https://doi.org/10.1016/j.knosys.2018.02.022

A multi-objective evolutionary approach to training set selection for support vector machine

Giovanni Acampora[a,*], Fransisco Herrera[b,c], Genoveffa Tortora[d], Autilia Vitiello[d]

[a] *Department of Physics "Ettore Pancini", University of Naples Federico II, 80126 Naples, Italy*
[b] *Department Computer Science and Artificial Intelligence, University of Granada, E-18071 Granada, Spain*
[c] *Faculty of Computing and Information Technology, King Abdulaziz University, 21589, Jeddah, Saudi Arabia*
[d] *Department of Computer Science, University of Salerno, 84084 Fisciano, Italy*

Abstract

The Support Vector Machine (SVM) is one of the most powerful algorithms for machine learning and data mining in numerous and heterogenous application domains. However, in spite of its competitiveness, SVM suffers from scalability problems which drastically worsens its performance in terms of memory requirements and execution time. As a consequence, there is a strong emergence of approaches for supporting SVM in efficiently addressing the aforementioned problems without affecting its classification capabilities. In this scenario, methods for Training Set Selection (TSS) represent a suitable and consolidated preprocessing technique to compute a reduced but representative training dataset, and improve SVM's scalability without deprecating its classification accuracy. Recently, TSS has been formulated as an optimization problem characterized by two objectives (the classification accuracy and the reduction rate) and solved through the application of evolutionary algorithms. However, so far, all the evolutionary approaches for TSS have been based on a so-called multi-objective *a priori* technique, where multiple objectives are aggregated together into a single

☆Fully documented templates are available in the elsarticle package on CTAN.

*Corresponding author

Email addresses: giovanni.acampora@unina.it (Giovanni Acampora), herrera@decsai.ugr.es (Fransisco Herrera), tortora@unisa.it (Genoveffa Tortora), avitiello@unisa.it (Autilia Vitiello)

objective through a weighted combination. This paper proposes to apply, for the first time, a Pareto-based multi-objective optimization approach to the TSS problem in order to explicitly deal with both its objectives and offer a better trade-off between SVM's classification and reduction performance. The benefits of the proposed approach are validated by a set of experiments involving well-known datasets taken from the UCI Machine Learning Database Repository. As shown by statistical tests, the application of a Pareto-based multi-objective optimization approach improves on state-of-the-art TSS techniques and enhances SVM efficiency.

Keywords: Training set selection, Multi-objective optimization, Support Vector Machine

1. Introduction

The Support Vector Machine (SVM) is a supervised learning method rooted in the Statistical Learning Theory, developed by Vladimir Vapnik [1] and co-workers at AT&T Bell Laboratories in 1995. SVM has demonstrated highly competitive performance in numerous real-world applications, such as bioinformatics, text mining, face recognition, and image processing, which has established SVM as one of the state-of-the-art tools for machine learning and data mining [2]. However, SVM suffers from a widely recognized scalability problem [3] because its execution time increases rapidly with the dimension of the training data. Moreover, in many real-world applications, training data can contain noisy or wrong information and also the performance of best classifiers could worsen when they deal with these data [4].

Training Set Selection (TSS) [5] represents a suitable and consolidated approach to face these problems. TSS consists of a pre-processing technique that selects only the relevant dataset instances before performing training and classification tasks. Thanks to the creation of a reduced training dataset composed of the most relevant instances, TSS techniques achieve a twofold benefit: on one hand, the accuracy of SVM can be improved, while, on the other hand, the

computation complexity is decreased.

Actually, TSS has mainly been investigated for the k-Nearest Neighbor (k-NN) classifier, where it is referred to as Prototype Selection (PS). Research studies in this field have shown how PS can successfully improve the accuracy of the k-NN classification [6]. Hence, research efforts have been carried out to study whether TSS techniques can improve also the SVM performance. In particular, Verbiest et al. in [4] show that TSS strategies based on evolutionary approaches such as genetic algorithms can significantly improve the accuracy of SVM classification. The developed evolutionary algorithms address TSS as a combinatorial optimization problem characterized by two objectives: 1) maximize the number of the instances that are correctly classified by a SVM classifier when using the reduced training data and 2) maximize the amount of reduction achieved with respect to the original training set. The evolutionary algorithms developed so far use an *a priori* approach where multiple objectives are combined into a single one through a linear aggregation based on a so-called *equilibrate factor*.

