in \Omega \\
 & && \|d\|_\infty \leq \Delta_{k_r},
 \end{aligned} \tag{13}$$

where the set  $\Omega = \{\alpha \in \mathbb{R}^n : 0 \leq \alpha \leq \mathcal{C}\}$  is the box constraint in Problem (2) and  $\Delta_{k_j} > 0$  is the trust region radius. Thus, a solution  $\bar{d}_r$  of subproblem (13) will compose a new trial point  $\alpha^k + \bar{d}_r$  for the filter, which will be further evaluated. That is the reason for the  $r$  iteration index, since in the  $k$ -iteration of the Filter algorithm we can have  $r$ -iterations until the trial point be accepted.

We expect that the trial point can fulfill the filter criteria, which is to be in a non forbidden region, and provides a sufficient decrease in the condition used by trust-region algorithm. So, the radius  $\Delta$  must be chosen so that the subproblem (13) has a solution.

As Peričaro, Ribeiro and Karas [27] mentioned, the solution  $\bar{d}_r$  can be decomposed in a sum of two components: a feasibility step  $n^k$  and a tangential step  $t_{\Delta_{k_r}}$ . The feasibility step  $n^k$  must satisfy the constraints of subproblem (13), reducing the infeasibility measure  $h$ , while the tangential step  $t_{\Delta_{k_r}}$  must produce a satisfactory decrease in the model (13).

Here, we compute  $n^k$  by projecting  $\alpha^k$  into  $\mathcal{L}_k$  in the form of an optimization problem,

$$\begin{aligned}
 & \underset{n}{\text{minimize}} && \frac{1}{2} \|n\|^2 \\
 & \text{subject to} && y^T n + y^T \alpha^k - \epsilon = 0 \\
 & && 0 \leq n + \alpha^k \leq \mathcal{C}
 \end{aligned} \tag{14}$$

and we compute an approximated tangential step by solving the following optimization problem.

$$\begin{aligned}
 & \underset{t}{\text{minimize}} && (\nabla f^k + B_k n^k)^T t + \frac{1}{2} t^T B_k t \\
 & \text{subject to} && y^T t = 0 \\
 & && l \leq t \leq u
 \end{aligned} \tag{15}$$





where  $l = \min\{\alpha^k + n^k, \Delta_{k_r} + n^k\}$  and  $u = \min\{C - \alpha^k - n^k, \Delta_{k_r} - n^k\}$ .

Once the step  $\bar{d}$  is computed, then the trial point  $\alpha^k + \bar{d}_r$  must be evaluated. Thus, consider  $ared$  as a real reduction in objective function,

$$ared = f(\alpha^k) - f(\alpha^k + \bar{d}_r), \tag{16}$$

and  $pred$  as the predicted reduction by model  $m_k$ ,

$$pred = m_k(\alpha^k) - m_k(\alpha^k + \bar{d}_r). \tag{17}$$

In that way, the step  $\bar{d}_r$  will be accepted when both the real reduction is less than a fraction  $\eta \in (0, 1)$  of the predicted reduction and the predicted reduction is less than a fraction  $c_p > 0$  of the infeasibility measure, that is,

$$ared > \eta \cdot pred \quad \text{and} \quad pred \geq c_p \cdot h(\alpha^k + \bar{d}_r)^2, \tag{18}$$

because, otherwise, the model  $m_k$  poorly fits the objective function in the region delimited by  $\Delta_{k_r}$  and, in this case, we decrease the radius by  $\zeta$  and get a new model.

In the case where  $\mathcal{L} = \emptyset$  or the subproblem (13) is not compatible, that is,

$$\|n^k\| > \xi \cdot \Delta_{k_r}$$

where  $\xi \in (0, 1)$ , the proposed algorithm will call a restoration procedure, aiming to obtain a point  $\alpha^{k+1}$  not in the temporary filter but that has a low measure of infeasibility. Here, we simple apply the [16] algorithm if necessary.

Therefore, we present the Filter-SQP strategy for the training SVM Problem (2) in the Framework 1. We point out that the Jacobian matrix and the Lagrangian multipliers of the feasible set from Problem (2) are defined as  $A = (A_E, A_I)^T$  and  $\lambda = (\lambda_E, \lambda_I)^T$ , respectively. For the stopping criteria, we assume both optimality conditions and KKT conditions.

**Framework 1:** Filter-SQP strategy

**Step 0:** Choose  $\epsilon, \delta, c_p > 0$  and  $\beta, \eta, \zeta, \xi \in (0, 1)$ . Set  $k = 0, F_k = \emptyset, \mathcal{F}_k = \emptyset, \bar{\mathcal{F}}_k = \emptyset, \Delta_{k_0} \in [\Delta_{\min}, \Delta_{\max}]$ , with  $0 < \Delta_{\min} < \Delta_{\max}$ , and  $\alpha^k \in \mathbb{R}^n$ .

**Step 1:** Set  $r = 0$  and  $\mathcal{L}_k$  by (12).

If  $\mathcal{L}_k = \emptyset$ , use a restoration procedure to obtain  $\alpha^{k+1} \notin \mathcal{F}_k$  and go to *Step 3*. Else, compute the



feasibility step

$n^k$  by solving the Problem (14).

**Step 2:** If  $\|n^k\| > \xi \cdot \Delta_{k_r}$ , use a restoration procedure to obtain  $\alpha^{k+1} \notin \mathcal{F}_k$  and go to *Step 3*. Otherwise, compute the tangential step

$t_{\Delta_{k_r}}$  by solving the Problem (15),

compose the trial point

$$\alpha^k + \bar{d}_r = \alpha^k + n^k + t_{\Delta_{k_r}},$$

and evaluate the trial point. If the trial point satisfies (18), make

$$\alpha^{k+1} = \alpha^k + \bar{d}_r,$$

and go to *Step 3*. Otherwise, set  $\Delta_{k_{r+1}} = \eta \cdot \Delta_{k_r}$  and  $r = r + 1$ , and do *Step 2* again.

**Step 3:** Check if  $\alpha^{k+1}$  obey the Filter rule. If  $f(\alpha^{k+1}) \geq f(\alpha^k)$ , then update the Filter's sets by (10).

**Step 4:** Test convergence: If

$$\|P_{\mathcal{L}_k}(\alpha^{k+1} - \nabla f(\alpha^{k+1})) - \alpha^k\|_\infty \leq \delta \text{ and } h(\alpha^{k+1}) \leq \delta, \tag{19}$$

where  $P_{\mathcal{L}_k}$  is the orthogonal projection onto  $\mathcal{L}_k$ , or

$$\|\nabla f(\alpha^{k+1}) + A^T \lambda\| \leq \delta \text{ and } \|\lambda_I \circ c_I\|_\infty \leq \delta \text{ and } h(\alpha^{k+1}) \leq \delta, \tag{20}$$

where  $\circ$  is the Hadamard product, are satisfied, then stop. Otherwise, set  $k = k + 1$  and return to *Step 1*.

The Theorem 3.1 below states the convergence of the Framework 1.

