Support Vector Machine with feature selection: A multiobjective approach

Javier Alcaraz, Martine Labbé, Mercedes Landete

Expert Systems With Applications 204 (2022) 117485

Contents lists available at ScienceDirect

Expert Systems With Applications

journal homepage: www.elsevier.com/locate/eswa

Support Vector Machine with feature selection: A multiobjective approach

Javier Alcaraz[a,*], Martine Labbé[b], Mercedes Landete[a]

[a] Departamento de Estadística, Matemáticas e Informática. Instituto Centro de Investigación Operativa, Universidad Miguel Hernández de Elche, Spain
[b] Computer Science Department, Université Libre de Bruxelles, Belgium

ARTICLE INFO

Keywords:
Support vector machine
Feature selection
Multi-objective optimization
NSGA-II
AUGMECON2

ABSTRACT

Support Vector Machines are models widely used in supervised classification. The classical model minimizes a compromise between the structural risk and the empirical risk. In this paper, we consider the Support Vector Machine with feature selection and we design and implement a bi-objective evolutionary algorithm for approximating the Pareto optimal frontier of the two objectives. The metaheuristic is based on the non-dominated sorting genetic algorithm and includes problem-specific knowledge. To demonstrate the efficiency of the algorithm proposed, we have carried out extensive computational experiments comparing the Pareto-frontiers given by the exact method AUGMECON2 and the metaheuristic approach respectively in a set of well known instances. In this paper, we also discuss some properties of the points in the Pareto frontier.

1. Introduction

Supervised classification is the procedure of classifying a set of objects (also called vectors) into classes. The procedure is designed with a training objects set and the goal is to use it for classifying new objects. In supervised classification the statistical learning from data is devoted to classification. The problem of classifying objects into classes is faced in many different fields such as insurance companies (to differentiate bad from good customers), medicine (to determine whether a tumor is benign or malignant), chemistry or image data. The book by Vapnik (2013) is a comprehensive introduction to the relevance of learning theory for designing supervised classification.

1.1. The Support Vector Machine

The SVM was first introduced by Cortes and Vapnik (1995) and Vapnik (1995) as a new type of universal learning machine that implements the strategy of keeping the value of the empirical risk fixed minimizing the confidence interval. It is based on the way proposed by Vapnik and Chervonenkis (1964) to build the optimal separating hyperplane for pattern recognition, used by Vapnik and Chervonenkis (1974) in the case of separable data and later generalized by Vapnik (1995) for the case of non-separable data. The separating hyperplane is the intermediate hyperplane of two parallel hyperplanes, one letting above the vectors of the first class and the other letting below the vectors of the second class. The distance between the two parallel hyperplanes is called the margin. If data is linearly separable, this margin is said to be hard. If a linear boundary is not feasible or misclassifications are allowed in the hope of achieving better generality, this margin is said to be soft. For the soft margin SVM, good separating hyperplanes are those such that the margin between the two parallel hyperplanes is large and the distance of the misclassified vectors to the corresponding parallel hyperplane is small. The SVM has proven to be a very effective tool for supervised classification. The paper by Burges (1998) constitutes a practical tutorial on SVM.

The output of an SVM is a classification hyperplane π and the goals are twofold: the maximization of the margin between two equidistant parallel hyperplanes to π and the minimization of the number of misclassified vectors by the two equidistant parallel hyperplanes. In the standard approach, to maximize the margin, one term of the objective function to be minimized consists in the inverse of the margin that, in turn, is a quadratic and convex term. Hence, the terms in the standard objective function are not an exact translation of the problem goals but a sensible modeling. In this paper, we analyze the effect on the Pareto frontier of the translation. It is shown that the set of efficient points is nearly but not exactly the same.

Feature selection is a widely used process for reducing the number of variables. On the one hand, the selection of the most representative variables allows to reduce the size of the problem to solve, what evidently adds operationality to the problem. On the other hand, it avoids the over-adjustment and thus confers robustness to the solution.

1.2. Literature review

Several authors have proposed models that consider feature selection. In the work carried out by Maldonado et al. (2014), the authors

* Corresponding author.
E-mail addresses: jalcaraz@umh.es (J. Alcaraz), martine.labbe@ulb.be (M. Labbé), landete@umh.es (M. Landete).

https://doi.org/10.1016/j.eswa.2022.117485
Received 8 September 2021; Received in revised form 3 April 2022; Accepted 30 April 2022
Available online 11 May 2022

introduce two SVM mixed integer models with feature selection, the work of Aytug (2015) solves the SVM with feature selection with the help of benders decomposition. Recently, Gaudioso et al. (2017) propose a Lagrangian relaxation approach for the SVM problem with feature selection and the models developed by Maldonado et al. (2014) are later enhanced (Labbé et al., 2019).

Despite the high prediction rate of the SVM technique in a wide range of real applications, the classification accuracy of the SVM highly depends on specifying the model parameters as well as selecting the subset of features. Several metaheuristic have been developed with this aim. To cite only some of them, (Huang & Wang, 2006) and later (Zhao et al., 2011) propose genetic algorithms which are also compared to the grid algorithm and others using standard benchmark instances; Lin et al. (2008) propose a particle swarm optimization and demonstrate its efficiency by comparing it to grid search in different public datasets; García-Pedrajas et al. (2014) propose a memetic algorithm for dealing with many instances and many features simultaneously by performing joint instance and feature selection; Gauthama Raman et al. (2017) present an adaptive and a robust intrusion detection technique for parameter setting and feature selection in SVM; Aladeemy et al. (2017) propose a variation of Cohort Intelligence algorithm for SVM with feature selection; Bouraoui et al. (2018) propose a multi-objective approach to simultaneously optimize SVM parameters and feature subset using different kernel functions; Faris et al. (2018) present a multi-verse optimizer approach for feature selection and optimizing SVM parameters based on a robust system architecture; Tao et al. (2019) present a good classifier for data regarding hospitalization expenses which is obtained by using feature selection in SVM; Ibrahim et al. (2019) develop a novel metaheuristic, the Grasshopper Optimization Algorithm, which is inspired by grasshoppers searching for food and approved its ability to solve some biomedical datasets; Cheng et al. (2020) explore the benefit of subdividing the training set into smaller regions when training sets are large-scale; Dudzik et al. (2021) propose an evolutionary technique that efficiently classifies difficult datasets, including very large and extremely imbalanced cases; Al-Zoubi et al. (2021) present an improved evolutionary variant of competitive swarm optimizer and its superiority over a genetic algorithm is shown; All these works learn from a training set and check the performance with a testing set. Most of them select optimal features and optimize the parameters of SVM simultaneously with the aim of reducing the number of features while trying to maintain the predictive capability.

1.3. The Support Vector Machine from a multi-objective perspective: the Pareto front

Solving the SVM, a decision maker has a solution to a particular compromise between the structural and the empirical risk. Solving the SVM for different compromises, the decision maker has different alternative solutions for different scenarios. The benefit of a Pareto front is that it provides a decision maker with a comprehensive set of alternative solutions from which a better decision can definitely be made. The Pareto front of any bi-objective optimization problem gives a lot of relevant information to the decision maker, who can achieve better results. The problem of obtaining the Pareto front for the SVM with feature selection has not been solved by any of the existing previous works within the field of SVM. All the previous multi-objective exact or genetic algorithms with or without feature selection focus on achieving the best classification results for a set of instances but do not show the Pareto front to the decision maker. An efficient tool for obtaining the Pareto front of the SVM with feature selection was something important that remained to be done. In this paper, we give several important properties of the Pareto front of the soft margin SVM with feature selection and we design an efficient metaheuristic tool for obtaining it. We do not split information into training sets and testing sets because we do not give a single classifier. Our two objective functions are the structural risk and the empirical risk, so we do not need to tune SVM parameters and we move the decision to the decision maker.

From the point of view of the model, the selection of features is carried out by introducing binary variables to the model that indicate which features are selected. Thus, the SVM model with feature selection becomes a mixed-integer model and therefore a non-convex model. Since the SVM problem without feature selection is a linear programming problem and thus a convex problem, its Pareto frontier can be obtained by varying the parameter C. However, since the SVM problem with feature selection is not convex, obtaining its Pareto frontier requires other techniques. The drawbacks of scalarization for non-convex problems as well as recent results on non-convex multi-objective optimization problems and methods can be consulted in the book by Pardalos et al. (2017). There are several methods that are capable of generating a set of well-distributed Pareto solutions on convex and non-convex frontiers. AUGMECON2 is one of these methods, which is precisely that used in this paper.

In this paper we present a metaheuristic approach to approximate the Pareto-optimal frontier simultaneously considering the two SVM objectives and making the selection of features. As far as we know this type of metaheuristic has not been proposed before. The main advantage of the metaheuristic we propose compared with the ones in the literature, is that our approach gives a good description of the Pareto front of the SVM with feature selection whereas the others focus on optimizing the subset of features and parameters, and then efficiently classify some benchmark datasets.

This paper is organized as follows, Section 2 presents the SVM mathematical model and its connections with the related hyperplane distances. In Section 3, some properties concerning the concept of efficiency and the objectives considered are established. The exact approach employed to solve the SVM with feature selection is presented in Section 4 and, in Section 5, the metaheuristic algorithm, based on the non-dominated sorting genetic algorithm is described in detail. Finally, the computational experiment to evaluate the efficiency of our approach is presented and discussed in Section 6.

2. Support Vector Machine model

Consider a training set Ω of vectors partitioned into two classes. Vectors are represented by a pair $(x_i, y_i) \in \mathbb{R}^n \times \{-1, 1\}$, where n is the number of features observed for each vector, x_i contains the feature values for vector i and y_i indicates to which of the two classes of Ω vector i belongs. If Ω is linearly separable, there exist $v \in \mathbb{R}^n, \theta \in \mathbb{R}$, and $\mu \in \mathbb{R}_0^+$ such that all vectors in the class for which $y_i = 1$ satisfy $v^T x_i \leq \theta - \mu$ and all vectors in the class for which $y_i = -1$ satisfy $v^T x_i \geq \theta + \mu$. W.l.o.g. dividing by μ, the Support Vector Machine Problem consists in determining the hyperplane $f(x) = w^T \cdot x + b$ that *optimally* separates the vectors in the training set. Optimality is twofold, one wishes to maximize the distance between two parallel hyperplanes supporting some vectors of the two classes and to minimize the sum of classification errors. The classical hard margin SVM model (Bradley & Mangasarian, 1998) minimizes a compromise between the above two objectives called respectively the structural risk and the empirical risk.

$$\min_{w,b,\xi} \quad \frac{1}{2}\|w\|^2 + C\sum_{i=1}^{m}\xi_i \tag{1}$$

$$\text{s.t.} \quad y_i(w^T x_i + b) \geq 1 - \xi_i \qquad i = 1, \dots, m, \tag{2}$$

$$\xi_i \geq 0 \qquad i = 1, \dots, m, \tag{3}$$

The n-dimensional vector w contains variables w_j that, as well as b, take values in $\mathbb{R}$ and represent the coefficients of the two parallel hyperplanes $w^T x + b = 1$ and $w^T x + b = -1$. The first term $\frac{1}{2}\|w\|^2$ of the objective function (1) represents the structural risk since $\|w\|$ is twice the inverse of the distance between these two hyperplanes. The second term $C\sum_{i=1}^{m}\xi_i$ is the empirical risk given by the sum of the deviation of misclassified objects multiplied by C which is a parameter that regulates the trade-off between the two objectives. Parameter

C establishes how important it is to avoid missclassification in the training data. Constraints (2) and (3) ensure that either vectors i in the class represented by $y_i = 1$ satisfy $(w^T x_i + b) \geq 1$ and vectors in class $y_i = -1$ satisfy $(w^T x_i + b) \leq -1$ or constraint (2) is violated by a positive amount ξ_i, denoted deviation. The slack variables ξ_i make the difference with the soft margin SVM model.