However, *a priori* approaches suffer from some well-known drawbacks [7]: (1) they require to explicitly and exactly set the equilibrate factor; (2) as discussed in a large number of studies [8][9], they fail in capturing optimal solutions in non-convex regions due to the linear nature of the used aggregation. Moreover, the principal reason why a problem has a multi-objective formulation is because it is not possible to have a single solution which simultaneously optimizes all objectives. Therefore, an algorithm that gives a large number of alternative solutions in a single run could be of great practical value [10].

Starting from these considerations, this paper proposes to apply, for the first time, a multi-objective *a posteriori* approach (also known as *Pareto-based multi-objective optimization* algorithm) to the TSS problem. Pareto-based multi-objective optimization algorithms can afford to multiple-objective nature of the TSS problem by producing in each run a set of viable alternative solutions (referred to as non-dominated solutions) representing the best possible trade-offs between the antagonistic objectives characterizing it. Ideally, each alternative

solution produced is optimal in the sense that it will not be possible to improve the value of any one of the objectives without simultaneously degrading the quality of one or more of the other objectives. Such a solution set is called *Pareto-optimal set*. Over the years, several Pareto-based multi-objective optimization algorithms have been developed and successfully applied to relevant and variegated application domains such as gene expression [11], handwritten character recognition [12], ontology alignment [13] among others. In particular, in this work, the TSS problem is addressed by applying one of the most popular Pareto-based multi-objective optimization algorithms, the Pareto Envelope-based Selection Algorithm II (PESA-II) [14]. The most attractive characteristic of PESA-II is the grid-based fitness assignment mechanisms that maintain diversity in both environmental selection and mating selection [15]. However, having a set of solutions instead of a single one does not make a TSS approach applicable automatically in real scenarios. Therefore, the proposed TSS technique, that we call *ParetoTSS*, includes a decision-making mechanism based on a sum model to select a single solution from the non-dominated set.

The performance of ParetoTSS is investigated in a set of experiments involving well-known datasets taken from the UCI Machine Learning Database Repository [16]. The experiments are divided in three sessions: in the first one, the performance of ParetoTSS is analyzed by means of a validation procedure as realistic as possible; in the second one, the ability of ParetoTSS in improving SVM is studied in terms of classification time and accuracy; finally, in the third experimental session, a comparison between ParetoTSS and the state-of-the-art approaches is carried out in terms of accuracy and reduction rate. After performing an empirical analysis and a set of non-parametric statistical tests, the results show that ParetoTSS is characterized by a high performance and its application leads to the increasing of SVM classification accuracy and efficiency and improvements on state-of-the-art TSS techniques.

The paper is organized as follows. Section 2 presents a description of the TSS problem and a review of the state-of-the-art approaches aimed at addressing it. A formulation of the TSS problem as a Pareto-based multi-objective

optimization problem is given in Section 3. In Section 4, we give details about the proposed algorithm, ParetoTSS. A detailed discussion of the experiments and of the results obtained by the algorithm ParetoTSS are reported in Section 5. Section 6 concludes the paper.

2. Training set selection and State-of-the-art

With its solid theoretical foundation and also proven effectiveness (see Appendix A), SVM has contributed to researchers' success in many fields [17]. However, SVM suffers from a widely recognized scalability problem in both memory requirement and computational time [3]. SVM computation and memory requirements increase rapidly with the number of instances in the data sets. Precisely, the computation time of SVM training is quadratic in the number of training instances [17]. Hence, research efforts have been carried out to develop TSS techniques with the goal of reducing the training data size without losing classification accuracy. Training Set Selection (TSS) is a pre-processing method used to select relevant instances in the training data before applying a classifier. TSS is very useful in many real-world applications, where datasets can contain noisy or wrong information that make difficult the classification also for the best classifiers.