**Theorem 3.1.** *The Framework 1 is well-defined and the sequence  $(\alpha^k)$  generated by it is convergent.*

*Proof.* From Step 1 and Step 2 of Framework 1 we always compute a  $\alpha^{k+1} \notin \mathcal{F}_k$ . This means that whenever the current point is not stationary, a new not forbidden point can be chosen. And since the optimization subproblems solved in (14) and (15) are quadratic in a compact set, by the Bolzano-Weierstrass theorem we always compute a solution. In [27] a more general scenario is presented. About the convergence of the Filter-SQP algorithm, notice that the objective function and constraints are at least twice differentiable and the Problem (2) is a quadratic box constraints. This means that any sequence of  $(\alpha^k)$  generated by Framework 1 always remains in a convex compact domain. The convergence of Framework 1 follows from Section 4 of [27]. □



In the next section we present numerical experiments by running the algorithm with a benchmark datasets extracted in LIBSVM website.

## 4 Numerical Experiments

In this section we illustrate the performance of our proposed algorithm. All the computational results were obtained using an Intel (R) Core i7-9750H CPU @ 2.60GHz 2.60 GHz processor, 16Gb of RAM and Windows 10 operating system, implemented in a MATLAB® software, version R2019a. The Filter-SQP method implemented in this work is available at GitHub repository [https://github.com/tiagobeautiful/Remat-Opt\\_Algorithm/](https://github.com/tiagobeautiful/Remat-Opt_Algorithm/).

### 4.1 Framework of the experiment

The problems we analyzed were extracted from LIBSVM website<sup>1</sup>. The selected datasets are depicted in Table 1. We also describe the number of instances to train and test the model, attributes, non-zero elements and the density, given by the ratio between non-zeros and  $m \times p$ . The class proportion of train and test sets is also displayed. For the case where a dataset does not have a pre-defined test set, highlighted in italic in the second column of Table 1, we split randomly the original dataset to 80% training and 20% testing.

<sup>1</sup>Available at: <https://www.csie.ntu.edu.tw/~cjlin/libsvmtools/datasets/>. Accessed in: August 2022.



Table 1 – Description of the 29 binary-class data sets

Id	Dataset	$n_{train}$	$n_{test}$	$p$	Non-zero	Density (%)	Class proportion (-/+)	
							Train	Test
1	<i>australian</i>	552	138	14	7724	79.96	54.7/45.3	58.7/41.3
2	<i>australian.scale</i>	552	138	14	8447	87.44	57.4/42.6	47.8/52.2
3	<i>breast.cancer.scale</i>	547	136	10	6830	100	63.8/36.2	69.9/30.1
4	<i>colon.cancer</i>	50	12	2000	124000	100	66.0/34.0	58.3/41.7
5	<i>diabetes</i>	615	153	8	5381	87.58	34.5/65.5	36.6/63.4
6	<i>diabetes.scale</i>	615	153	8	6135	99.85	36.3/63.7	29.4/70.6
7	<i>duke</i>	39	5	7129	313676	100	47.2/52.8	50.0/50.0
8	<i>fourclass</i>	690	172	2	1724	100	63.8/36.2	66.9/33.1
9	<i>fourclass.scale</i>	690	172	2	1717	99.59	65.4/34.6	60.5/39.5
10	<i>german.numer</i>	800	200	24	17989	74.95	70.1/29.9	69.5/30.5
11	<i>german.numer.scale</i>	800	200	24	23001	95.84	69.9/30.1	70.5/29.5
12	<i>heart</i>	216	54	13	2636	75.1	55.6/44.4	55.6/44.4
13	<i>heart.scale</i>	216	54	13	3378	96.24	56.9/43.1	50.0/50.0
14	<i>ionosphere</i>	281	70	34	10551	88.41	34.5/65.5	41.4/58.6
15	<i>mushrooms</i>	6500	1624	112	170604	18.75	48.6/51.4	46.4/53.6
16	<i>sonar</i>	167	41	60	12479	99.99	55.1/44.9	46.3/53.7
17	a1a	1605	30956	119	451592	11.65	75.4/24.6	75.9/24.1
18	a2a	2265	30296	119	451592	11.65	74.7/25.3	76.0/24.0
19	a3a	3185	29376	122	451592	11.37	75.7/24.3	75.9/24.1
20	a4a	4781	27780	122	451592	11.37	75.2/24.8	76.1/23.9
21	leukemia	38	34	7129	513288	100	28.9/71.1	41.2/58.8
22	madelon	2000	600	500	1299999	100	50.0/50.0	50.0/50.0
23	splice	1000	2175	60	190500	100	48.3/51.7	48.0/52.0
24	splice.scale	1000	2175	60	190500	100	48.3/51.7	48.0/52.0
25	svmguide1	3089	4000	4	28304	99.82	35.3/64.7	50.0/50.0
26	svmguide3	1243	41	22	22775	80.63	76.2/23.8	100.0/0.0
27	w1a	2477	47272	300	579586	3.88	97.1/2.9	97.0/3.0
28	w2a	3470	46279	300	579586	3.88	96.9/3.1	97.0/3.0
29	w3a	4912	44837	300	579586	3.88	97.1/2.9	97.0/3.0

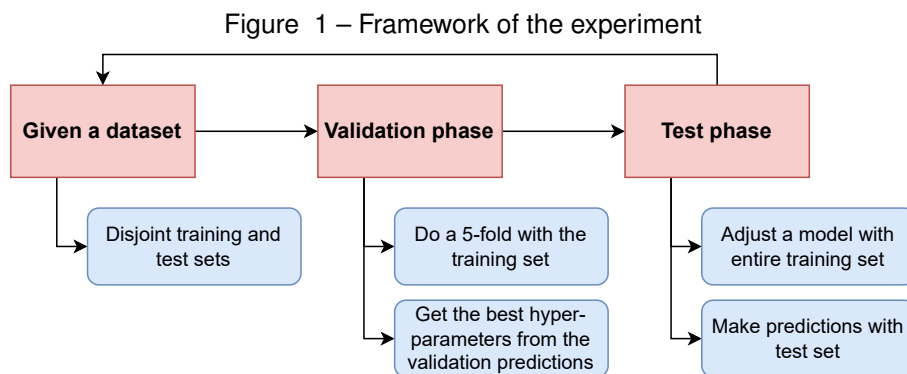
From Table 1 it is possible to see that most of the dataset has high density, which means they have many non-null elements. In case the datasets has missing data, we considered as zero and we do not perform any statistical treatment to the sparse datasets. Moreover, we noticed that some of the datasets is non balanced between the classes.



For each dataset, the experiments were run first to tune the hyper-parameters of Problem (2) and later predict the results given a test set. So, we will test all methods based on different parameters and choose the best results obtained in all cases. Aiming to get the best algorithm’s performance, we made a grid search [14] to tune  $\mathcal{C}$  and  $\gamma$  hyper-parameters in Problem (2). We choose  $\mathcal{C} \in \{0.1, 1, 10, 10^2, 10^3, 10^4, 10^5\}$  and the Gaussian radial basis function,

$$K(x_i, x_j) = \exp(-\gamma\|x_i - x_j\|^2),$$

where  $\gamma \in \{50, 10, 4, 1, 1/p, 10^{-3}, 10^{-4}, 10^{-5}, 10^{-6}\}$ , with  $p$  being the number of problem attributes, as the kernel function. The grid search will be done in a 5-fold validation [17] only with the training set. Thus, the best parameters will be chosen by the maximum average accuracy in the 5-fold validation phase. Later, we made a test phase to make predictions after a trained model with the best hyper-parameters. Figure 1 illustrate the framework of the experiment.