In the following we consider the two objectives $O_1 = \frac{1}{2}\|w\|^2$ and $O_2 = \sum_{i=1}^{m} \xi_i$ separately.

2.1. SVM objective function and hyperplane distances

As stated before the goals in SVM are the maximization of the distance between two parallel hyperplanes supporting some vectors of the two classes and the minimization of the sum of classification errors. It is easy to check that the objective values O_1 and O_2 allow us to exactly compute these two goal values, i.e., the distance between the two parallel hyperplanes defined by the variables w and b and the sum of the distance of the misclassified vectors to the corresponding hyperplane.

Let (w, b, ξ) a feasible solution to the SVM model. Then, $\pi_1 \equiv w^T x + b = 1$ and $\pi_2 \equiv w^T x + b = -1$ are the two parallel hyperplanes and the distance between them is

$$d(\pi_1, \pi_2) = \frac{2}{\|w\|} = \frac{2}{\sqrt{2O_1}}. \tag{4}$$

The sum of the distance of the misclassified vectors is the sum of the distance of the misclassified vectors of class "1" to the hyperplane π_1 plus the sum of the distance of the misclassified vectors of class "−1" to the hyperplane π_2. If $\xi_i = max\{0, 1 - y_i(w^T x_i + b)\}$, this sum is the following:

$$\sum_{i:\xi_i>0, y_i=1} d(x_i, \pi_1) + \sum_{i:\xi_i>0, y_i=-1} d(x_i, \pi_2) = \sum_{i:\xi_i>0, y_i=1} \frac{\xi_i}{\|w\|} + \sum_{i:\xi_i>0, y_i=-1} \frac{\xi_i}{\|w\|} = \frac{O_2}{\sqrt{2O_1}}. \tag{5}$$

Summarizing, from any feasible solution of the SVM model (w, b, ξ) with objective values O_1 and O_2, the distance between the two parallel hyperplanes defined by w and b and the sum of the distances between any misclassified vector and its hyperplane can be explicitly computed.

Example 1. Let $\Omega = \{1, 2, 3, 4, 5, 6\}$ be the set of vectors in Fig. 1. Let us assume that vectors of class -1 are presented in gray and the vectors of class 1 appear in black. Table 1 gives three feasible solutions to the SVM model. These feasible solutions are the intermediate hyperplanes. Fig. 1 illustrates the intermediate hyperplanes (non-dashed lines) as well as the parallel hyperplanes (dashed-lines) for the three solutions. The intermediate hyperplane is always defined by $w_1 x + w_2 y + b = 0$ while the parallel hyperplanes are $\pi_1 : w_1 x + w_2 y + b = -1$ and $\pi_2 : w_1 x + w_2 y + b = 1$. Table 2 gives the associated objective values as well as the distances between the two parallel hyperplanes defined by the feasible solutions and the distances between each misclassified vector and its hyperplane: the distance between the two parallel hyperplanes is 4.0931 in the first SVM solution, 1 in the second SVM solution and 2.6833 in the third SVM solution. In Table 1, the ξ-variables with a positive value indicate misclassified vectors: vector 2 is misclassified in the first case, all the vectors are well classified in the second case and vectors 1 and 6 are misclassified in the third case. In the latter calculation, vectors 1 and 2 are misclassified or not with respect to hyperplane π_1 while vectors 3, 4, 5, and 6 are misclassified or not with respect to the hyperplane π_2. "-" values in distance columns of Table 2 indicate that the vector is well classified. O_1 in Table 2 is $0.5(w_1^2 + w_2^2)$, where w_1 and w_2 are in Table 1 and O_2 in Table 2 is $\xi_1 + \xi_2 + \xi_3 + \xi_4 + \xi_5 + \xi_6$, where ξ_i are in Table 1. Values in Table 2 illustrate that $d(\pi_1, \pi_2) = 2/\sqrt{2O_1}$ and $d(1, \pi_1) + d(2, \pi_1) + d(3, \pi_2) + d(4, \pi_2) + d(5, \pi_2) + d(6, \pi_2) = O_2/\sqrt{2O_1}$.

Table 1
Feasible solutions to the SVM model for the example in Fig. 1.

Fig.	w_1	w_2	b	ξ_1	ξ_2	ξ_3	ξ_4	ξ_5	ξ_6
Fig. 1(a)	0.0606	0.4848	−1.5455	0	1.0303	0	0	0	0
Fig. 1(b)	0	2	−7	0	0	0	0	0	0
Fig. 1(c)	0.3333	−0.6667	0.3333	1	0	0	0	0	3.3333

3. SVM efficiency and distance efficiency

We define two different types of efficient points depending on the goal considered.

Definition 1. A point (w, b, ξ) is **SVM efficient** iff there exists no other point (w', b', ξ') satisfying (2) and (3) such that

- $O_1(w', b', \xi') \leq O_1(w, b, \xi)$, and
- $O_2(w', b', \xi') \leq O_2(w, b, \xi)$

with a least one strict inequality.

Definition 2. A point (w, b) is **distance efficient** iff there exists no other point (w', b') such that

- $d(\pi_1', \pi_2') \geq d(\pi_1, \pi_2)$, and
- $\sum_{i:\theta_i'>0, y_i=1} d(x_i', \pi_1) + \sum_{i:\theta_i'>0, y_i=-1} d(x_i', \pi_2) \leq \sum_{i:\theta_i>0, y_i=1} d(x_i, \pi_1) + \sum_{i:\theta_i>0, y_i=-1} d(x_i, \pi_2)$.

with a least one strict inequality, where $\theta_i = max\{0, 1 - y_i(w^T x_i + b)\}$ and $\theta_i' = max\{0, 1 - y_i(w'^T x_i + b')\}$.

Intuitively, in an efficient solution, one of the parallel hyperplanes should hold on at least one vector x_{i_1} with $y_{i_1} = 1$ and the other hold on at least one vector x_{i_2} with $y_{i_2} = -1$, i.e., that each of them holds a vector of a different class. These vectors lying on one of the hyperplanes are called the support vectors. However, Proposition 3 shows that there exist SVM efficient solutions such that both parallel hyperplanes lean on vectors of the same class.

It is generally agreed that the number of support vectors in practice is very large. For instance, Geebelen et al. (2012) point out that the SVM run time complexity can be considerably higher compared to other methods, because of the large number of support vectors, which constitutes an important drawback and Cuong and Thien (2016) argue that in the SVM technique the number of support vectors obtained from the training phase is usually large, and slows down the classification phase. Many methods are proposed to treat this issue with the target of reducing the number of support vectors but without loss of solution quality.

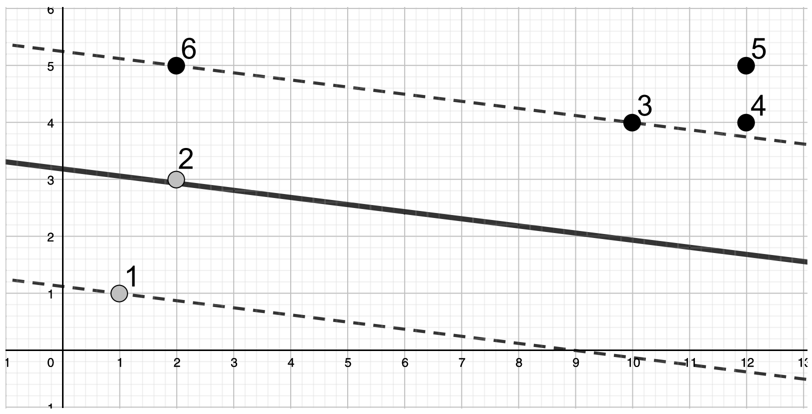
Intuitively, the set of SVM efficient solutions and the set of parallel hyperplanes that lean on vectors of different classes and that are distance efficient, henceforth distance efficient hyperplanes, should coincide. However, it is not the case. There are solutions that are SVM efficient but not define distance efficient hyperplanes and conversely. Proposition 1 states that SVM efficient solutions (w, b, ξ) with $w = 0$ do not define distance efficient hyperplanes at distance infinity. Proposition 2 states that distance efficient solutions with *large* values for the sum distances of misclassified vectors are not SVM efficient.

Let $S_{SVM} = \{(w, b, \xi) : (2), (3)\}$ be the set of feasible SVM solutions.

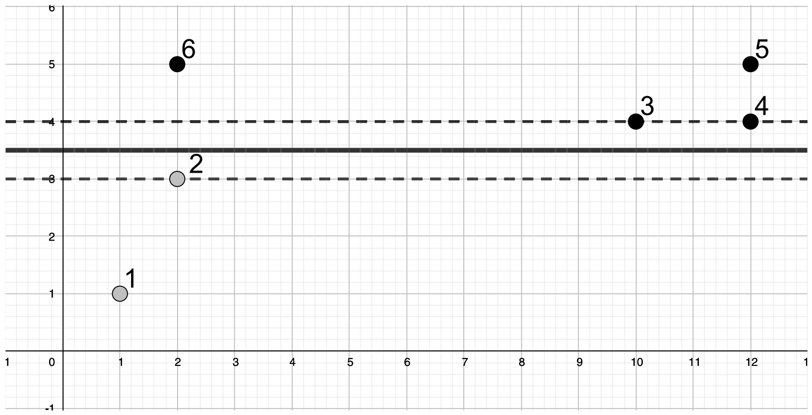
Proposition 1. *Some SVM efficient points are not distance efficient points.*

Proof. The point $(w, b, \xi) = (0, 0, 1)$ belongs to S_{SVM} and is SVM efficient since $O_1(w, b, \xi) \geq 0$ for all points $(w, b, \xi) \in S_{SVM}$. However, there does not exist a corresponding pair of hyperplanes at infinite distance. □

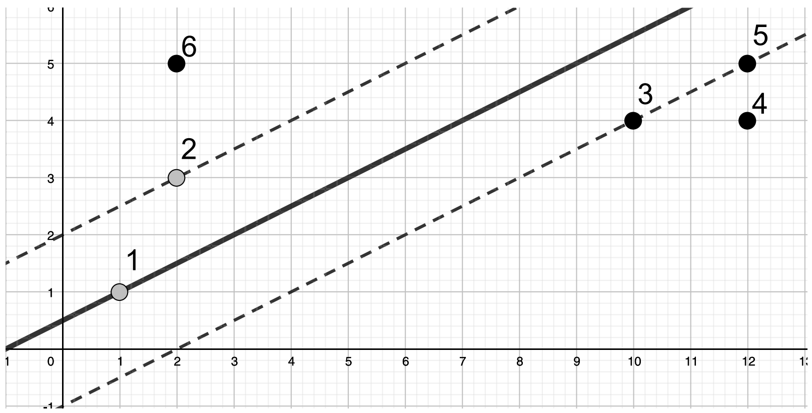
Proposition 2. *Some distance efficient points are not SVM efficient points.*



(a) π_1 supports vector 1, π_2 supports vectors 3 and 6



(b) π_1 supports vector 2, π_2 supports vectors 3 and 4



(c) π_1 supports vector 2, π_2 supports vectors 3 and 5

Fig. 1. Example with $m = 6$ and $n = 2$. Gray color indicates class "−1", black color indicates class "1".