Formally, let TR be the original training set composed of n instances. Each instance I_i is a pair (x_i, y_i) with $i = 1, \ldots, n$, where x_i defines an input vector of attributes and y_i defines the corresponding class label. Each input vector contains m input attributes that are quantitative or qualitative information that describe the corresponding instance. The goal of any TSS technique is to produce a set of instances $S \subseteq TR$ to be used to train a classifier capable of classifying new instances with the same or higher classification accuracy of the same classifier trained with the original training data TR. The cardinality of the set S should be smaller than the cardinality of the set TR in order to reduce the classifier time complexity. The amount of reduction of S with respect to TR is referred to as *reduction rate*. Starting from this description, it is possible

to state that the success of a TSS technique is assessed by using two aspects: (1) the classification accuracy obtained by the classifier when using the reduced set of instances, and (2) the amount of reduction achieved with respect to the original training set. TSS techniques that provide the best trade-off between both measures are the best performers. Hereafter a description of the TSS techniques implemented so far is given.

2.1. State-of-the-art

As we have mentioned, TSS has been successfully applied to the k-NN classifier, where it is referred to as Prototype Selection (PS) [6] [18] [19] [20] [21]. However, recently researchers are investigating about the TSS-based improvement of performance of other kinds of classifiers [22], such as SVM. In particular, in this scenario, two different research branches are arising: the former focuses on the application of evolutionary algorithms, whereas, the latter pays particular attention to non-evolutionary techniques such as clustering and data geometry.

As for the first category, research efforts have been mainly focused on adapting evolutionary-based approaches already developed for KNN to SVM. In particular, in the recent work of Verbiest et al. [4], a set of systematic experiments has been performed to compare three evolutionary-based TSS techniques, GGA [23], CHC [24] and SSGA [24], originally created for KNN, both among them and with no-evolutionary approaches such as ENN [25] and MCIS [26]. According to these experiments, GGA results to be the most suited TSS method for SVM, by establishing itself as the state-of-the-art for evolutionary-based TSS approaches. Briefly, GGA is a TSS technique based on genetic algorithms. Therefore, it manages a population, randomly initialized, that evolves through the application of crossover and mutation operators. The main feature of this algorithm is the exploitation of two mutation rates: 1) the probability of introducing an instance referred to as p_{m1} and 2) the probability of removing an instance referred to as p_{m0}. The probability p_{m1} is smaller than the probability p_{m0} in order to force the algorithm to obtain higher reduction rates. The evaluation of the reduced training set is based on two objectives: the accuracy

obtained on the training data set and the reduction rate of the reduced training set with respect to the original training data. These two values are balanced by using a so-called *equilibrate factor* α. The algorithm ends after a prefixed number of fitness evaluations.

However, in spite of the application of evolutionary algorithms to TSS for SVM, Pareto-based multi-objective optimization approaches have not been investigated so far, even though these approaches have been applied to SVM for other optimization goals. In particular, several research efforts have been made to address the selection of SVM parameters through Pareto-based multi-objective optimization algorithms such as NSGA-II [27][28] and MOPSO [29] by obtaining good results. Some preliminary studies have been conducted to reduce the complexity time of SVM through Pareto-based multi-objective optimization algorithms. In [30], a Pareto-based multi-objective optimization algorithm such as NSGA-II is applied to reduce the classification complexity time of SVM by selecting the best support vectors among a set of support vectors obtained by applying a standard algorithm to train the SVM. It is worth noting that the selection task refers to support vectors and not training instances. Moreover, in [31], NSGA-II has been applied to produce a sub-optimal training set to train a SVM. To achieve this aim, NSGA-II considers as solutions single instances of an original training set. As a consequence, the application of genetic operators results in optimal solutions that may not belong to the original training set. Therefore, in [31], even if the addressed goal is the training sample selection, it does not face TSS problem since TSS deals with selection of instances from the original training set without producing new instances for SVM training. Finally, in [32], an approach based on multi-objective optimization algorithms is proposed for selecting a combination of instance sets and SVM hyper-parameters. Therefore, all developed Pareto-based multi-objective optimization algorithms applied for SVM in literature does not address the TSS problem, and, as a consequence, they are out of the scope of this work.