Source: Elaborated by the authors.

## 4.2 Performance analysis

We will test the algorithm’s performance and report metrics based on a confusion matrix. A confusion matrix is a table that is used to evaluate the performance of a classification model. It shows the number of correct and incorrect predictions made by the model compared to the actual labels of the dataset. The matrix is constructed by counting the occurrences of true positive (TP), true negative (TN), false positive (FP), and false negative (FN) predictions. The true positive represents the cases where the model correctly predicted the positive class, while the true negative represents the correct predictions of the negative class. False positive occurs when the model incorrectly predicts the positive class, and false negative occurs when the model incorrectly predicts the negative class.



The matrix provides a comprehensive view of the model's performance, enabling the calculation of various evaluation metrics. In this paper, we analyse:

- *Accuracy*: measures the ratio of correctly classified observations by total observations from the sample. It is calculated by

$$\text{Accuracy} = \frac{TP + TN}{TP + TN + FP + FN};$$

- *Balanced accuracy*: measures the same as accuracy, but it takes into account the balance of classes in the dataset, defined as

$$\text{Balanced accuracy} = \frac{1}{2} \left( \frac{TP}{TP + FN} + \frac{TN}{TN + FP} \right);$$

- *Matthews correlation coefficient*: introduced by Matthews [24], this coefficient evaluate the predictions from the classification model. It is defined by

$$\phi = \frac{TP \cdot TN - FP \cdot FN}{\sqrt{(TP + FP) \cdot (TP + FN) \cdot (TN + FP) \cdot (TN + FN)}}.$$

The Matthews correlation coefficient penalizes the FP and FN predictions, and it is a better evaluator if the classes have different sizes [1]. The  $\phi$  coefficient ranges from -1 to 1, which  $\phi = 1$  indicate the perfect prediction,  $\phi = 0$  no better than a random choice and  $\phi = -1$  indicate the inverse classification.

We consider the case of a draw in the results. For that, consider the ratio as the percentage misclassified instances rate (MIR) in the test phase for a method,

$$\text{MIR} = \frac{FN + FP}{TP + TN + FP + FN} \cdot 100.$$

MIR display the fraction of incorrect predictions. When we get the absolute difference of MIR ratio between methods under 2%, it will be considered a draw. Since MIR depends on the number of test instances, taking the absolute difference of MIR values show how discrepant the methods behaved. So 2% is a reasonable choice for this comparison.

We also plot a performance profile based on [10], displayed in log2 scale. Given a CPU runtime  $v$ , for a  $k \in \mathcal{P}$  problems and a  $m \in \mathcal{M}$  method, the performance ratios used in this comparative study are defined by



$$\text{Performance ratio} = \frac{v_{k,m}}{\min\{v_{k,j} : j \in M\}},$$

and the overall evaluation performance of a particular solver  $m$  is given by

$$\text{Overall performance} = \frac{1}{n_p} \text{card}\{k \in \mathcal{P} : r_{s,k} \leq \kappa\},$$

where  $n_p$  is the number of problems in the set  $\mathcal{P}$ .

### 4.3 Algorithms utilized

The generated optimization problems were solved by the proposed algorithm, LIBSVM version 3.25 and MATLAB® *quadprog* function in default mode.

About the proposed algorithm, we adopted the tolerance  $\delta$  of the stopping criteria (19) and (20) as  $10^{-5}$ . Moreover, we limited the Filter algorithm to the maximum of 50 iterations. The parameters from [27] Filter-SQP algorithm are described as follows:  $\epsilon = 10^{-8}$ ,  $\beta = 0.1$ ,  $\alpha^0 = 0$ ,  $B_k = \nabla^2 f(\alpha) = P$ ,  $\Delta_0 = 10^6 \cdot \min\{\|\nabla f(\alpha^0)\|, h(\alpha^0)\}$ ,  $\xi = 0.8$ ,  $\eta = 0.01$ ,  $c_p = 10^{-4}$  and  $\zeta = 0.5$ .

In order to solve the feasibility step  $n^k$ , as in the quadratic Problem (14), we apply two different algorithms to compare the performance of Filter algorithm. The first one is the *quadprog* routine of MATLAB®, which will be called *Filter-A*, and the other being an Augmented Lagrangian based method detailed in [34], which will be called *Filter-B*. The advantage of [34] method is that it determine the problem's solution by a closed formula depending only on sum and product of vectors.

For the evaluation of the tangential step  $t_{\Delta}$ , we considered the infinity norm in the definition of the trust regions. To solve the quadratic subproblem (15) in Filter-A and Filter-B, it was applied the *quadprog* routine of MATLAB® using the interior-point algorithm. Although Problem (2) has the same characteristics compared to subproblem (15), these problems are quite different in the context.

For the sake of simplicity we list the acronyms on Table 2.





Table 2 – Synthesis of algorithms utilized in this experiment

Method	Strategy	Observation
Filter-A	Filter-SQP	(14) and (15) solved by quadprog
Filter-B	Filter-SQP	(14) solved by [34] (15) solved by quadprog
quadprog	Interior-point	default mode
LIBSVM	Decomposition	–

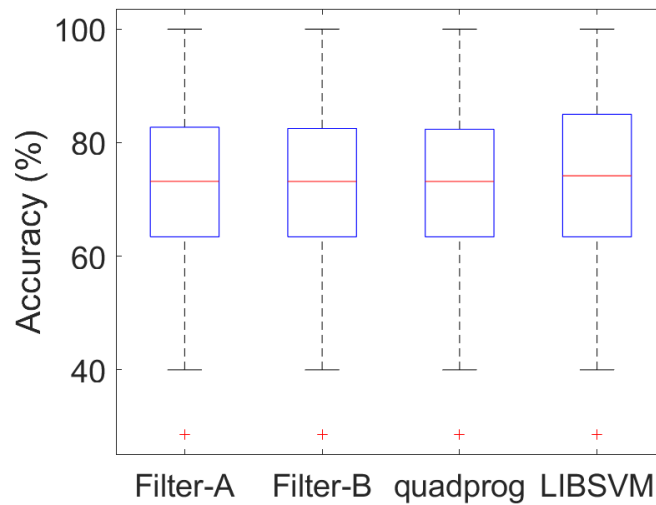
#### 4.4 Results and discussion

Regarding the parameter tuning, for each algorithm and dataset, we tried to solve 315 problems. In terms of success to find the solution, 100% of problems converged for LIBSVM and *quadprog*. The proposed algorithm, Filter-A and Filter-B, converged for 99.4% and 98.12% of problems, respectively. Although the current solution appeared to be optimal in both cases, Filter-A and Filter-B were not able to meet the stopping criteria (19)-(20). Thus, for the cases where Filter-A and Filter-B did not reach the solution, we excluded the problems in the analysis.

Figure 2 shows a boxplot of resulting accuracy with the grid search realized, which was the metric chosen as the best parameters. It is remarkable that Filter-A and Filter-B are competitive with *quadprog* and LIBSVM, as the median accuracy and the 3rd quartile of Filter-A and Filter-B are both slightly smaller than the LIBSVM algorithm. Because of  $\gamma$ 's values in RBF kernel, lower accuracy values were expected in the evaluation (represented in the boxplot as the minimum value and outliers). We must note that both Filter's based algorithms have a similar behavior when compared to *quadprog*.