Table 2
Distances and objective values for the example in Fig. 1.

Fig.	$d(\pi_1, \pi_2)$	$d(1, \pi_1)$	$d(2, \pi_1)$	$d(3, \pi_2)$	$d(4, \pi_2)$	$d(5, \pi_2)$	$d(6, \pi_2)$	O_1	O_2
Fig. 1(a)	4.0931	–	2.1086	–	–	–	–	0.1194	1.0303
Fig. 1(b)	1	–	–	–	–	–	–	2	0
Fig. 1(c)	2.6833	1.3416	–	–	–	–	4.4721	0.2778	4.3333

Table 3
SVM efficient solutions.

O_1	O_2	$d(\pi_1,\pi_2)$	$\sum_{i:max\{0,1-y_i(w^T x_i+b)\}>0,y_i=1} d(x_i,\pi_1)+$ $\sum_{i:max\{0,1-y_i(w^T x_i+b)\}>0,y_i=-1} d(x_i,\pi_2)$
a_1	b_1	$2/\sqrt{2a_1}$	$b_1/\sqrt{2a_1}$
a_2	b_2	$2/\sqrt{2a_2}$	$b_2/\sqrt{2a_2}$
…			
a_{t-1}	b_{t-1}	$2/\sqrt{2a_{t-1}}$	$b_{t-1}/\sqrt{2a_{t-1}}$
a_t	b_t	–	–

Proof. Let Table 3 be the set of efficient SVM efficient points, where $a_1 > a_2 > \cdots a_t$ and $b_1 < b_2 < \cdots b_t$. Let (w,b,ξ) be a point in S_{SVM} such that $O_2(w,b,\xi) = B > b_t$. Such a point exists since given a point $(w^1,b^1,\xi^1) \in S_{SVM}$, any point (w^1,b^1,ξ^2) with $\xi_i^2 > \xi_i^1$ for all i belongs to S_{SVM}. If $O_1(w,b,\xi) > a_1 B^2/b_1^2$, then (w,b,ξ) is not SVM efficient since the point that gives (a_t,b_t) dominates it but it is distance efficient since $2/\sqrt{2O_1} < 2/\sqrt{a_1}$ and $B/\sqrt{2O_1} < b_1/\sqrt{2a_1}$. □

The next proposition states that it might happen that both efficiency criteria, distance and SVM, yield to different Pareto frontiers, however, it follows from the proof that this is only the case at the border of the Pareto frontier. In general, both criteria agree on the classification.

Remark 1. The SVM Pareto frontier and the distance Pareto frontier coincide for intermediate points.

The following proposition states that for the SVM efficient points with $w = 0$ both parallel hyperplanes lean on vectors of the same class.

Proposition 3. *SVM efficient points with $w = 0$ have all the support vectors in the same class.*

Proof. Let $(0,b,\xi)$ an SVM efficient point. (x_i,y_i) is a support vector of class "1" iff $y_i(w^T x_i + b) = 1$ and $y_i = 1$. It is a support vector of class "−1" iff $y_i(w^T x_i + b) = 1$ and $y_i = -1$. Let (x_{i_1},y_{i_1}) be a support vector of class "1" and (x_{i_2},y_{i_2}) be a support vector of class "−1". From the proof of Proposition 4 it is known that $(0,b,\xi)$ satisfies $b = \max_{y_i=1}(1-\xi_i) = \min_{y_i=-1}(\xi_i - 1)$, but $\max_{y_i=1}(1-\xi_i) = 1-\xi_{i_1} = 1$ and $\min_{y_i=-1}(\xi_i - 1) = \xi_{i_2} - 1 = -1$, what is absurd. □

Example 2 (*Cont. Example 1.*)**.** The three feasible solutions in Table 1 correspond with parallel hyperplanes leaning on support vectors of the two classes. The first row in Table 1 gives one parallel hyperplane leaning on vectors 3 and 6 and the other leaning on vector 1. The second row gives one parallel hyperplane leaning on vectors 3 and 4 and the other leaning on vector 2. The third row gives one parallel hyperplane leaning on vectors 3 and 5 and the other leaning on vector 2. The second row gives an SVM and distance efficient solution.

4. AUGMECON2 for the SVM with feature selection

4.1. AUGMECON2

The exact Pareto set in multi-objective integer programming can be achieved by using the generation method AUGMECON (Mavrotas, 2009). AUGMECON2 (Mavrotas & Florios, 2013) is an improvement of the previous method AUGMECON specifically developed for more than two objectives. Both methods, AUGMECON and AUGMECON2, coincide when the number of objectives is two.

Assume that $f_1(x)$ and $f_2(x)$ are the two objective functions to be maximized, S is the feasible region for the multi-objective integer programming and $r_2 = UB_2 - LB_2$ is the range of $f_2(x)$.

The range r_2 is divided into q equal intervals using $q-1$ intermediate equidistant grid points. Thus, $q+1$ grid points are used to vary parametrically the RHS (e_2) of $f_2(x) : e_2 = LB_2 + \ell r_2/q$ for $\ell = 0,\ldots,q$. In the AUGMECON methods the problem iteratively solved is the following:

$$\begin{aligned} \max \quad & f_1(x) + \epsilon s_2/r_2 \\ \text{s.t.} \quad & f_2(x) - s_2 = e_2 \qquad (6) \\ & x \in S, s_2 \in \mathbb{R}^+ \end{aligned}$$

where ϵ is an adequately small number (usually between 10^{-3} and 10^{-6}) and s_2 is the surplus variable of constraint (6).

In each iteration the surplus variable s_2 is used for calculating the bypass coefficient

$$b = int(qs_2/r_2).$$

$int()$ is the function that returns the integer part of a real number. If $b > r_2/q$, then in the next iteration the same solution will be obtained with the only difference given by the surplus variable. The bypass coefficient b actually indicates how many consecutive iterations we can bypass.

The AUGMECON2 method uses the lexicographic optimization for computing the range r_2. In general terms, the lexicographic optimization of a series of objective functions consists in optimizing the first objective function and then among the possible alternative optima optimize the second objective function and so on.

4.2. Feature selection

Feature selection is a common problem in classification problems. On the one hand, obtaining information on each feature involves a cost and, on the other, it is not true that by increasing the number of features, better results are obtained. The results may not improve in terms of distance between the hyperplanes or in terms of magnitude of misclassified vectors. Moreover, the higher the number of characteristics, the worse the interpretability of the results will be.

In this paper we focus on the SVM model with feature selection with a budget constraint. Thus, we add the feature selection to the model and we check that all the results in Section 3 also apply when feature selection is considered. Moreover, Eqs. (4) and (5) remain valid if one computes the distances with the selected features for the SVM.

Regarding the model, it entails the addition of the following constraints.

$$\sum_j t_j = p \qquad (7)$$

$$|w_j| \le M t_j \qquad j = 1,\ldots,n \qquad (8)$$

$$t_j \in \{0,1\} \qquad j = 1,\ldots,n \qquad (9)$$

Inequality (7) states that the budget allows to have p features at most. Inequalities (8) are replaced by $-Mw_j \le t_j$ and $t_j \le Mw_j$ for $j = 1,\ldots,n$. The new feasible region is $S = \{(w,b,\xi,t)$: (2), (3), (7), (8), (9)$\}$.

Remark 2. Propositions 1–3 remain valid when feature selection is taken into account. In the proof of Proposition 1 it is enough to observe that $(w,b,\xi,t) = (0,0,1,t)$ belongs to S for any $t \in \{0,1\}^n$ with $\sum_j t_j = p$. In the proof of Proposition 2 the point $(w,b,\xi) \in S_{SVM}$ such that $\sum_i \xi_i = B > b_t$ can be replaced by a point $(w,b,\xi,t) \in S$ such that $\sum_i \xi_i = B > b_t$ and the points (w^1,b^1,ξ^1) and (w^1,b^1,ξ^2) by (w^1,b^1,ξ^1,t) and (w^1,b^1,ξ^2,t) respectively, all with the same binary vector with p positive items, t. Analogously for the proof of Proposition 3.

When feature selection is taken into account, the first objective function for the AUGMECON2 method is $-O_1$, the second is $-O_2$. Thus, the problem iteratively solved by AUGMECON2 is the following,

$$\begin{aligned} \min \quad & O_1(x) - \epsilon s_2/r_2 \qquad (10) \\ \text{s.t.} \quad & O_2(x) + s_2 = e_2 \qquad (11) \\ & x \in S, s_2 \in \mathbb{R}^+ \qquad (12) \end{aligned}$$

The lexicographic upper bound UB_2 on O_2 is defined as

$$UB_2 = \min\{O_2(w,b,\xi,t) : (w,b,\xi,t) \in S, O_1(w,b,\xi,t) = O_1^*\}$$

where

$$O_1^* = \min\{O_1(w',b',\xi',t') : (w',b',\xi',t') \in S\}$$

Clearly, UB_2 is an upper bound on the value of O_2 for any SVM efficient point. Further, it is easy to determine as shown in the following proposition.

Proposition 4. *Let* $m_1 = \sum_{i:y_i=1} 1$ *and* $m_{-1} = \sum_{i:y_i=-1} 1$. *Then,*

$$UB_2 = \min\{O_2 : (w,b,\xi,t) \in S, w = 0\} = 2\min\{m_1, m_{-1}\}$$

Proof. First, $\min\{O_1 : (w,b,\xi,t) \in S\} = 0$. This minimum is achieved for $(w,b,\xi,t) = (0,0,1,t)$ where t is any binary vector with p positive items. Then, UB_2 is the optimal value of the problem (P) $\min\{O_2 : (w,b,\xi,t) \in S, w = 0\}$ ($O_1 = 0$ is equivalent to $w = 0$). Replacing w by 0 in (2) and (3) yields

$$\begin{aligned} y_i b &\geq 1 - \xi_i & i &= 1,\dots,m \\ \xi_i &\geq 0 & i &= 1,\dots,m \end{aligned}$$

or equivalently

$$\begin{aligned} \xi_i &\geq \max\{0, 1-b\} & \text{if } y_i &= 1 \\ \xi_i &\geq \max\{0, 1+b\} & \text{if } y_i &= -1. \end{aligned}$$

Thus, in an optimal solution to (P), $\xi_i = \max\{0, 1-b\}$ for all i such that $y_i = 1$ and $\xi_i = \max\{0, 1+b\}$ for all i such that $y_i = -1$, which implies that

$$UB_2 = m_1 \max\{0, 1-b\} + m_{-1}\max\{0, 1+b\}.$$

The RHS of the last equation is a piecewise linear function of b that attains its minimum at $b = -1$ if $m_1 \leq m_{-1}$ and at $b = 1$ otherwise, yielding the desired value of UB_2.