Among the approaches belonging to the second category, the most competitive ones are (i) KMSVM [33] which is based on k-means clustering; (ii) Linear

Fuzzy Support Vector Machine (LFSVM) [34] which is a geometry-based algorithm based on the idea of class centroid; (iii) Pre-Selection sample based on Class Centroid (PSCC) [35] and Vector Projection Support Vector Machine (VPSVM) [36] which are also geometry-based algorithms making use of the centroid of class for cutting down the training set; (iv) Shell Extraction (SE) [37] which is a geometry-based method that extracts the useless instances from the training set to preserve the maximum of support vectors. However, during an extensive experimental session conducted in [37], SE has emerged to be the best algorithm belonging to the second category by establishing itself as the state-of-the-art for TSS techniques based on non-evolutionary approaches. SE achieves this result thanks to its capability of extracting the instances which are not likely to be support vectors. In order to achieve this goal, SE considers a round area, referred as Reduction Sphere (RS), whose instances will be deleted. The number of deleted instances depends on the radius of RS controlled by two user parameters λ and δ. Differently from our TSS technique, SE requires the user to adjust the reduction intensity by setting the parameter ξ.

Starting from this analysis, in our experimentation, the state-of-the-art algorithms, GGA [23] (evolutionary) and SE [37] (non-evolutionary), will be compared with our TSS technique based on a Pareto-based multi-objective optimization approach to prove the high performance of the proposed approach.

3. The Pareto-based evolutionary approach to Training Set Selection

Recently, TSS has been formulated as an optimization problem based on the following objectives: 1) maximize the accuracy, i.e., the number of the instances correctly classified by a classifier, when using the reduced training data and 2) maximize the amount of reduction achieved with respect to the original training set. These two objectives are conflicting because, typically, increasing the amount of reduction leads to a decreasing in the accuracy and vice versa. In case of conflicting objectives, there does not exist a single solution that simultaneously optimizes each objective. Hence, this observation has arisen

our proposal to apply a Pareto-based multi-objective optimization algorithm such as PESA-II to implement the ParetoTSS algorithm. This new algorithm is capable of producing a set of viable alternative solutions representing the best possible trade-offs between the antagonistic objectives characterizing the problem.

Before explaining the details of the proposed approach in the next section, let us introduce the Pareto-based multi-objective optimization algorithms and our formal definition of the TSS problem as a Pareto optimization problem.

3.1. Pareto-based multi-objective optimization principles

In general, a Pareto-based multi-objective optimization problem can be described as a vector function f that maps a tuple of m parameters (decision variables) to a tuple of n objectives. Formally (for a maximization problem):

$$\max \mathbf{y} = f(\mathbf{x}) = (f_1(\mathbf{x}), f_2(\mathbf{x}), \ldots, f_n(\mathbf{x}))$$

$$\text{subject to } \mathbf{x} = (x_1, x_2, \ldots, x_m) \in X$$

$$\mathbf{y} = (y_1, y_2, \ldots, y_n) \in Y$$

where $\mathbf{x}$ is called the *decision vector*, X is the *parameter space*, $\mathbf{y}$ is the *objective vector* and Y is the *objective space*. The set of all decision vectors for which the corresponding objective vectors cannot be improved in any dimension without degrading in another is known as *Pareto-optimal front*. Mathematically, the concept of Pareto optimality is the following: let us consider a maximization problem and two decision vectors $\mathbf{a}, \mathbf{b} \in X$. Then, $\mathbf{a}$ is said to dominate $\mathbf{b}$ (also written as $\mathbf{a} \succ \mathbf{b}$ [38]) iff

$$\forall i \in \{1, 2, \ldots, n\} : f_i(\mathbf{a}) \geq f_i(\mathbf{b}) \wedge$$

$$\exists j \in \{1, 2, \ldots, n\} : f_j(\mathbf{a}) > f_j(\mathbf{b})$$

Additionally, $\mathbf{a}$ is said to weakly dominate $\mathbf{b}$ (also written as $\mathbf{a} \succeq \mathbf{b}$) iff $\mathbf{a} \succ \mathbf{b}$ or $f(\mathbf{a}) = f(\mathbf{b})$. All decision vectors which are not dominated by any other decision vector of a given set are called *non-dominated* [39]. Fig. 1 graphically shows the Pareto-based multi-objective optimization principles.