Figure 2 – Boxplot of accuracy in grid search process for 29 datasets



Source: Elaborated by the authors.

Once the  $\gamma$  and  $C$  parameters were adjusted, we advance to the test phase. All optimization problems generated were run for all algorithms, which the prediction results are compiled as follows, reporting the number of support vectors, accuracy, balanced accuracy and, when it is possible, the  $\phi$  coefficient. For each table presented, the best value is in bold.

In Table 3 we depicted the amount of support vectors and cpu-time for training the model. Identify a small subset of crucial training examples called support vectors play a vital role in defining the decision boundary and can significantly reduce the complexity of the model. From Table 3, the number of support vectors and the training time of a SVM can vary depending on the algorithm employed, but all optimization algorithm used in this comparison got similar values in general.



Table 3 – Support vectors and train cpu-time comparisons of Filter-A, Filter-B, *quadprog* and LIBSVM with RBF kernel function. In the table, “A = Filter-A”, “B = Filter-B”, “C=*quadprog*” and “D=LIBSVM”

Id	$(n_{train}, n_{test}, p)$	Support vectors				Train Time			
		A	B	C	D	A	B	C	D
1	(552, 138, 14)	265	236	236	<b>204</b>	2.32	<b>0.008</b>	0.064	0.927
2	(552, 138, 14)	<b>210</b>	<b>210</b>	<b>210</b>	<b>210</b>	0.127	<b>0.009</b>	0.039	<b>0.009</b>
3	(547, 136, 10)	<b>45</b>	55	55	48	3.01	0.009	0.036	<b>0.003</b>
4	(50, 12, 2000)	<b>40</b>	<b>40</b>	<b>40</b>	<b>40</b>	0.037	<b>0.001</b>	0.002	0.007
5	(615, 153, 8)	315	<b>306</b>	315	336	1.059	<b>0.012</b>	0.042	0.017
6	(615, 153, 8)	<b>354</b>	<b>354</b>	<b>354</b>	<b>354</b>	0.148	<b>0.011</b>	0.048	0.013
7	(36, 8, 7129)	<b>34</b>	<b>34</b>	<b>34</b>	<b>34</b>	0.048	<b>0.003</b>	0.004	0.017
8	(690, 172, 2)	<b>40</b>	<b>40</b>	<b>40</b>	110	0.615	0.015	0.057	<b>0.004</b>
9	(690, 172, 2)	<b>184</b>	<b>184</b>	<b>184</b>	186	0.182	0.015	0.061	<b>0.009</b>
10	(800, 200, 24)	446	446	446	<b>380</b>	1.792	<b>0.023</b>	0.073	0.289
11	(800, 200, 24)	482	482	482	<b>405</b>	2.981	<b>0.023</b>	0.092	0.14
12	(216, 54, 13)	<b>84</b>	106	<b>84</b>	85	0.939	<b>0.001</b>	0.008	0.013
13	(216, 54, 13)	170	170	170	<b>96</b>	0.036	<b>0.002</b>	0.006	<b>0.002</b>
14	(281, 70, 34)	<b>52</b>	<b>52</b>	<b>52</b>	70	0.353	<b>0.002</b>	0.009	0.003
15	(6500, 1624, 112)	<b>6500</b>	<b>6500</b>	<b>6500</b>	<b>6500</b>	12.91	<b>4.874</b>	12.511	16.001
16	(167, 41, 60)	<b>80</b>	82	82	82	0.236	<b>0.001</b>	0.004	0.003
17	(1605, 30956, 123)	766	760	760	<b>645</b>	88.095	0.216	5.87	<b>0.155</b>
18	(2265, 30296, 123)	1081	1071	1105	<b>876</b>	890.321	0.465	17.885	<b>0.371</b>
19	(3185, 29376, 123)	1430	1430	1763	<b>1184</b>	576.783	<b>0.757</b>	39.005	1.471
20	(4781, 27780, 123)	3743	2402	4781	<b>1774</b>	7682.521	1.732	124.629	<b>1.217</b>
21	(38, 34, 7129)	30	30	<b>29</b>	30	0.12	<b>0.018</b>	0.03	<b>0.018</b>
22	(2000, 600, 500)	<b>2000</b>	<b>2000</b>	<b>2000</b>	<b>2000</b>	6.641	<b>1.923</b>	3.378	2.343
23	(1000, 2175, 60)	<b>480</b>	<b>480</b>	481	743	22.266	<b>0.078</b>	1.237	0.1
24	(1000, 2175, 60)	608	608	605	<b>592</b>	5.971	<b>0.084</b>	0.954	0.094
25	(3089, 4000, 4)	2477	2171	2667	<b>256</b>	64.757	<b>0.09</b>	3.471	0.242
26	(1243, 41, 22)	471	<b>436</b>	<b>436</b>	<b>436</b>	4586.374	0.571	65.694	<b>0.065</b>
27	(2477, 47272, 300)	824	193	247	<b>162</b>	952.892	0.335	25.111	<b>0.073</b>
28	(3470, 46279, 300)	1195	1195	273	<b>247</b>	676.845	0.59	84.699	<b>0.125</b>
29	(4912, 44837, 300)	306	402	325	<b>278</b>	1764.449	1.173	213.672	<b>0.251</b>



As for accuracy values, displayed in Table 4, in 10 problems the resulting values were the same for 4 methods (Ids 2, 3, 4, 7, 9, 13, 15, 21, 22, 27), with LIBSVM method getting better values in 13 occasions than Filter-A, Filter-B and quadprog (Ids 1, 8, 14, 16, 17, 18, 19, 20, 23, 24, 26, 28, 29). As for the balanced accuracy and  $\phi$  coefficient results, depicted in Table 4 and 5, respectively, it was reported that in 8 datasets the algorithms got the same values (Ids 2, 4, 7, 9, 13, 15, 21, 22) for both metrics. It is important to note that these 8 datasets got the same values of accuracy as well for all algorithms in the experiment. It is also noteworthy that, for this numerical experiment, Filter-A and Filter-B got higher values in 11 datasets for balanced accuracy than LIBSVM (Ids 3, 8, 10, 12, 14, 16, 17, 18, 27, 28, 29), while getting lower values in 12 datasets in relation of  $\phi$  coefficient values (Ids 1, 6, 8, 12, 14, 16, 17, 20, 23, 24, 28, 29).