Remark 3. The lexicographic lower bound on O_2 is

$$LB_2 = \min\{O_2 : (w,b,\xi,t) \in S\}.$$

5. A metaheuristic approach for the SVM with feature selection

Multi-objective evolutionary algorithms (MOEAs) can be used to solve difficult NP-hard optimization problems when several objectives are considered. They become a good alternative to exact techniques when these require an excessive computational effort to solve large or even small instances. The Non-dominated Sorting Genetic Algorithm, NSGA-II (Deb et al., 2002) is one of mostly used MOEAs and has been successfully applied to find a diverse set of solutions and to converge near the true Pareto-optimal set in very different types of optimization problems with more than one objective. NSGA-II is an improvement of its predecessor NSGA (Srinivas & Deb, 1995), which was one of the first MOEAs. NSGA-II is based on the non-dominance concepts. In order to compare solutions in a population, these are sorted in different fronts F_i. In F_1 we have the solutions which are not dominated by any other solution in the population. F_2 is formed with the solutions which are dominated by one or more solutions from F_1 and so on. Solutions in front F_i are better than solutions in front F_j if $i < j$. The comparison between solutions in the same front is carried out through a distance function. This function represents the distance in the objective space of a given solution to the rest of the individuals and therefore the best solutions in the front will be those with a larger distance. This permits to guide the process toward a uniformly spread-out Pareto-front. The algorithm includes a procedure that, given a current population of solutions, creates a new population by applying the genetic operators, such as crossover and mutation. This procedure is not defined in the general template of NSGA-II and needs to be carefully designed, along with the encoding of the solutions and other procedures in order to apply this algorithm to solve a given optimization problem. In the next subsections, we present the algorithm based on NSGA-II that we propose to solve the SVM with feature selection and describe the main structures and procedures that are included in it.

5.1. Main loop

The multi-objective metaheuristic algorithm we have designed follows the general idea of NSGA-II, and the steps are shown in the flowchart presented in Fig. 2. After creating the initial population it is evaluated, i.e., each objective is computed for every solution in the population. The two objectives considered are: the maximization of distance between the hyperplanes supporting vectors of each class (O_1') and the minimization of the sum of the missclassification errors (O_2'). These objectives correspond to columns 3 and 4 of Table 3. Then, the fast non-dominated sorting is applied over the population. It groups the solutions in the different fronts of dominance and calculates, for each solution some parameters related to dominance which will be used in the selection of parents to undergo the crossover operation. Then, the iterative process starts and it will finish when a given criterion is satisfied. In our case we have two different stopping criteria, computation time and number of iterations. It is useful to have the possibility of selecting different criteria in order to allow an appropriate comparison with other techniques. Each iteration consists first of creating a new population, Q_t from the current population P_t by applying different mechanisms. Therefore, the following steps will be repeated N times, creating one new solution in Q_t each time. First, we select a couple of solutions, a mother and a father to undergo the crossover operation. This selection of parents is carried out by the selection mechanism through a tournament based on dominance parameters. Then, mother and father undergo the crossover to determine a new solution which occupies the corresponding place in Q_t. After that, the mutation mechanism alters, with a given probability of P_MUT the information describing the solution to introduce variability in the population or even to recover good characteristics which could be lost during the evolution process. Once the new population has been built and evaluated, a double sized population R_t is formed with all the solutions in P_t and Q_t. In order to reduce the size of the population to N we will choose the best individuals in R_t. Therefore, to make that selection it is necessary to apply the non-dominated sorting to R_t. The solutions in the best fronts will be copied first. If there is not free space for all the solutions in a given front, we will select those individuals with a larger distance to the rest. Therefore, when that happens the i_distance_assignment procedure will be applied to the individuals in that front to calculate this distance. The distance of a solution j, to the rest of solutions of a front, F_i is given by:

$$d_j(F_i) = \frac{O_1'(j+1) - O_1'(j-1)}{max_1(F_i) - min_1(F_i)} + \frac{O_2'(j+1) - O_2'(j-1)}{max_2(F_i) - min_2(F_i)}$$

where O_1' and O_2' represent the two objectives, $O_k'(j)$ the value of objective k in solution j, max_1 and max_2 are the maximum values of both objectives given by the solutions in F_i and min_1 and min_2 are the minimum values.

This distance is used to determine, between two solutions in the same front, which to choose to pass to the next generation. As it is interesting to obtain sets of solutions covering most of the Pareto-optimal frontier, the distance of one solution in a front provides the degree of proximity to other solutions of the set around it.

When the stopping criterion is satisfied, the evolution process finishes and the result of the algorithm will be the solutions belonging to the first front, i.e. the non-dominated solutions of the final population. This procedure is represented in pseudocode in Algorithm 1.

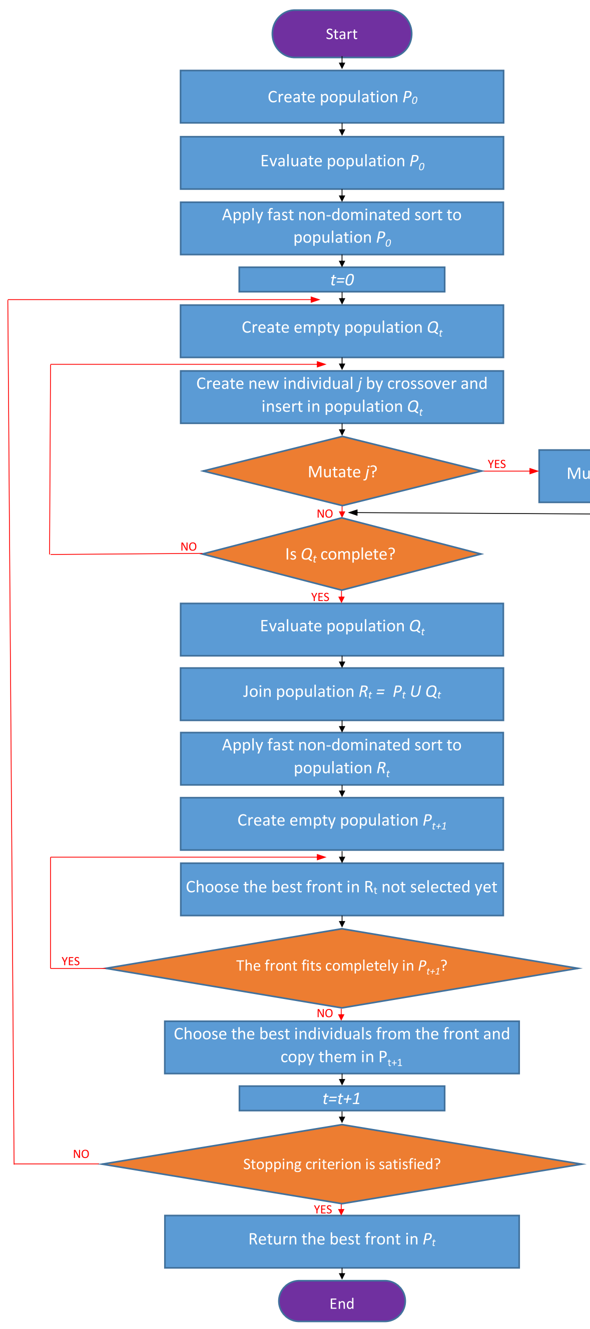

Start
Create population P_0
Evaluate population P_0
Apply fast non-dominated sort to population P_0
$t=0$
Create empty population Q_t
Create new individual j by crossover and insert in population Q_t
Mutate j?
YES
Mutation of individual j
NO
Is Q_t complete?
NO
YES
Evaluate population Q_t
Join population $R_t = P_t \cup Q_t$
Apply fast non-dominated sort to population R_t
Create empty population P_{t+1}
Choose the best front in R_t not selected yet
The front fits completely in P_{t+1}?
YES
NO
Choose the best individuals from the front and copy them in P_{t+1}
$t=t+1$
Stopping criterion is satisfied?
NO
YES
Return the best front in P_t
End

Fig. 2. Main steps of NSGA-II-SVM-f.

In the following subsections, we will detail the main features of our algorithm, such as the encoding of the solutions, the way to generate the initial population, the mechanism to select the parents to undergo the crossover, the crossover technique and the process to carry out the mutation.

5.2. Encoding

A feasible solution for the SVM with feature selection, where p is the number of features to select, can be built selecting $p+1$ vectors in Ω, p of them in class "-1" and the other in class "1" or viceversa. In that way, the two parallel hyperplanes can be defined. One of the hyperplanes supports p vectors of a given class, and the other hyperplane is built supporting 1 vector in the other class and parallel to the other hyperplane. Hence, the search space is formed by all the pairs of hyperplanes built in this way. Although the number of support vectors chosen to build the hyperplanes are 1 and p, once constructed the hyperplanes, many more vectors could support each one of the hyperplanes. Therefore, the two objectives O_1' and O_2' can be computed. The first objective is to be maximized and the second to be minimized. In that sense, the encoding proposed to represent the solutions to the SVM with feature selection has three components: (i) mode: two possible values. Mode = A indicates that we have p vectors in Ω in class "-1" and one in class "1"; Mode = B indicates that hyperplanes are built with one vector in class "-1" and p vectors in class "1". (ii) vectors: an array with the indices of the $p+1$ vectors used to build the two hyperplanes. The first position/s in the array will be occupied with the vector/s in class "-1" and the following position/s with the vector/s in class "1". We can consider, w.l.o.g. that all the vectors in

Algorithm 1: NSGA-II-SVM-f

```
1  $P_0$ = create_initial_population($N$);
2  $t = 0$;
3  $P_0$ = evaluate_population($P_0$);
4  $F$ = fast_non_dominated_sort($P_0$);
5  while not stopping_criterion do
6      for $j = 1$ to $N$ do
7          (mother, father)=tournament_dominance_selection($P_t$);
8          $Q_t[j]$=crossover(mother, father);
9          if random() $\leq PMUT$ then
10             $Q_t[j]$=mutation($Q_t[j]$);
11     $Q_t$ = evaluate_population($Q_t$);
12     $R_t = P_t \cup Q_t$;
13     $F$ = fast_non_dominated_sort($R_t$);
14     $P_{t+1} = \emptyset$;
15     $i = 1$;
16     while $|P_{t+1}| + |F_i| \leq N$ do
17         $P_{t+1} = P_{t+1} \cup F_i$;
18         $i = i + 1$;
19     if $|P_{t+1}| < N$ then
20         i_distance_assignment($F_i$) ;
21         sort($F_i$, i_distance);
22         $P_{t+1} = P_{t+1} \cup F_i[1 : (N - |P_{t+1}|)]$;
23     $t = t + 1$;
24 return $F_1$
```

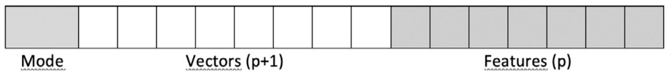


Fig. 3. Encoding for a solution with selection of 7 features.