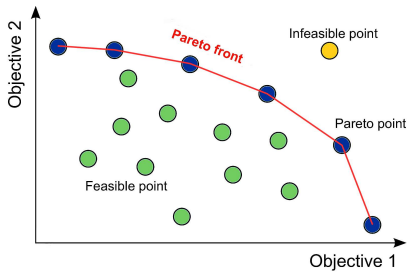


Figure 1: Example of Pareto front for a problem with two objectives to be maximized. Green points are feasible solutions. Blue points are non-dominated solutions that form the Pareto front. The points in the higher right area represent solutions that are desired but not feasible (yellow).

3.2. Training Set Selection as a Pareto optimization problem

By taking into account the two objectives characterizing the TSS problem (accuracy and reduction) and the Pareto-based multi-objective optimization principles, it is possible to formulate TSS as a Pareto optimization problem as in Definition 1.

Definition 1 (TSS problem). *Let TR be the original training set and Λ_{TR} be the set of all possible subsets S of TR, the TSS problem can be formulated as below:*

$$max\ \mathbf{y} = F(S) = [f_1(S), f_2(S)]\ with\ S \in \Lambda_{TR} \tag{1}$$

where $f_1 : \Lambda_{TR} \rightarrow \mathbb{R}$ is the first objective defined as follows:

$$f_1(S) = acc(TR, S)\ with\ S \in \Lambda_{TR} \tag{2}$$

where acc is a function that computes the classification accuracy in percentage obtained by a SVM classifier by considering TR as testing set and S as training set and $f_2 : A_{TR} \rightarrow \mathbb{R}$ is the second objective defined as follows:

$$f_2(S) = red(TR, S) = \frac{|TR| - |S|}{|TR|} \cdot 100\ with\ S \in \Lambda_{TR} \tag{3}$$

where red is a function that computes the reduction rate in percentage of the set S with respect to the original set TR.

4. The algorithm ParetoTSS

The ParetoTSS algorithm is mainly based on PESA-II, a Pareto-based approach proposed by Corne et al. [14] in 2001. It is one of the most popular Pareto-based approaches and it has been widely applied in many fields [40][41] since then. Its main characteristics are: i) the use of two populations, called *internal population* and *external population*; ii) the use of hyper-grid division of phenotype space which allows to maintain diversity in the algorithm; iii) the use of a mating selection process called *region-based*; iv) the use of the environmental selection process depending on the region density. In order to apply PESA-II for solving the TSS problem, we have defined an adequate solution encoding. Moreover, we have defined recombination and mutation operators suitable for TSS problem and a restarting mechanism for correcting eventual solutions which become unfeasible during the PESA-II evolution. Like all Pareto-based multi-objective optimization algorithms, PESA-II produces a set of non-dominated solutions, each one representing the trade-off between the considered antagonistic objectives. However, in order to take advantage of PESA-II application in TSS, the proposed algorithm, ParetoTSS, includes a method for extracting a single solution from the Pareto front. Hereafter, the general features of the PESA-II algorithm are recalled in Subsection 4.1. Then, the solution encoding for TSS problem is described in Subsection 4.2, whereas, the recombination and mutation operators and the restarting mechanism are described in Subsection 4.3. Subsection 4.4 presents the method to select a single solution from the produced Pareto front. Subsection 4.5 presents a description of the whole behavior of the proposed approach, ParetoTSS. At conclusion, Subsection 4.6 discusses the time complexity of ParetoTSS.

4.1. PESA-II algorithm

During its evolution, PESA-II maintains two populations: an internal population IP of fixed size, and an external population EP of non-fixed but limited size [42]. The external population is actually the archive that stores the current

approximation to the Pareto front, whereas, the internal population stores the new solutions generated by variation operations and vying for incorporation into the archive.

The solutions in the archive are stored in *hyper-boxes*, obtained by means of a hyper-grid which divides the objective space. The number of solutions within a hyper-box is referred to as *density*, and it is used to choose solutions both in the mating selection and in the environmental selection.