Table 4 – Accuracy and balanced accuracy comparisons of Filter-A, Filter-B, *quadprog* and LIBSVM with RBF kernel function. In the table, “A = Filter-A”, “B = Filter-B”, “C=*quadprog*” and “D=LIBSVM”

Id	$(n_{train}, n_{test}, p)$	Accuracy				Balanced Accuracy			
		A	B	C	D	A	B	C	D
1	(552, 138, 14)	0.711	0.763	0.763	<b>0.812</b>	0.363	0.448	0.448	<b>0.611</b>
2	(552, 138, 14)	<b>0.866</b>	<b>0.866</b>	<b>0.866</b>	<b>0.866</b>	<b>0.719</b>	<b>0.719</b>	<b>0.719</b>	<b>0.719</b>
3	(547, 136, 10)	<b>0.963</b>	<b>0.956</b>	<b>0.956</b>	<b>0.949</b>	<b>0.482</b>	0.471	0.471	0.459
4	(50, 12, 2000)	<b>0.757</b>	<b>0.757</b>	<b>0.757</b>	<b>0.757</b>	<b>0.571</b>	<b>0.571</b>	<b>0.571</b>	<b>0.571</b>
5	(615, 153, 8)	0.707	<b>0.711</b>	<b>0.711</b>	0.659	0.762	0.752	0.752	<b>0.790</b>
6	(615, 153, 8)	<b>0.681</b>	0.681	0.681	0.675	0.791	0.791	0.791	<b>0.834</b>
7	(36, 8, 7129)	<b>0.875</b>	<b>0.875</b>	<b>0.875</b>	<b>0.875</b>	<b>0.727</b>	<b>0.727</b>	<b>0.727</b>	<b>0.727</b>
8	(690, 172, 2)	0.996	0.996	0.996	<b>1.000</b>	<b>0.500</b>	<b>0.500</b>	<b>0.500</b>	0.498
9	(690, 172, 2)	<b>1.000</b>	<b>1.000</b>	<b>1.000</b>	<b>1.000</b>	<b>0.567</b>	<b>0.567</b>	<b>0.567</b>	<b>0.567</b>
10	(800, 200, 24)	0.725	<b>0.759</b>	0.725	0.728	0.311	<b>0.443</b>	0.390	0.343
11	(800, 200, 24)	<b>0.705</b>	<b>0.705</b>	0.695	0.677	0.311	0.311	<b>0.316</b>	0.265
12	(216, 54, 13)	<b>0.808</b>	<b>0.808</b>	<b>0.808</b>	0.804	<b>0.529</b>	<b>0.529</b>	<b>0.529</b>	0.500
13	(216, 54, 13)	<b>0.889</b>	<b>0.889</b>	<b>0.889</b>	<b>0.889</b>	<b>0.587</b>	<b>0.587</b>	<b>0.587</b>	<b>0.587</b>
14	(281, 70, 34)	0.884	0.867	0.867	<b>0.914</b>	0.769	<b>0.777</b>	<b>0.777</b>	0.774
15	(6500, 1624, 112)	<b>1.000</b>	<b>1.000</b>	<b>1.000</b>	<b>1.000</b>	<b>0.698</b>	<b>0.698</b>	<b>0.698</b>	<b>0.698</b>
16	(167, 41, 60)	0.898	0.898	0.898	<b>0.925</b>	<b>0.712</b>	<b>0.712</b>	<b>0.712</b>	0.700
17	(1605, 30956, 123)	0.831	0.830	0.830	<b>0.844</b>	<b>0.777</b>	<b>0.777</b>	<b>0.777</b>	0.754
18	(2265, 30296, 123)	0.829	0.829	0.829	<b>0.842</b>	<b>0.791</b>	<b>0.791</b>	<b>0.791</b>	0.767
19	(3185, 29376, 123)	0.828	0.828	0.827	<b>0.835</b>	0.795	0.795	<b>0.796</b>	0.752
20	(4781, 27780, 123)	0.838	0.838	0.838	<b>0.846</b>	0.741	0.740	0.739	<b>0.759</b>
21	(38, 34, 7129)	<b>0.824</b>	<b>0.824</b>	<b>0.824</b>	<b>0.824</b>	<b>0.786</b>	<b>0.786</b>	<b>0.786</b>	<b>0.786</b>
22	(2000, 600, 500)	<b>0.683</b>	<b>0.683</b>	<b>0.683</b>	<b>0.683</b>	<b>0.683</b>	<b>0.683</b>	<b>0.683</b>	<b>0.683</b>
23	(1000, 2175, 60)	0.892	0.892	0.892	<b>0.903</b>	0.894	0.894	0.894	<b>0.904</b>
24	(1000, 2175, 60)	0.895	0.895	0.895	<b>0.897</b>	0.896	0.896	0.896	<b>0.897</b>
25	(3089, 4000, 4)	0.971	0.971	0.971	<b>0.973</b>	-	-	-	-
26	(1243, 41, 22)	0.780	<b>0.805</b>	<b>0.805</b>	0.756	-	-	-	-
27	(2477, 47272, 300)	<b>0.977</b>	<b>0.977</b>	<b>0.977</b>	<b>0.977</b>	<b>0.769</b>	0.768	0.768	0.751
28	(3470, 46279, 300)	0.979	0.979	0.979	<b>0.981</b>	<b>0.779</b>	<b>0.779</b>	0.778	0.748
29	(4912, 44837, 300)	0.980	0.980	0.980	<b>0.982</b>	<b>0.803</b>	<b>0.803</b>	<b>0.800</b>	0.768



Table 5 – Matthews correlation values comparisons of Filter-A, Filter-B, *quadprog* and LIBSVM with RBF kernel function. In the table, “A = Filter-A”, “B = Filter-B”, “C=*quadprog*” and “D=LIBSVM”

Id	$(n_{train}, n_{test}, p)$	$\phi$			
		A	B	C	D
1	(552, 138, 14)	0.474	0.546	0.546	<b>0.615</b>
2	(552, 138, 14)	<b>0.748</b>	<b>0.748</b>	<b>0.748</b>	<b>0.748</b>
3	(547, 136, 10)	<b>0.890</b>	0.88	0.885	0.881
4	(50, 12, 2000)	<b>0.507</b>	<b>0.507</b>	<b>0.507</b>	<b>0.507</b>
5	(615, 153, 8)	0.425	<b>0.429</b>	<b>0.429</b>	0.340
6	(615, 153, 8)	0.359	0.359	0.359	<b>0.372</b>
7	(36, 8, 7129)	<b>0.775</b>	<b>0.775</b>	<b>0.775</b>	<b>0.775</b>
8	(690, 172, 2)	0.987	0.987	0.987	<b>1.000</b>
9	(690, 172, 2)	<b>1.000</b>	<b>1.000</b>	<b>1.000</b>	<b>1.000</b>
10	(800, 200, 24)	0.481	<b>0.486</b>	0.431	0.462
11	(800, 200, 24)	<b>0.422</b>	<b>0.422</b>	0.393	0.387
12	(216, 54, 13)	0.624	0.624	0.624	<b>0.626</b>
13	(216, 54, 13)	<b>0.786</b>	<b>0.786</b>	<b>0.786</b>	<b>0.786</b>
14	(281, 70, 34)	0.798	0.770	0.770	<b>0.859</b>
15	(6500, 1624, 112)	<b>1.000</b>	<b>1.000</b>	<b>1.000</b>	<b>1.000</b>
16	(167, 41, 60)	0.806	0.806	0.806	<b>0.853</b>
17	(1605, 30956, 123)	0.544	0.544	0.544	<b>0.549</b>
18	(2265, 30296, 123)	<b>0.556</b>	<b>0.556</b>	<b>0.556</b>	0.553
19	(3185, 29376, 123)	<b>0.560</b>	<b>0.560</b>	<b>0.560</b>	0.528
20	(4781, 27780, 123)	0.525	0.525	0.525	<b>0.554</b>
21	(38, 34, 7129)	<b>0.663</b>	<b>0.663</b>	<b>0.663</b>	<b>0.663</b>
22	(2000, 600, 500)	<b>0.367</b>	<b>0.367</b>	<b>0.367</b>	<b>0.367</b>
23	(1000, 2175, 60)	0.788	0.788	0.788	<b>0.807</b>
24	(1000, 2175, 60)	0.792	0.792	0.792	<b>0.793</b>
25	(3089, 4000, 4)	-	-	-	-
26	(1243, 41, 22)	-	-	-	-
27	(2477, 47272, 300)	0.577	<b>0.578</b>	<b>0.578</b>	0.560
28	(3470, 46279, 300)	0.601	0.601	0.601	<b>0.618</b>
29	(4912, 44837, 300)	0.637	0.636	0.639	<b>0.651</b>



From the Tables 3-5, in 5 datasets we got the same results for the amount support vectors, accuracy, balanced accuracy and  $\phi$  coefficient (Ids 2, 4, 7, 15, 21). Also, although all algorithms converged to similar points, they did not performed well in Id-22 and Id-15 datasets to the values selected for  $\gamma$  and  $C$  hyper-parameters. In both cases were selected all samples in the training phase as support vectors, meanwhile having a competitive accuracy and  $\phi$  coefficient. For the Id-25 and Id-26 datasets we cannot compute the values from balanced accuracy and  $\phi$  coefficient, because all the algorithms did not predict any TN and FN in this run. This is justified by the non-balanced class proportion in the training phase.