Ω are numbered from 1 to m. (iii) features: an array with the indices of the p features selected in the solution. We can consider, w.l.o.g. that the features are numbered from 1 to n. The encoding we propose for the selected features consists of a non ordered list of a subset of the complete set of features, and this type of encoding is similar to the one proposed in Alcaraz et al. (2020) to encode a list of open facilities. Therefore, the proposed encoding has a fixed length of $2p+2$ positions: 1 mode, $p+1$ vectors and p features. Fig. 3 shows the encoding proposed for a given database with a selection of 7 features. Therefore, the total length of the chromosome is 16, the first position indicates the mode, the following 8 indicate the vectors to build the hyperplanes and the last 7 the features selected among the n available. If the mode equals A, the positions 1th to 7th in the vector array correspond to vectors in class "−1" and position 8th to the only vector in class "1". By contrast, if mode indicates B, the first position is occupied by a vector in class "−1", and positions 2th to 8th, by the seven vectors in class "1".

At this point, we would like to remark that although this encoding could exclude some feasible solutions to the problem, those in which the number of support vectors for one of the hyperplanes is lower than the number of features, we think that this is not a drawback in practice given that, as we discussed in Section 3, the number of support vectors in the solutions to the problem is usually very large.

Example 3. Fig. 4 shows two different solutions for the data given in Example 1, where $m = 6$, $n = p = 2$. When $n = p$ there is no feature selection (all the features are present) therefore, all the features will always appear in the features array. As we can observe, if $p = 2$ we have three vectors in the array, two of one class and one of the other. In the example, the first solution has mode = A, therefore the first two vectors are in class "−1" and the third in class "1". On the other hand, the second solution, with mode = B, uses the first vector in class "−1" and the others in class "1" to build the parallel hyperplanes and it corresponds with the solution presented in Fig. 1(a) and Tables 1 and 2.

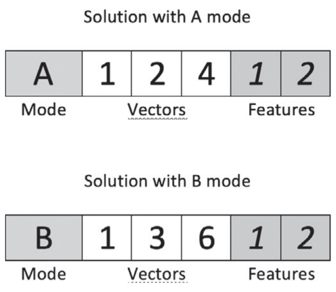


Fig. 4. Encoding for two different solutions of Example 1.

5.3. Initial population

Each one of the solutions of the initial population, sized N, is generated in the following way. First, the mode is generated, each with probability 0.5. Then, the vector array is also generated in a random way. Each vector is chosen from the eligible vectors in the corresponding class. The selected features are also chosen in a random way from the complete features set. Clones are allowed in the initial population. It means that if the new generated created coincides in both objective values with another solution previously generated, it is included in the population. Generating the initial population randomly has an advantage, the reduced computation times needed in contrast to the methods that generate it using some heuristic that can generate solutions with better objective value. These methods can generate better solutions in terms of objective value but they need a considerably larger amount of time. Most metaheuristic methods use the random generation given that, if well designed, they can evolve from this initial population of solutions, with a low quality, to a high quality population.

5.4. Tournament selection based on dominance

This procedure selects two solutions from the current population P_t, a mother and a father, to produce an offspring, copied in the new population Q_t. To select each one of the parents, a tournament is carried out between two solutions randomly chosen from P_t. Let us consider two random candidates participating in the tournament. The winner of the tournament is decided considering, in order, the following criteria, where a criterion is employed if the previous one produce a tie:

1. The winner is the solution which belongs to a lower front.
2. The winner is the solution which dominates a higher number of solutions in P_t.
3. The winner is randomly chosen between both candidates.

The procedure returns two solutions, a mother as winner of the first tournament and a father as the winner of the second. Therefore, the selection is based on the dominance ideas. First, between two solutions in different fronts, we prefer the solution which belongs to a lower front. Otherwise, if both solutions belong to the same front, then we choose the solution which dominates a higher number of solutions in the population. In case of ties, the winner is randomly chosen among the candidates.

Table 4
Metaheuristic population with 8 solutions grouped in 4 different domination fronts.

Sol.	Mode	Vectors	Features	O'_1	O'_2	Front	#Dom. sol.	Dom. Sol.
S_1	B	1 3 6	1 2	4.0931	2.1086	F_1	4	S_4, S_6, S_7, S_8
S_2	B	1 2 5	1 2	1.7650	0	F_1	3	S_5, S_7, S_8
S_3	A	1 2 4	1 2	8.4971	11.6276	F_1	1	S_8
S_4	B	1 5 6	1 2	4	4	F_2	3	S_6, S_7, S_8
S_5	B	2 3 4	1 2	1	0	F_2	1	S_8
S_6	B	2 3 5	1 2	2.6833	5.8138	F_3	1	S_8
S_7	B	1 3 5	1 2	1.3416	4.4721	F_3	1	S_8
S_8	A	1 2 6	1 2	0.8944	25.9384	F_4	0	–

Example 4. Let us suppose that the population of the metaheuristic is formed, during the genetic process, with the 8 solutions presented Table 4 when the dataset presented in Fig. 1 is being processed. The second, third and fourth columns present the information that the encoding includes, the mode, the vectors supported by the hyperplanes and the features selected. Some of these solutions have been used previously in different figures. S_1 matches Figs. 1(a) and 4(b), S_3 matches Fig. 4(a) and S_5 and S_6 correspond to Fig. 1(b) and (c) respectively. Columns 5 and 6 show the values for the two objectives considered in the metaheuristic, O'_1 and O'_2. The next three columns represent the front that each solution belongs to, the number of solutions each solution dominates and the list of these dominated solutions. As we can observe, the population is partitioned into 4 different fronts. The first front, F_1, has three different solutions, S_1, S_2 and S_3 and these are the solutions which are not dominated by any other. The second front, F_2, is made up of the solutions which are dominated by one or more solutions in front F_1 but are not dominated by any solution in fronts F_2, F_3 or F_4. Specifically, S_4 is dominated by S_1 and S_5 is dominated by S_2. F_3 is made up of two solutions, S_6 (dominated by S_1 and S_4) and S_7 (dominated by S_1, S_2 and S_4). The last front, F_4 has only one solution, which is dominated by all the solutions in the population.

If the tournament selection based on dominance is employed to select the best between solutions S_3 and S_4, then S_3 would be selected since it belongs to a lower front. If solutions S_4 and S_5, that belong to the same front, are compared, the winner would be S_4 because it dominates a higher number of solutions. If we compare S_6 and S_7, as they belong to the same front and dominate the same number of solutions, the winner would be selected randomly.

5.5. Crossover

We have designed a specific crossover operator which incorporates knowledge of the problem in order to guide the process to the search of quality solutions. In the crossover process, we first select the best of the parents to undergo the operation, *best_parent*, considering the criteria established in Section 5.4 to determine the winner of the tournaments. Then, the mode of the offspring is inherited from the *best_parent*. Inheriting the mode means also to inherit the structure of the vector array. Then, two different procedures need to be carried out, in this order: (i) the crossover of the features arrays, and (ii) the crossover of the vector arrays. In the following subsections we describe them.

5.5.1. Crossover of features

The first step consists in selecting all the features which are common in both parents. Those features are directly inherited by the offspring. If the number of inherited features is lower than p the rest of features is chosen, one by one, in the following iterative way. First, we select the parent from where to choose the next feature to inherit. The *best_parent* has a probability of $PCROSS\_BEST$ of being chosen and the other a probability of $1-PCROSS\_BEST$. Once the parent selected, a random feature, among the features which have not been inherited yet from its features array, is selected and inherited by the offspring. As higher the $PCROSS\_BEST$ is, more important the role of the *best_parent* is in the crossover of features. If $PCROSS\_BEST = 0.5$ means that the process chooses the parent where to inherit the features from, in a random way, without assigning a prevailing role to the parent with a better performance in terms of dominance ideas. Once the crossover of features has finished, the parent from which more features have been inherited is called the *preferred_parent* and this will be used in the crossover of vectors.

5.5.2. Crossover of vectors

As the *preferred_parent* has had a predominant role in the crossover of features, the idea is to maintain this role in the crossover of vectors. As the mode has been inherited from the *best_parent*, the structure of the vector array also coincides. Let us remind that if the inherited mode is A, the first p positions in the vector array are occupied by vectors in class "−1" and the last position with a vector in class "1". Else, if the offspring mode is B, the first position is occupied by a vector in class "−1" and the rest by vectors in class "1". Therefore, if the offspring mode is A, the first p positions of the vector array are copied from the *preferred_parent*. The last position in the array, the vector in class "1" is randomly chosen from the list of vectors in class "1" in the other parent (the size of this list can be 1 or p depending on the mode of the of this parent). By contrast, if the offspring inherits $mode = B$, it inherits the vectors in class "1" from the *preferred_parent* (positions 2 to $p+1$) and the vector in class "−1" (position 1 of the array) from the list of vectors in class "−1" of the other parent, list which could have 1 or p vectors (depending on its mode). This procedure is presented, in pseudocode, in Algorithm 2. The procedure, receives three arguments, the positions in population Q_t of the two parents to undergo the crossover: the preferred_parent (p1) and the other parent (p2). The last argument is related to the offspring or child generated, c, and is the position in population Q_t where the array of vectors of the offspring must be generated, Q_t[c].vectors.

Algorithm 2: Procedure vectors_crossover(p1, p2, c)

```
1  if $Q_t[c].mode == A$ then /* Preferred parent and child have mode=A */
2      for $k = 1$ to $p$ do
3          $Q_t[c].vectors[k] = P_t[pp].vectors[k]$;
4      if $Q_t[p2].mode == A$ then /* Parents and child have mode=A */
5          $Q_t[c].vectors[p+1] = P_t[p2].vectors[p+1]$;
6      else /* Parents have a different mode */
7          $Q_t[c].vectors[p+1] = select\_1\_feature(P_t[p2].vectors, 2, p+1)$;
8  else /* Preferred parent and child have mode=B */
9      for $k = 2$ to $p+1$ do
10         $Q_t[c].vectors[k] = P_t[p1].vectors[k]$;
11     if $Q_t[p2].mode == B$ then /* Both parents have mode=B */
12         $Q_t[c].vectors[1] = P_t[p2].vectors[1]$;
13     else /* Parents have a different mode */
14         $Q_t[c].vectors[1] = select\_1\_feature(P_t[p2].vectors, 1, p)$;
```

5.6. Mutation

The mutation is a very important mechanism in the evolution of the species and this importance is also present in the multi-objective genetic algorithms. A mutation mechanism not well-designed may force the algorithm to present a premature converge or even to not converge.