The mating selection process consists of choosing the solutions to be undergone to evolutionary operations. In PESA-II, the mating selection process is implemented in a region-based manner rather than in an individual-based manner (i.e. the typical selection mechanism of the other Pareto-based multi-objective evolutionary algorithms including PESA [43], the predecessor of PESA-II). Precisely, a hyper-box is first selected and then the solution to be undergone to evolutionary operations is randomly chosen from the selected hyper-box. Thus, highly crowded hyper-boxes do not contribute more solutions than less crowded ones [15]. This is useful to encourage solutions to cover the whole objective space, rather than bunch together in one region [42].

As for the environmental selection process, it consists into updating the archive, that is choosing the solutions that will compose the archive set in the next generation. In PESA-II, during the environmental selection process, the solutions in the internal population are inserted into the archive set one by one, thus the grid environment updated step by step [15]. A solution is inserted in the archive if it is non-dominated within the internal population, and it is non-dominated by any current member of the archive. Once a solution has entered the archive, corresponding adjustments of the archive and grid environment are implemented. Firstly, the members in the archive which the solution dominates are removed to ensure that only non-dominated individuals exist in the archive. Secondly, the grid environment is checked to see whether its boundaries have changed with the addition or removal of solutions in the archive. Finally, if the addition of a solution renders the archive overfull, the replacement is performed according to the density of hyper-boxes. Indeed, an arbitrary individual in the

most crowded hyper-box will be removed [15].

An example of environmental selection process is given in Fig. 2. The individuals A-E in Fig. 2(a) are the candidates to be archived contained in IP. Fig. 2(b) shows the archiving process of Z. Z is non-dominated in IP and non-dominated by the current members of EP, hence it deserves to enter the archive. Initially, since no members of the archive are dominated by Z, no individuals are removed. With the insertion of Z, the grid environment boundaries are updated. However, the archive size is five, so an individual must be removed. The most crowded hyper-box is that containing B and C, so one of these individuals is removed arbitrarily (in this case C). Fig. 2(c) shows the archiving process of Y. Y is non-dominated in IP and non-dominated by the current members of EP, hence it deserves to enter the archive. Once Y has entered the archive, E is removed as it is dominated by Y. The insertion of Y does not require adjustments of the grid environment boundaries. Fig. 2(d) shows the archiving process of X. X is non-dominated in IP and non-dominated by the current members of EP, hence it deserves to enter the archive. Once X has entered the archive, A is removed as it is dominated by X. The removal of A leads to adjustments of the grid environment boundaries.

4.2. The solution encoding

As described in Appendix A, given a training set TR of size n, the solution to the TSS problem is to detect the smallest set of training instances which allows to predict the class label of a new instance with the same accuracy provided by the original training set TR. Therefore, the search space associated with the instance selection of TR is composed of all subsets of the set TR. Hence, a solution of the TSS problem should represent a subset of TR. We encode a solution as a vector of n bits (one for each instance in the training set TR). A bit is set to 1 when the corresponding training instance is included in the subset of TR.