Taking into account our draw criteria and the results from Table 6, only in 5 datasets Filter-A and Filter-B got lower metric values than LIBSVM (Ids 1, 5, 14, 16, 25 and Ids 5, 14, 16, 25, respectively) and only 3 times Filter-A got lower values than quadprog (Ids 1, 10, 25). Filter-B had no lower values than quadprog in this experiment by the draw criteria. Figure 3 illustrate the MIR differences for each dataset.



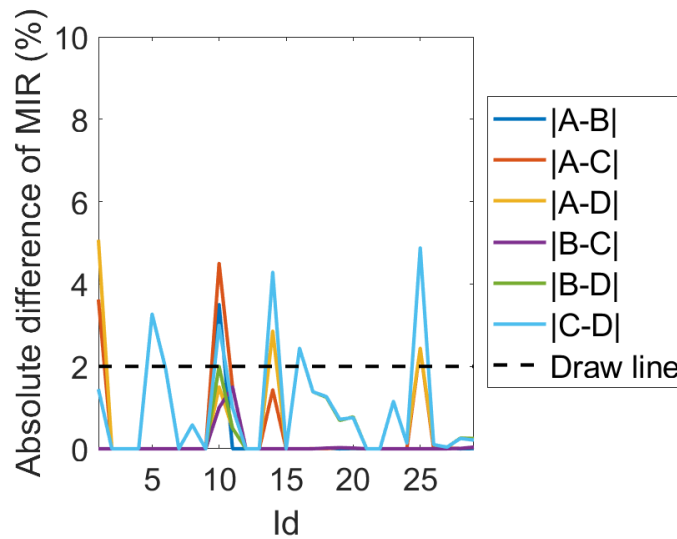


Table 6 – Misclassification instances rate percentage and differences of Filter-A, Filter-B, *quadprog* and LIBSVM with RBF kernel function. In the table, “A = Filter-A”, “B = Filter-B”, “C=*quadprog*” and “D=LIBSVM”

Id	MIR (%)				Absolute differences between methods (%)					
	A	B	C	D	A-B	A-C	A-D	B-C	B-D	C-D
1	25.362	21.739	21.739	20.290	3.623	3.623	5.072	0.000	1.449	1.449
2	13.043	13.043	13.043	13.043	0.000	0.000	0.000	0.000	0.000	0.000
3	5.147	5.147	5.147	5.147	0.000	0.000	0.000	0.000	0.000	0.000
4	25.000	25.000	25.000	25.000	0.000	0.000	0.000	0.000	0.000	0.000
5	26.144	26.144	26.144	29.412	0.000	0.000	3.268	0.000	3.268	3.268
6	26.797	26.797	26.797	24.837	0.000	0.000	1.961	0.000	1.961	1.961
7	12.500	12.500	12.500	12.500	0.000	0.000	0.000	0.000	0.000	0.000
8	0.581	0.581	0.581	0.000	0.000	0.000	0.581	0.000	0.581	0.581
9	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000
10	21.000	24.500	25.500	22.500	3.500	4.500	1.500	1.000	2.000	3.000
11	23.500	23.500	25.000	24.000	0.000	1.500	0.500	1.500	0.500	1.000
12	18.519	18.519	18.519	18.519	0.000	0.000	0.000	0.000	0.000	0.000
13	11.111	11.111	11.111	11.111	0.000	0.000	0.000	0.000	0.000	0.000
14	10.000	11.429	11.429	7.143	1.429	1.429	2.857	0.000	4.286	4.286
15	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000
16	9.756	9.756	9.756	7.317	0.000	0.000	2.439	0.000	2.439	2.439
17	16.950	16.953	16.953	15.570	0.003	0.003	1.379	0.000	1.383	1.383
18	17.095	17.081	17.098	15.830	0.013	0.003	1.264	0.017	1.251	1.267
19	17.228	17.228	17.256	16.541	0.000	0.027	0.688	0.027	0.688	0.715
20	16.220	16.224	16.206	15.450	0.004	0.014	0.770	0.018	0.774	0.756
21	17.647	17.647	17.647	17.647	0.000	0.000	0.000	0.000	0.000	0.000
22	31.667	31.667	31.667	31.667	0.000	0.000	0.000	0.000	0.000	0.000
23	10.805	10.805	10.805	9.655	0.000	0.000	1.149	0.000	1.149	1.149
24	10.483	10.483	10.483	10.345	0.000	0.000	0.138	0.000	0.138	0.138
25	1.450	1.450	1.450	1.350	2.439	2.439	2.439	0.000	4.878	4.878
26	21.951	19.512	19.512	24.390	0.000	0.000	0.100	0.000	0.100	0.100
27	2.291	2.278	2.278	2.318	0.013	0.013	0.028	0.000	0.040	0.040
28	2.146	2.146	2.137	1.886	0.000	0.009	0.259	0.009	0.259	0.251
29	2.007	2.012	1.972	1.762	0.004	0.036	0.245	0.040	0.250	0.210



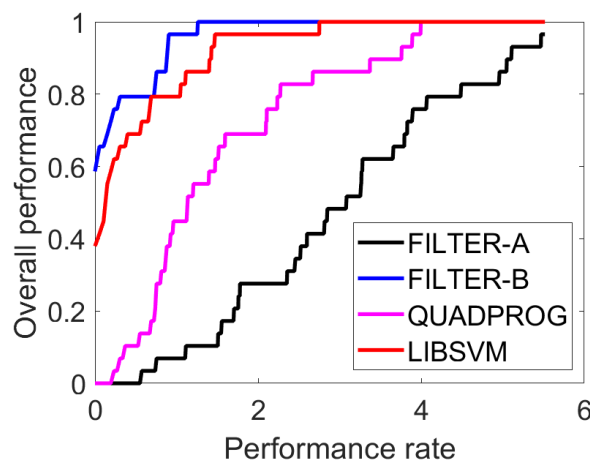
Figure 3 – MIR absolute differences for all datasets. In the figure, “A = Filter-A”, “B = Filter-B”, “C=quadprog” and “D=LIBSVM”



Source: Elaborated by the authors.