Table 5
Large datasets: characteristics.

Instance	#types	#vectors	#features
Colon	2	62	2000
Leukemia	2	72	5327
DBLCL	2	77	7129
Carcinoma	2	36	7457
Arrythmia	2	420	258
Mfeat	2	2000	649

As we mentioned before, every solution will undergo the mutation mechanism with a given probability $PMUT$. The mutation mechanism we propose implies, as in the crossover technique, two different processes, the mutation of features and the mutation of vectors. Both mechanisms follow the same template: they move through the corresponding array, features or vectors, and each item is exchanged with another, chosen in a random way among the items not present in the solution, with a given probability, which may be different in each process. Therefore, each vector in the vector array is exchanged with a different vector of the same class with a probability of *PMUT_VECTORS*. In the same way, each feature in the solution is interchanged with a different feature, chosen randomly, with a given probability of *PMUT_FEAT*. These parameters need to be set in order to run the algorithm to solve a given instance of the problem.

6. Computational experience

In order to compare the performance of both methods, the exact approach (AUGMECON2) when solving the SVM linear programming model and the metaheuristic algorithm (NSGA-II-SVM-f) we have carried out a computational experience, which has been developed on a PC with 2.33 GHz Intel Xeon dual core processor, 8.5 GB of RAM, and operating system LINUX Debian 4.0. The exact method has been solved by CPLEX v11.0 and the evolutionary algorithm in C++.

We have selected different datasets with a big number of features previously employed in different works (see, for example, Labbé et al., 2019). The main characteristics of the instances are given in Table 5. The two last datasets, Arrythmia and Mfeat can be found in the UCI repository (Asuncion & Newman, 2007) and a description of the remaining sets can be found in several works (e.g., Alon et al., 1999; Carrizosa et al., 2010; Golub et al., 1999; Maldonado et al., 2014; Notterman et al., 2001; Shipp et al., 2002). Later, these datasets are considered as instances with a big number of features and, as regard the sample size, the first four are classified as small and the two last as big (Labbé et al., 2019).

The aim of this computational experience is to run both methods solving each instance in order to get the Pareto-optimal front or, at least, the best approximation. Comparing the frontiers given by the methods would lead us to a consistent comparison of their efficiency when solving the Support Vector Machine Problem with Feature Selection.

To compare frontiers given by different multi-objective optimization methods is a complex task, given that we can establish different quality measures. Following Zitler et al. (2000), in general, a good approximation of the Pareto-optimal front should consider three different aspects: The distance between the approximation frontier and the Pareto-optimal front in the objective space should be minimized; A good distribution of the solutions in the approximation front in the solution space is desirable; A wide range of values, for both objectives, should be covered by the solutions in the approximation frontier. However, sometimes, when comparing different approximation methods, the Pareto-optimal front is not known and some of these measures cannot be calculated. There are several performance indicators to determine the quality of an approximated front and they can be classified in three different categories (Audet et al., 2021), :

- Cardinality: Quantify the number of non-dominated points generated by an algorithm.
- Convergence: Quantify how close an approximation frontier is from the Pareto-optimal front in the objective space.
- Distribution: Quantify how well every region of the objective space is represented, the distance between the points in the frontier (spread) and the distance between the extreme points of the front (extent).

There are also some indicators which combine convergence and distribution characteristics. All the datasets have been solved by both methods setting the number of selected features $p = 5$. The experience has not been repeated for other values of p because the objective is not to see how the objective value varies when p varies but to analyze the properties of the boundary in terms of certain metrics. The time limit imposed to the exact method to solve each instance has been fixed in 7 h. First, two hours are employed in a pre-processing procedure in order to get the bounds LB_2 and UB_2 (UB_2 goes straightforward but LB_2 not). Then, the range is divided into five intervals and AUGMECON2 is run for one hour to find a point in the frontier in that interval. When the time limit of one hour is reached, the best feasible solution found so far is retrieved if there is any. Therefore, AUGMECON2 provides a frontier of a maximum of 5 points, and some of these points could be dominated by some others (when the solver is stopped because of the time limit the solution might not be in the Pareto frontier). On the other hand, the time limit imposed to the metaheuristic is of one hour per instance. Within this time limit, the metaheuristic obtains an approximation of the Pareto-optimal front, with a maximum of N solutions, being N the population size, which is a parameter to be established before running the algorithm. The metaheuristic gives, as a result, all the non-dominated solutions which are present in the final population, which form the frontier required. Therefore, this frontier will be built with a maximum of N individuals. Given that the metaheuristic is a randomized technique and running it different times may lead to different results, we have run the metaheuristic three separate times per instance and we have calculated the average in all the metrics computed. In order to establish a fixed configuration for the metaheuristic to run all the datasets we have carried out several preliminary experiments. Based on that we have decided to use the following configuration in all the instances: $N = 500$; $PCROSS\_BEST = 0.8$; $PMUT = 0.7$; $PMUT\_FEAT = PMUT\_VECTORS = 0.4$. However, these preliminary studies indicated that slightly varying the above parameters, except the population size, does not have a great influence in the final results.

Given that we are employing the exact method (AUGMECON2) with a time limit, the outcome, i.e. the frontier given by this method becomes an approximation frontier. Therefore, we do not have a Pareto-optimal front to compare with and both methods give approximation frontiers that we want to compare.

We have calculated four different indicators in order to compare both approximation frontiers: two cardinality indicators and two distribution indicators. As regard the last, one is to compare the extent and the other to compare the spread. All these measures do not depend on external parameters and therefore the comparison is much more robust. The first indicator is the so-called Overall non-dominated Vector Generation (OVNG), proposed by Veldhuizen and David (1999) that returns the number of non-dominated points in the approximation frontier. The second indicator is the C-metric proposed by Zitler and Thiele (1998) and used later in several studies (e.g., Zitler et al., 2000) and gives, for two frontiers, F_1, F_2 the fraction of solutions in F_2 that are dominated by one or more solutions in F_1, $C(F_1, F_2)$. In order to compare the maximum extent in which the front spreads out we calculate the Zitler's metric M_3 (Zitler et al., 2000) which, in the case of two objectives, this equals the distance of the two outer solutions, and consequently, a higher distance is desired. The last considered indicator to compare both approximation frontiers is the one proposed by Custòdio et al.

Table 6
AUGMECON2 vs. NSGA-II. Comparison of frontiers for large instances.

Instance	AUGMECON2 (7 h)				NSGA-II-SVM-f (1 h)			
	OVNG	C	M_3	Γ	OVNG	C	M_3	Γ
Colon	3	66.70%	58.12	45.32	500.00	3.00%	91.25	1.54
Leukemia	5	60.00%	253.36	163.55	299.33	2.25%	109.70	5.80
DBLCL	5	33.33%	60.11	35.31	306.33	11.72%	79.01	1.11
Carcinoma	4	41.67%	152.18	108.22	94.67	0.00%	46.08	4.62
Arrythmia	5	40.00%	494.31	777.78	500.00	2.27%	597.40	9.85
Mfeat	5	26.67%	653.05	336.23	500.00	9.13%	1283.83	201.01
Average	4.50	44.72%			366.72	4.73%		

(2011), Γ, that, when considering a bi-objective problem, reduces to consider the maximum distance between two consecutive points in the Pareto front approximation, therefore, a lower value of Γ is desirable.

In Table 6 we show the metrics calculated for both methods when solving the datasets presented in Table 5. The time limit for AUGMECON2 is 7 h and for the metaheuristic of 1 h. For each method, $OVNG$ indicates the number of non-dominated points in the approximation frontier obtained as outcome, C, the percentage of solutions in the frontier given by the corresponding method which are dominated by points of the frontier generated by the other method, M_3 measures the extent of the frontier and Γ the size of holes in the front.

If we look first to the cardinality indicators we can deduce that the metaheuristic obtains in 1 h of computation time better frontiers than the exact approximation in 7 h. OVNG indicates that AUGMECON2 gives frontiers with 4 non-dominated points, on average, compared to the 366 non-dominated solutions, on average, of the metaheuristic frontiers. The number of points varies between 3 and 5 with AUGMECON2 and between around 34 and 500 in the case of NSGA-II. However, it is interesting to study how many points in a frontier are dominated by points on the other frontier. The C-metric indicates that, in most of the instances solved there are points in the approximation frontier given by one method that are dominated by points in the other frontier. In the case of AUGMECON2, the 44.72%, on average, of the frontiers given by the method are dominated by points in the NSGA-II frontier. On the other hand, only a 4.73% of the NSGA-II frontier, on average, are dominated by points of the approximation given by the exact method. In the Carcinoma dataset, for instance, none of the points in the frontier found by the metaheuristic is dominated by points of the AUGMECON frontier. Therefore, the two cardinality metrics are much better, by far, for the metaheuristic frontiers.

Let us analyze now the distribution indicators, that in this case, measure the extent and the distance between consecutive points in the frontiers. As M_3 calculates the extent of the frontier, a higher value indicates a better performance. AUGMECON2 gives frontiers with a better extent in only two (Leukemia and Carcinoma) of the six datasets, showing a better performance of the metaheuristic in the other four. The last indicator, Γ, which evaluates the maximum distance between two adjacent points in a frontier shows that the frontiers given by the metaheuristic approach are better in all the datasets solved. High values of this indicator for a method show that the generated frontiers present big holes and this is not a desirable fact. In all the datasets of our study, the Γ values are much higher for AUGMECON2 that for the NSGA-II approach, and in most cases is more than 30 times higher, indicating that in the metaheuristic approximations the points are much closer and therefore the frontier shows a better spread. As the four indicators show different aspects of the frontiers and one frontier could be better than the other in one indicator but worse in another and in order to determine the best approximation for each dataset, we have determined the number of indicators, for each dataset, in which one approach is better than the other. The conclusion is that the frontiers given by the metaheuristic are better than those given by the exact approach in all the datasets: NSGA-II gives better results in all the indicators in 4 datasets (Colon, DBLCL, Arrythmia and Mfeat) and in 3 of the 4 indicators for the rest of the datasets (Leukemia and Carcinoma).

As an example, we graphically present the comparison of frontiers for the Colon dataset in Fig. 5. The graph shows the results for one of the three runs of the metaheuristic. It is interesting to point out that the computation time employed by AUGMECON2 to generate the frontier is of seven hours and the employed by the metaheuristic of only one hour. We can clearly observe that the frontier obtained by AUGMECON2 is formed with only three non-dominated points (OVNG) compared to the 500 points given by NSGA-II. Moreover, two of these three points ($C = 66.7\%$) are dominated by points in the metaheuristic frontier. By contrast, only a few points in the NSGA-II frontier are dominated by those in the AUGMECON2 frontier. Concerning the extent, the range of values covered by the metaheuristic frontier is larger than the one corresponding to the AUGMECON2 frontier and this one presents larger holes, i.e. the maximal distance between adjacent points is much higher.