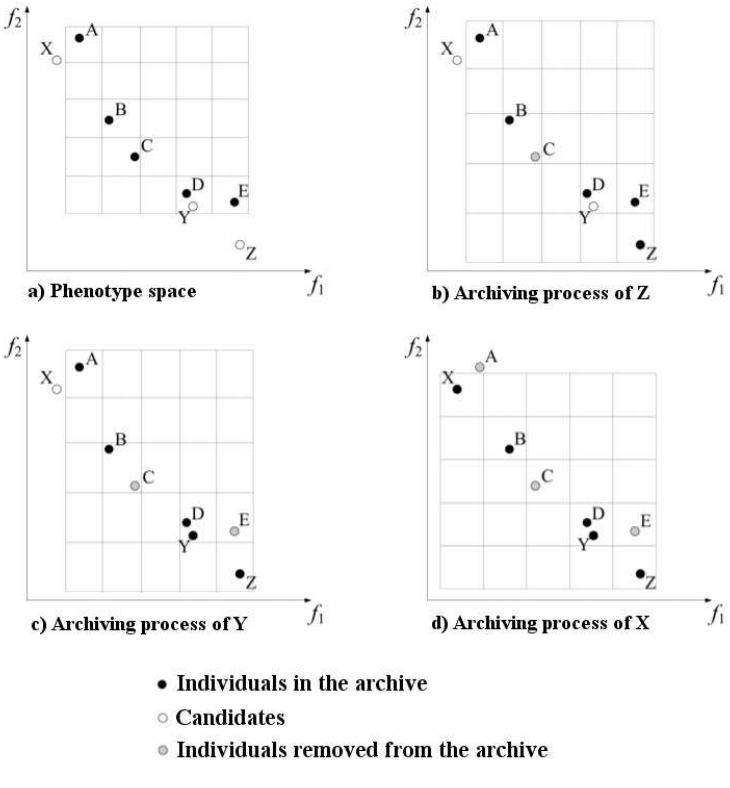


Figure 2: An example of the environmental selection in PESA-II [15].

4.3. Recombination, mutation and restarting

ParetoTSS uses recombination and mutation operators suitable for the binary encoding. In particular, it uses a modified version of the *Heuristic Uniform Crossover* (HUX) and a modified version of the *bit flip mutation*. In general, the HUX crossover works as follows. Once two parents are selected, the HUX crossover generates two sons by exchanging half of the bits that are different in the parents. The bit position to be exchanged is randomly determined [44]. Like all the crossover operators, HUX is applied with a certain rate p_c. In order to improve the reduction rate, we have implemented a modified version of HUX. In detail, if the selected bit position is on, HUX switches it off, otherwise, it switches it on according to a probability referred to as *probability of inclusion* (denoted as $p_{inclusion}$).

As for the mutation operator, the bit flip mutation acts independently on each bit in a solution and changes the value of the bit (from 0 to 1 and vice versa) with probability p_m. However, in order to improve the reduction rate, ParetoTSS uses also a modified version of the bit flip mutation. In particular, we consider two different mutation probabilities: 1) the probability of a 0 to 1 mutation (denoted as p_{m1}) and 2) the probability of a 1 to 0 mutation (denoted as p_{m0}). The probability p_{m1} will be set smaller than the probability p_{m0} in order to force the algorithm to obtain higher reduction rates.

It is worth noting that this design choice may lead ParetoTSS to produce training sets containing no instances (empty training sets), and for which no information of accuracy and reduction rate can be obtained. In these cases, the proposed algorithm uses a *Restart* operator to randomly add instances from the original training set to the empty training set with a probability p_r, and make this again a feasible solution for the TSS problem.

4.4. Selecting a solution from the Pareto front

PESA-II, like all Pareto-based approaches, returns a set of non-dominated solutions found during the search, which is expected to be an approximation to the true Pareto optimal set. In our case, each of these non-dominated solutions

represents a set of instances to be used as a reduced data set for the SVM classifier. Theoretically, none of these solutions is better/worse than any other. They are equally acceptable solutions. However, having a set of solutions instead of a single one does not make sense for the TSS problem. Therefore, ParetoTSS includes a decision-making mechanism to select a single solution from the non-dominated set. The decision-making mechanism which we use is based on a sum model. It consists of choosing the solution characterized by the higher sum of the objective values (being TSS defined as a maximization problem). Formally, let us consider that, during its evolution, ParetoTSS produces a Pareto front composed of m solutions, where each one represents a possible reduced training set S; then, the best solution returned by the algorithm is the one that satisfies the following expression:

$$\psi^* = \max_i(\psi_i) \text{ for } i = 1, 2, \ldots, m \tag{4}$$

where ψ^* is the score of the best solution and ψ_i is the score of the i-th reduced training set computed by the following formula:

$$\psi_i = f_{1i} + f_{2i} \text{ with } i \in \{1, 2, \ldots, m\} \tag{5}$$

where f_{1i} and f_{2i} are the values, respectively, of the first and second objective related to the i-th reduced training set. Fig. 3 shows a plot displaying a Pareto front composed of six solutions (A, B, C, D, E and F) and the corresponding solution scores. In details, the Pareto front is displayed in the two-dimensional plane (f_1, f_2), whereas, the scores ψ_i (with $i = 1, ..., 6$) of solutions are displayed on the third dimension. The best solution selected by the proposed decision-making mechanism is E, i.e. the one with the highest score.