Thus, we can conclude that Filter-A and Filter-B were competitive with quadprog and LIBSVM for the metrics of classification performance. Now we will analyze the runtime for the algorithms. Table 3 contains the values of cpu-time in the test phase, while in Figure 4 we show the CPU-time performance profile, for the algorithms applied in all 29 datasets. The Filter-B algorithm solves 100% of problems, being the fastest in 58.6% of them. LIBSVM solves 100% of problems, but fastest in 37.9% of them. The worst performance, in relation of cpu-time, was *quadprog* and Filter-A, respectively.

Figure 4 – Performance profile: cpu training time



Source: Elaborated by the authors.



Figure 4 accentuate the efficiency of the proposed Filter-B. The main reason of that good performance of Filter-B in detriment of Filter-A is that Filter-B solved datasets with a sparse/low density structures and non-balanced training class proportion faster, such as lds 15, 17, 18, 19, 20, 25, 26, 27, 28, 29. Filter-A and Filter-B differs in how subproblem (14) was solved, which means that [34] was more efficient to provide better train time results than quadprog.

As shown, the Filter-SQP method is a promising alternative SVM training, offering a reliable and efficient solution for optimization. With a guarantee of convergence in its optimization, the Filter-B proves to be a solid choice for handling SVM problems. By adopting the Filter-SQP, specially the case of Filter-B, practitioners of SVM can benefit from a dependable and stable approach, avoiding common convergence pitfalls and improving training efficiency. Additionally, the Filter-B offers a high level of flexibility, adapting to different datasets and SVM configurations. Thanks to the guarantee of convergence, users can trust the consistency and accuracy of the results obtained by the Filter-SQP. This provides greater confidence in choosing this method as a reliable alternative for SVM training, regardless of the size or complexity of the problem at hand.

## 5 Conclusions

In this paper we propose a Filter algorithm applied to SVM binary classification problem, applying a SQP method for the filter step computation. The SQP method makes possible to apply any Hessian approximation for the quadratic Lagrangian approximation of SVM problem, but in the numerical experiments we choose to compute the exact Hessian of the quadratic Lagrangian approximations. Filter-A and Filter-B offers several advantages for solving the feasibility step in Filter-SQP method. Firstly, Filter-SQP is known for its robustness and efficiency in handling constraints. By incorporating a filter mechanism, it effectively manages the constraint violation and feasibility issues, ensuring that the solutions satisfy the imposed constraints. However, there are certain limitations and trade-offs associated with using the Filter-SQP. One notable disadvantage is the computational complexity, especially when dealing with large-scale and sparse optimization problems. In this paper, we used two approaches (quadprog and Augmented Lagrangian) to solve the viability step. The Augmented Lagrangian method introduces additional variables and penalty terms, which can significantly increase the computational burden. Furthermore, the reliance on quadprog as the underlying solver may restrict the applicability of the method to specific problem formulations and constraints. It is essential to consider the problem size, complexity, and available computational resources when de-



ciding to employ the Filter-A and Filter-B. In order to compare the algorithms, we have implemented them in MATLAB® and performed extensive numerical tests with 29 datasets selected from the LIBSVM collection. The algorithms that we have used to calculate the step and the filter criteria have given rise to two variants of the Filter-SQP algorithm. The prediction/test phase have showed a significant difference between these two variants in cpu-time in training phase only, with the Filter-SQP with the Augmented Lagrangian based method for the computation of feasibility step being faster. For the speed, accuracy, balanced accuracy and Matthews correlation coefficient, the Filter algorithms got competitive results compared to LIBSVM and *quadprog*. In summary, the Filter-SQP method presents itself as a highly viable and promising solution for SVM training, ensuring convergence in its optimization. Its reliability, efficiency, and ability to handle a variety of scenarios make it an attractive alternative for those seeking to maximize the performance and accuracy of their SVM models.

## References

BALDI, P. ; BRUNAK, S.; CHAUVIN, Y.; ANDERSEN, C.; A. F.; NIELSEN, H. Assessing the accuracy of prediction algorithms for classification: an overview. **Bioinformatics**, v. 16, n. 5, p. 412-424, 2000. DOI: <https://doi.org/10.1093/bioinformatics/16.5.412>.

BISHOP, C. M. **Pattern recognition and machine learning**. [S.l.]: Springer, 2006.

BOSE, B. E.; GUYON, I. M.; VAPNIK, V. N. A training algorithm for optimal margin classifiers. *In*: FIFTH ANNUAL WORKSHOP ON COMPUTATIONAL LEARNING THEORY, Pittsburgh, Pennsylvania, USA, 1992. **Proceedings** [...]. New York, NY, USA: Association for Computing Machinery, 1992, p. 144-152. DOI: <https://doi.org/10.1145/130385.130401>.

CHANG, C.-C.; LIN, C.-J. LIBSVM: A Library for Support Vector Machines. **ACM Transactions on Intelligent Systems and Technology**, v. 2, n. 3, p. 1-27, 2011. DOI: <https://doi.org/10.1145/1961189.1961199>.

CHAUHAN, V. K.; DAHIYA, K.; SHARMA, A. Problem formulations and solvers in linear SVM: a review. **Artificial Intelligence Review**, v. 52, n. 2, p. 803-855, 2019. DOI: <https://doi.org/10.1007/s10462-018-9614-6>.

CHEN, H. L.; YANG, B.; WANG, S. J.; Wang, G.; LIU, D. Y.; LI, H. Z.; LIU, W. B. Towards an optimal support vector machine classifier using a parallel particle swarm optimization strategy. **Applied Mathematics and Computation**, v. 239, p. 180-197, 2014. DOI: <https://doi.org/10.1016/j.amc.2014.04.039>.

CORTES, C.; VAPNIK, V. Support-vector networks. **Machine Learning**, v. 20, p. 273-297, 1995. DOI: <https://doi.org/10.1007/BF00994018>.



CRISTIANINI, N.; SHAWE-TAYLOR, J. **An introduction to support vector machines and other kernel-based learning methods**. [S.l.]: Cambridge University Press, 2000. DOI: <https://doi.org/10.1017/CB09780511801389>.

DENG, N.; TIAN, Y.; ZHANG, C. **Support Vector Machines**: Optimization based theory, algorithms, and extensions. [S.l.]: Chapman & Hall, CRC Press, 2012.

DOLAN, E. D; MORÉ, J. J. Benchmarking optimization software with performance profiles. **Mathematical Programming**, v. 91, p. 201-213, 2002. DOI: <https://doi.org/10.1007/s101070100263>.

FAN, R.-E.; CHANG, K.-W.; HSIEH, C.-J.; WANG, X.-R.; LIN, C.-J. Liblinear: A library for large linear classification. **Journal of Machine Learning Research**, v. 9, p. 1871-1874, 2008.

FERRIS, M. C.; MUNSON, T. S. Interior-point methods for massive support vector machines. **SIAM Journal on Optimization**, v. 13, n. 3, p. 783-804, 2002. DOI: <https://doi.org/10.1137/S1052623400374379>.

FLETCHER, R.; LEYFFER, S. Nonlinear programming without a penalty function. **Mathematical Programming**, v. 91, p. 239-269, 2002. DOI: <https://doi.org/10.1007/s101070100244>.

FROHLICH, H.; ZELL, A. Efficient parameter selection for support vector machines in classification and regression via model-based global optimization. *In*: IEEE INTERNATIONAL JOINT CONFERENCE ON NEURAL NETWORKS, Montreal, QC, Canada, 2005. **Proceedings** [...]. v. 3, 2005, p. 1431-1436. DOI: <https://doi.org/10.1109/IJCNN.2005.1556085>.