The previous computational experiments show that the metaheuristic is a good alternative to the exact method in those datasets in which the last cannot obtain the Pareto-optimal front due to the computational effort required. However, we also want to show that the evolutionary algorithm performs well in different datasets in which the exact approach can obtain the optimal front. We have also solved 4 different datasets, which have been also obtained from the UCI repository (Asuncion & Newman, 2007). The characteristics of these datasets are presented in Table 7. All the datasets have been solved by both methods with a selection of 5 features. AUGMECON2 has been running for one hour, and the non-dominated points which are obtained represent our Pareto-Optimal front. We have run the metaheuristic approach three separate times, imposing a time limit of one seventh of an hour and then we have calculated the average for all the metrics considered. In the previous experiments the metaheuristic also used one seventh of the time employed by the exact approach. We have also computed the measures used in the previous experiments and we have added two metrics, which combine convergence and distribution ideas, given that now, we have the optimal front to compare with. The first new metric, named modified inverted generational distance, $IGD+$, was proposed by Ishibuchi et al. (2015) and overcomes the drawbacks presented by GD (Veldhuizen & David, 1999) and IGD (Coello & Cortés, 2005). Following Audet et al. (2021) this measure takes into account the dominance relation between the points of the frontiers to be compared when computing the Euclidean distance and it is weakly Pareto compliant. As it represents a distance between the frontiers, a lower value is considered to be better. The second metric that permits to evaluate the quality of the approximation frontier compared with the Pareto-optimal front is the hyper volume ratio HVR or S-metric, which was proposed by Zitzler (1999). The hyper volume indicator, HV determines the volume of the space in the objective space dominated by the front generated by a given method. Therefore, the HVR metric computes the proportion of the space dominated by the Pareto-optimal front which is dominated by the approximation method.

The results are shown in Table 8. First of all, if e look at the OVNG metric, we can observe that the metaheuristic always gives a frontier with 500 solutions in contrast to the exact method, which gives the exact frontier with a variable number of points and it is always much less. Logically, none of the points in the AUGMECON2 frontier is

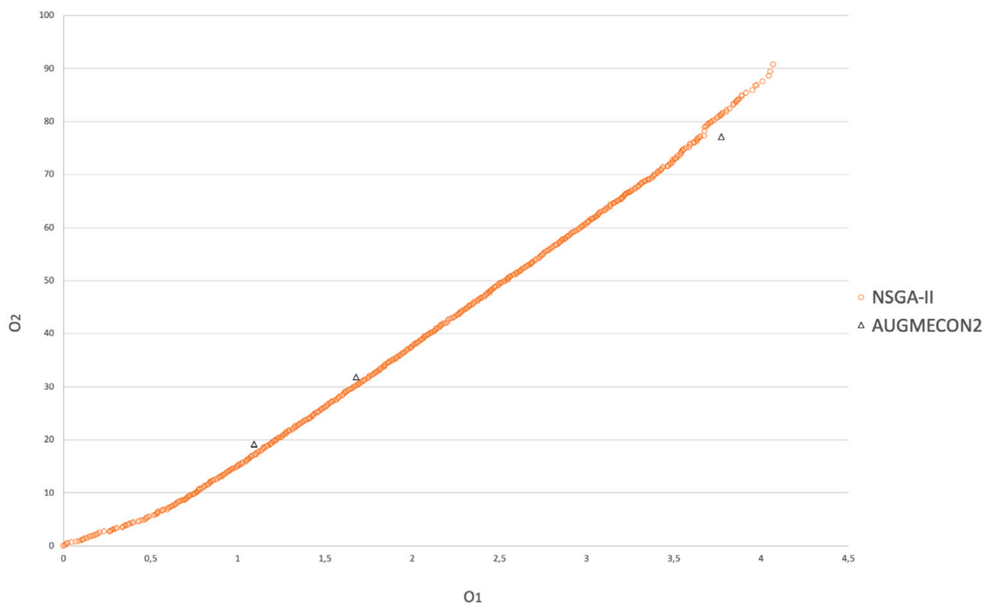


Fig. 5. Graphic comparison of the approximations given by AUGMECON2 (7 h) and NSGA-II (1 h) for the Colon dataset.

Table 7
Small datasets: characteristics.

Instance	#types	#vectors	#features
Housing	2	506	13
GC	2	1000	24
WBC	2	569	30
Iono	2	351	33

dominated by points of the approximation frontier and therefore, the C metric equals 0 in all the datasets. The percentage of approximation frontier dominated by the corresponding exact frontier varies from 2.87% in the GC dataset to more than 50% for the WBC instance. It means that some of the points of the approximation frontier are not dominated by points in the exact frontier. As regard the extent given my the M_3 metric, it is always higher in the frontiers given by the metaheuristic, except for the last dataset, Iono, in which they are rather similar. The Γ metric indicates a better performance for AUGMECON2 in 3 of the 4 the datasets, that is, the larger hole between adjacent points, is nearly always present in the approximation frontier, but in most cases, the differences are not very large. Looking at the new metrics $IGD+$ does not permit us to compare results, given that it measures the distance from exact and metaheuristic frontiers and we have only one metaheuristic method in our analysis. However, we can observe that the distance is rather low in all the instances, which indicates that the approximation frontier is quite near of the Pareto-optimal front. These values permit other researchers to compare new methods with our proposal in future research. Finally, the HVR metric indicates that the portion of Pareto-optimal front covered by the approximation frontier is always higher than 50% and in two of the four datasets, more than 85%. On the other hand, the number of support vectors is different for each Pareto front solution and it is usually large. The smallest dataset in terms of # vectors is the Iono instance, having 225 vectors in the positive class and 126 in the negative class. For this small instance, the number of positive class support vectors is in the interval $[99, 178]$ and the number of negative class support vectors is in the interval $[0, 99]$.

From these results we can conclude that, although the metaheuristic has been designed with the aim of being used to solve large instances in which the exact method cannot obtain the Pareto-optimal front, it has also demonstrated a good performance in small instances, here the exact method is capable of generating the optimal front.

Fig. 6 shows, as example, the frontiers given by both methods, AUGMECON2 with 1 h of computation time and NSGA-II with 1/7 h (in one of the three runs) when solving the WBC dataset. Now, none of the points given by the exact method is dominated by a point of the heuristic frontier. However, we can observe that the metaheuristic frontier presents larger holes ($\Gamma = 50.46$) than the exact frontier ($\Gamma = 10.89$). Moreover, both frontiers are rather near ($IGD+ = 0.1026$) and most of the region covered by the exact frontier is also covered by the metaheuristic frontier ($HVR = 91.97\%$). Let us recall that we are comparing the frontier given by the exact method in 3600 s. in contrast to the 514 s. employed by the metaheuristic.

7. Conclusions

In this paper, we deal with the classical soft margin Support Vector Machine problem with feature selection, where two objectives are considered, from a multi-objective perspective. The solution to multi-objective problems is a set of non-dominated solutions instead of a unique solution. From a mathematical point of view, getting the Pareto-optimal frontier in some real problems, or at least a good approximation, is a challenging task and represents the objective of this paper. However, it is not only interesting from a theoretical perspective. In practice, this frontier provides the decision maker with a rich set of alternative solutions from which a better decision can certainly be made. The exact methods require an excessive computational effort to compute each one of the points in the frontier, even in small instances. We have demonstrated how exact methods fail in the task of getting the Pareto-optimal front or even an approximation when the size of the instances increases. We propose a metaheuristic technique to solve the problem and obtain a good description of the Pareto frontier. To demonstrate the efficiency of the proposed method, we have carried out a computational experiment solving some well-known instances with an exact method and our algorithm. The quality of a frontier cannot be measured in terms of only one metric but rather a set of them. The results show that the metaheuristic obtains good descriptions of the frontiers in small instances, where the exact method is capable of finding a frontier although with a considerable computation time. In large instances, in which the exact method obtains only some few points of the frontier (which are sometimes dominated), the proposed algorithm returns frontiers of quality in terms of cardinality, distribution and convergence.

Table 8
AUGMECON2 vs. NSGA-II. Comparison of frontiers for small instances.

Instance	AUGMECON2 (1 h)					NSGA-II-SVM-f (1/7 h)						
	time (s)	OVNG	C	M_3	Γ	Time (s)	OVNG	C	M_3	Γ	$IGD+$	HVR
Housing	3600	30	0.00%	496.09	22.17	514	500.00	9.73%	573.36	19.08	0.1642	58.75%
GC	3600	13	0.00%	135.88	13.74	514	500.00	2.87%	1285.35	207.85	0.0095	97.75%
WBC	3600	74	0.00%	480.21	10.88	514	500.00	51.27%	553.04	41.90	0.0882	86.70%
Iono	3600	93	0.00%	378.69	6.43	514	500.00	39.67%	358.27	10.03	0.3106	52.97%

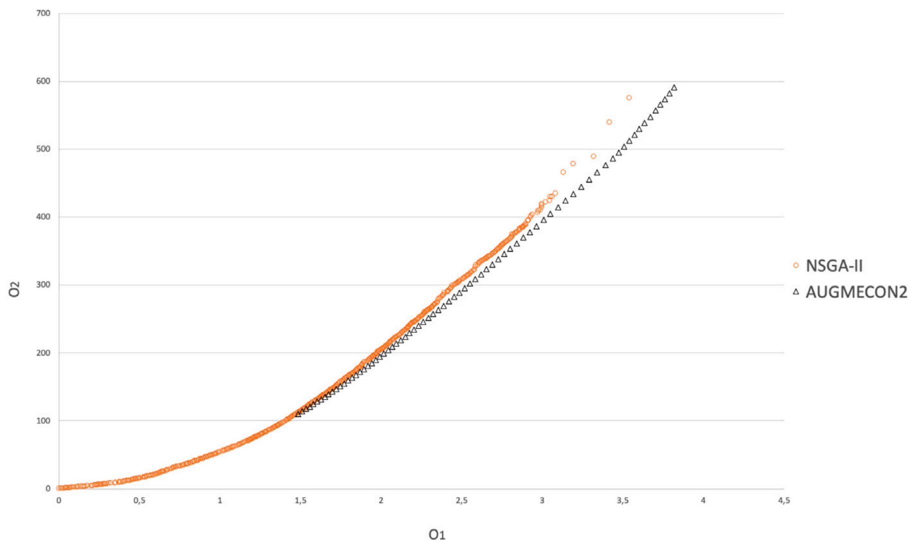


Fig. 6. Graphic comparison of the frontiers given by AUGMECON2 (1 h) and NSGA-II (1/7 h) for the WBC dataset.