GONZAGA, C. C.; KARAS, E.; VANTI, M. A globally convergent filter method for nonlinear programming. **SIAM Journal on Optimization**, v. 14, n. 3, p. 646-669, 2004. DOI: <https://doi.org/10.1137/S1052623401399320>.

GOULD, N. I. M.; LEYFFER, S.; TOINT, P. L. A multidimensional filter algorithm for nonlinear equations and nonlinear least-squares. **SIAM Journal on Optimization**, v. 15, n. 1, p. 17-38, 2004. DOI: <https://doi.org/10.1137/S1052623403422637>.

HASTIE, T.; TIBSHIRANI, R.; FRIEDMAN, J. **The elements of statistical learning**: data mining, inference, and prediction. Springer Series in Statistics. 2. ed. [S.l.]: Springer, 2009. DOI: <https://doi.org/10.1007/978-0-387-84858-7>.

HSIEH, C.-J.; CHANG, K.-W.; LIN, C.-J.; KEERTHI, S S.; SUNDARARAJAN, S. A dual coordinate descent method for large-scale linear SVM. *In*: 25TH INTERNATIONAL CONFERENCE ON MACHINE LEARNING, Helsinki, Finland, 2008. **Proceedings** [...]. New York, NY, USA: Association for Computing Machinery, 2008, p. 408-415. DOI: <https://doi.org/10.1145/1390156.1390208>.

IZMAILOV, A. F.; SOLODOV, M. V. **Newton-type methods for optimization and variational problems**. Springer Series in Operations Research and Financial Engineering. [S.l.]: Springer, 2014. DOI: <https://doi.org/10.1007/978-3-319-04247-3>.

JOACHIMS, T. **Making large-scale SVM learning practical**. [S.l.]: [S.n.], 1998, p. 41-56.



JOACHIMS, T. Training linear SVMs in linear time. *In: 12TH ACM SIGKDD INTERNATIONAL CONFERENCE ON KNOWLEDGE DISCOVERY AND DATA MINING*, Philadelphia, PA, USA, 2006. **Proceedings** [...]. New York, NY, USA: Association for Computing Machinery, 2006, p. 217-226. DOI: <https://doi.org/10.1145/1150402.1150429>.

LIN, C.-J.; WENG, R. C.; KEERTHI, S. S. Trust region newton method for logistic regression. **Journal of Machine Learning Research**, v. 9, n. 22, p. 627-650, 2008. Available in: <https://www.jmlr.org/papers/v9/lin08b.html>. Access at: October 29, 2023.

MANGASARIAN, O. L.; MUSICANT, D. R. Lagrangian support vector machines. **Journal of Machine Learning Research**, v. 1, n. 3, p. 161-177, 2001.

MATTHEWS, B. W. Comparison of the predicted and observed secondary structure of T4 phage lysozyme. **Biochimica et Biophysica Acta (BBA) - Protein Structure**, v. 405, n. 2, p. 442-451, 1975. DOI: [https://doi.org/10.1016/0005-2795\(75\)90109-9](https://doi.org/10.1016/0005-2795(75)90109-9).

NIU, D.; WANG, C.; TANG, P.; WANG, Q.; SONG, E. A sparse semismooth Newton based augmented Lagrangian method for large-scale support vector machines. **arXiv:1910.01312**, 2019. DOI: <https://doi.org/10.48550/arXiv.1910.01312>.

NOCEDAL, J.; WRIGHT, S. **Numerical optimization**. Springer Series in Operations Research and Financial Engineering. 2. ed. [S.l.]: Springer, 2006. DOI: <https://doi.org/10.1007/978-0-387-40065-5>.

PERIÇARO, G. A.; RIBEIRO, A. A.; KARAS, E. W. Global convergence of a general filter algorithm based on an efficiency condition of the step. **Applied Mathematics and Computation**, v. 219, n. 17, p. 9581-9597, 2013. DOI: <https://doi.org/10.1016/j.amc.2013.03.012>.

PICCIALLI, V.; SCIANDRONE, M. Nonlinear optimization and support vector machines. **4OR**, v. 16, p. 111-149, 2018. DOI: <https://doi.org/10.1007/s10288-018-0378-2>.

PLATT, J. **Sequential minimal optimization**: A fast algorithm for training support vector machines. [S.l.]: Microsoft, 1998. Available in: <https://www.microsoft.com/en-us/research/publication/sequential-minimal-optimization-a-fast-algorithm-for-training-support-vector-machines>. Access at: October 29, 2023.

SCHÖLKOPF, B.; SMOLA, A. J. **Learning with kernels**: support vector machines, regularization, optimization, and beyond. [S.l.]: MIT Press, 2001.

SMOLA, A. J.; VISHWANATHAN, S. V. N.; LE, Q. V. Bundle methods for machine learning. *In: TWENTY-FIRST ANNUAL CONFERENCE ON NEURAL INFORMATION PROCESSING SYSTEMS*, Vancouver, British Columbia, Canada, 2007. **Proceedings** [...]. [S.l.]: Curran Associates, 2007, p. 1377-1384. Available in: [https://proceedings.neurips.cc/paper\\_files/paper/2007/file/26337353b7962f533d78c762373b3318-Paper.pdf](https://proceedings.neurips.cc/paper_files/paper/2007/file/26337353b7962f533d78c762373b3318-Paper.pdf). Access at: October 1, 2023.

SHALEV-SHWARTZ, S.; Singer, Y.; Srebro, N.; Cotter, A. Pegasos: primal estimated sub-gradient solver for SVM. **Mathematical Programming**, v. 127, p. 3-30, 2011. DOI: <https://doi.org/10.1007/s10107-010-0420-4>.



SHAWE-TAYLOR, J.; CRISTIANINI, N. Kernel methods for pattern analysis. [S.l.]: Cambridge University Press, 2004.

TORREALBA, E. M. R.; SILVA, J. G.; MATIOLI, L. C.; KOLOSSOSKI, O.; SANTOS, P. S. M. Augmented Lagrangian algorithms for solving the continuous nonlinear resource allocation problem. **European Journal of Operational Research**, v. 299, n. 1, p. 46-59, 2022. DOI: <https://doi.org/10.1016/j.ejor.2021.11.027>.

WOODSEND, K.; GONDZIO, J. Exploiting separability in large-scale linear support vector machine training. **Computational Optimization and Applications**, v. 49, p. 241-269, 2011. DOI: <https://doi.org/10.1007/s10589-009-9296-8>.

WOODSEND, K.; GONDZIO, J. Hybrid mpi/openmp parallel linear support vector machine training. **Journal of Machine Learning Research**, v. 10, p. 1937-1953, 2009.

YAN, Y.; LI, Q. An efficient augmented lagrangian method for support vector machine. **Optimization Methods and Software**, v. 35, n. 4, p. 855-883, 2020. DOI: <https://doi.org/10.1080/10556788.2020.1734002>.

YIN, J.; LI, Q. A semismooth newton method for support vector classification and regression. **Computational Optimization and Applications**, v. 73, p. 477-508, 2019. DOI: <https://doi.org/10.1007/s10589-019-00075-z>.

## Acknowledgment

This study was financed in part by the Coordenação de Aperfeiçoamento de Pessoal de Nível Superior (CAPES), Finance Code 001, Brazil.