Having the Pareto-optimal front or a good approximation, with a considerable number of points is important in practice since it provides a wide variety of classifiers. Later, the selection of one or another could be done through ROC analysis, for example, being this one of the future lines of research of this paper. It would be interesting to know whether the points furthest to the right of the Pareto-optimal front are the best in terms of metrics such as AUC or F-score, or if the best in terms of these metrics are uniformly distributed along the frontier. Another interesting future line could consist of analyzing the relation of the objectives considered in this paper with the classification quality metrics commonly used to analyze the classification performance.

CRediT authorship contribution statement

Javier Alcaraz: Ideas, Methodology, Validation, Investigation, Data curation, Writing, Supervision, Funding acquisition, Project administration. **Martine Labbé:** Ideas, Methodology, Validation, Investigation, Data curation, Writing, Supervision, Project administration. **Mercedes Landete:** Ideas, Methodology, Validation, Investigation, Data curation, Writing, Supervision, Funding acquisition, Project administration.

Declaration of competing interest

The authors declare that they have no known competing financial interests or personal relationships that could have appeared to influence the work reported in this paper.

Acknowledgments

The authors thank the grants PID2019-105952GB-I00 funded by Ministerio de Ciencia e Innovación/ Agencia Estatal de Investigación /10.13039/501100011033, Spain, and PROMETEO/2021/063 funded by the government of the Valencian Community, Spain.

References

Al-Zoubi, A. M., Hassonah, M. A., Heidari, A. A., Faris, H., Mafarja, M., & Aljarah, I. (2021). Evolutionary competitive swarm exploring optimal support vector machines and feature weighting. *Soft Computing*, *25*(4), 3335–3352.

Aladeemy, M., Tutun, S., & Khasawneh, M. T. (2017). A new hybrid approach for feature selection and support vector machine model selection based on self-adaptive cohort intelligence. *Expert Systems with Applications*, *88*, 118–131.

Alcaraz, J., Landete, M., Monge, J., & Sainz-Pardo, J. L. (2020). Multi-objective evolutionary algorithms for a reliability location problem. *European Journal of Operational Research*, *283*, 83–93.

Alon, U., Barkai, N., Notterman, D., Gish, K., Ybarra, S., Mack, D., & Levine, A. (1999). Broad patterns of gene expression revealed by clustering analysis of tumor and normal colon tissues probed by oligo-nucleotide arrays. *Proceedings of the National Academy of Sciences*, 6745–6750.

Asuncion, A., & Newman, D. (2007). UCI machine learning repository. URL http://www.ics.uci.edu/~mlearn/MLRepository.html.

Audet, C., Bigeon, J., Cartier, D., Digabel, S. L., & Salomon, L. (2021). Performance indicators in multiobjective optimization. *European Journal of Operational Research*, *292*, 397–422.

Aytug, H. (2015). Feature selection for support vector machines using generalized benders decomposition. *European Journal of Operational Research*, *244*(1), 210–218.

Bouraoui, A., Jamoussi, S., & Benayed, Y. (2018). A multi-objective genetic algorithm for simultaneous model and feature selection for support vector machines. *Artificial Intelligence Review*, *50*, 261–281.

Bradley, P., & Mangasarian, O. (1998). Feature selection via concave minimization and support vector machines. *ICML*, *98*, 82–90.

Burges, C. J. C. (1998). A tutorial on support vector machines for pattern recognition. *Data Mining and Knowledge Discovery*, *2*(2), 121–167.

Carrizosa, E., Martin-Barragan, B., & Morales, D. R. (2010). Binarized support vector machines. *INFORMS Journal on Computing*, *22*, 154–167.

Cheng, F., Chen, J., Qiu, J., & Zhang, L. (2020). A subregion division based multi-objective evolutionary algorithm for SVM training set selection. *Neurocomputing*, [ISSN: 0925-2312] *394*, 70–83.

Coello, C., & Cortés, C. (2005). Solving multiobjective optimization problems using an artificial immune system. *Genetic Programming and Evolvable Machines*, *6*, 163–190.

Cortes, C., & Vapnik, V. (1995). Support-vector networks. *Machine Learning*, *20*(3), 273–297.

Cuong, N. M., & Thien, N. V. (2016). A method for reducing the number of support vectors in fuzzy support vector machine. In T. Nguyen, T. Do, H. A. L. Thi, & N. Nguyen (Eds.), *Advanced computational methods for knowledge engineering* (pp. 17–27). Springer.

Custòdio, A. L., Madeira, J., Vaz, A., & Vicente, L. (2011). Direct multisearch for multiobjective optimization. *SIAM Journal on Optimization*, *21*, 1109–1140.

Deb, K., Pratap, A., Agarwal, S., & Meyarivan, T. (2002). A fast and elitist multi-objective genetic algorithm: NSGA-II. *IEEE Transactions on Evolutionary Computation*, *6*, 182–197.

Dudzik, W., Nalepa, J., & Kawulok, M. (2021). Evolving data-adaptive support vector machines for binary classification. *Knowledge-Based Systems*, *227*, Article 107221.

Faris, H., Hassonah, M. A., Al-Zoubi, A., Mirjalili, S., & Aljarah, I. (2018). A multi-verse optimizer approach for feature selection and optimizing SVM parameters based on a robust system architecture. *Neural Computing and Applications*, *30*, 2355–2369.

García-Pedrajas, N., de Haro-García, A., & Pérez-Rodríguez, J. (2014). A scalable memetic algorithm for simultaneous instance and feature selection. *Evolutionary Computation*, *22*, 1–45.

Gaudioso, M., Gorgone, E., Labbé, M., & Rodríguez-Chía, A. (2017). Lagrangian relaxation for SVM feature selection. *Computers & Operations Research*, *87*, 137–145.

Gauthama Raman, M., Somu, N., Kirthivasan, K., Liscano, R., & Shankar Sriram, V. (2017). An efficient intrusion detection system based on hypergraph - genetic algorithm for parameter optimization and feature selection in support vector machine. *Knowledge-Based Systems*, *134*, 1–12.

Geebelen, D., Suykens, J. K., & Vandewalle, J. (2012). Reducing the number of support vectors of SVM classifiers using the smoothed separable case approximation. *IEEE Transactions on Neural Networks and Learning Systems*, *23*, 682–688.

Golub, T., Slonim, D., Tamayo, P., Huard, C., Gaasenbeek, M., Merisov, J., Coller, H., Loh, M., Downing, J., Caligiuri, M., Bloomfeld, C., & Lander, E. (1999). Molecular classification of cancer: Class discovery and class prediction by gene expression monitoring. *Science*, *286*, 531–537.

Huang, C.-L., & Wang, C.-J. (2006). A GA-based feature selection and parameters optimizationfor support vector machines. *Expert Systems with Applications*, *31*(2), 231–240.

Ibrahim, H. T., Mazher, W. J., Ucan, O. N., & Bayat, O. (2019). A grasshopper optimizer approach for feature selection and optimizing SVM parameters utilizing real biomedical data sets. *Neural Computing and Applications*, *31*(10), 5965–5974.

Ishibuchi, H., Masuda, H., Tanigaki, Y., & Nojima, Y. (2015). Modified distance calculation in generational distance and inverted generational distance. In A. Gaspar-Cunha, C. H. Antunes, & C. Coello (Eds.), *Evolutionary multi-criterion optimization* (pp. 110—125). Springer.

Labbé, M., Martínez-Merino, L., & Rodríguez-Chía, A. M. (2019). Mixed integer linear programming for feature selection in support vector machine. *Discrete Applied Mathematics*, *261*, 276–304.

Lin, S.-W., Ying, K.-C., Chen, S.-C., & Lee, Z.-J. (2008). Particle swarm optimization for parameter determination and feature selection of support vector machines. *Expert Systems with Applications*, [ISSN: 0957-4174] *35*(4), 1817–1824.

Maldonado, S., Perez, J., Weber, R., & Labbé, M. (2014). Feature selection for support vector machines via mixed integer linear programming. *Information Sciences*, *279*, 163–175.

Mavrotas, G. (2009). Effective implementation of the ϵ-constraint method in multi-objective mathematical programming problems. *Applied Mathematics and Computation*, *213*, 455–465.

Mavrotas, G., & Florios, K. (2013). An improved version of the augmented e-constraint method (AUGMECON2) for finding the exact pareto set in multi-objective integer programming problems. *Applied Mathematics and Computation*, *219*, 9652–9669.

Notterman, D., Alon, U., Sierk, & Levine, A. (2001). Transcriptional gene expression profiles of colorectal adenoma, adenocarcinoma, and normal tissue examined by oligonucleotide arrays. *Cancer Research*, *61*, 3124–3130.

Pardalos, P., Žilinskas, A., & Žilinskas, J. (2017). *Non-convex multi-objective optimization* (p. 123). Springer Optimization and its Applications.

Shipp, M., Ross, K., Tamayo, P., Weng, A. P., Kutok, J., Aguiar, R., Gaasenbeek, M., Angelo, M., Reich, M., Pinkus, G., Ray, T., Koval, M., Last, K., Norton, A., Lister, T., Mesirov, J., Neuberg, D., Lander, E., Aster, J., & Golub, T. R. (2002). Diffuse large b-cell lymphoma outcome prediction by gene-expression profiling and supervised machine learning. *Nature Medicine*, *8*, 68–74.

Srinivas, N., & Deb, K. (1995). Multiobjective function optimization using nondominated sorting genetic algorithms. *Evolutionary Computation*, *2*, 221–248.

Tao, Z., Huiling, L., Wenwen, W., & Xia, Y. (2019). GA-SVM based feature selection and parameter optimization in hospitalization expense modeling. *Applied Soft Computing*, *75*, 323–332.

Vapnik, V. (1995). *The nature of statistical learning theory*. Springer Science Business Media.

Vapnik, V. (2013). *The nature of statistical learning theory*. Springer science & business media.

Vapnik, V., & Chervonenkis, A. (1964). A note on one class of perceptrons. *Automatic Remote Control*, *25*.

Vapnik, V., & Chervonenkis, A. (1974). *Theory of pattern recognition (in Russian)*. Moscow: Nauka.

Veldhuizen, V., & David, A. (1999). *Multiobjective evolutionary algorithms: classifications, analyses, and new innovations*: *Technical report*, Dayton, Ohio: School of Engineering of the Air Force Institute of Technology.

Zhao, M., Fu, C., Ji, L., Tang, K., & Zhou, M. (2011). Feature selection and parameter optimization for support vector machines: A new approach based on genetic algorithm with feature chromosomes. *Expert Systems with Applications*, *38*(5), 5197–5204.

Zitler, E., Deb, K., & Thiele, L. (2000). Comparison of multiobjective evolutionary algorithms: Empirical results. *Evolutionary Computation*, *8*, 173–195.

Zitler, E., & Thiele, L. (1998). Multiobjective optimization using evolutionary algorithms—A comparative case study. In A. Eiben, & et al. (Eds.), *Parallel problem solving from nature* (pp. 292—301). Springer.

Zitzler, E. (1999). *Evolutionary algorithms for multiobjective optimization: Methods and applications* (Ph.D. thesis), Swiss Federal Institute of Technology.