4.5. Pseudo-code of ParetoTSS

Algorithm 1 is the pseudo-code of ParetoTSS. The algorithm takes in inputs the original training set TR, the size of the PESA-II population N, the size of the PESA-II archive $\overline{N}$, the crossover rate p_c,the probability of inclusion $p_{inclusion}$, the mutation probabilities p_{m0} and p_{m1}, the restart probability p_r and the

Algorithm 1 Pseudo-code of the proposed algorithm for TSS

procedure ParetoTSS(TR, N, $\overline{N}$, p_c, $p_{inclusion}$, p_{m0}, p_{m1}, p_r, t)
  $g \leftarrow 0$;
  $IP \leftarrow \emptyset$;
  $EP \leftarrow \emptyset$;
  while ($i < N$) **do** ▷ Generate randomly an initial population IP of N training sets
    s_i= initialize_training_set(i);
    evaluate(TR, s_i, p_r);
    $IP \leftarrow IP \cup \{s_i\}$;
    $i = i + 1$;
  end while
  $i \leftarrow 0$;
  while ($i < N$) **do** ▷ Update the archive EP with training sets contained in IP
    update_archive(EP, $\overline{N}$, IP_i);
    $i = i + 1$;
  end while
  while (t is not satisfied) **do** ▷ The algorithm evolves until termination criteria t are reached
    $IP \leftarrow \emptyset$;
    $i \leftarrow 0$;
    while ($i < N$) **do** ▷ Generate a new IP by means of evolutionary operators
      p_1,p_2= mating_selection(EP);
      if rand(0,1) $< p_c$ **then**
        p_1,p_2= modified_HUXcrossover(p_1,p_2,$p_{inclusion}$);
      end if
      p_1 = modified_BitFlipMutation(p_1, p_{m0}, p_{m1});
      evaluate(TR, p_1, p_r);
      $IP \leftarrow IP \cup \{p_1\}$;
      $i = i + 1$;
    end while
    $i \leftarrow 0$;
    while ($i < N$) **do** ▷ Update the archive EP with training sets contained in IP
      update_archive(EP, $\overline{N}$, IP_i);
      $i = i + 1$;
    end while
    $g \leftarrow g + 1$;
  end while
  $S_{best} \leftarrow$ decision_making_procedure(EP);
  return S_{best}; ▷ the best training set is S_{best}
end procedure

procedure Evaluate(TR, s, p_r) ▷ Evaluate a training set
  if $s == \emptyset$ **then** ▷ Apply restart operator if the training set s is empty
    $s \leftarrow$ restart_operator(s, p_r);
  end if
  $obj1 \leftarrow$ acc(TR, s); ▷ The first objective is set by following Eq. 2
  $obj2 \leftarrow$ red(TR, s); ▷ The second objective is set by following Eq. 3
end procedure

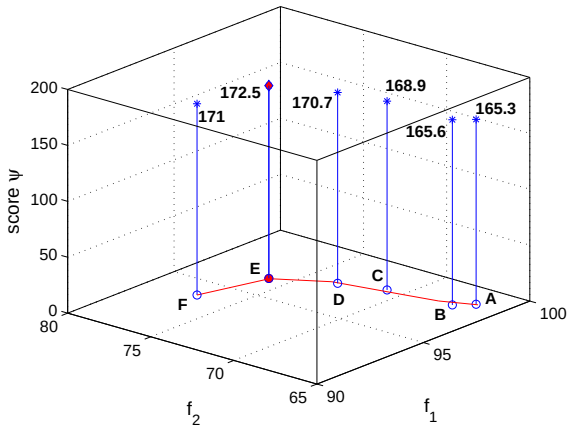


Figure 3: An example to show how ParetoTSS decision making mechanism works. The solution E is characterized by the highest score, and, as a consequence, it is set as the best solution to the TSS problem.

termination criteria t. The algorithm generates and evaluates each solution of an initial internal population (IP) of the training sets randomly selected. Then, it initializes the external population (EP) to the empty set. Successively, the algorithm incorporates the non-dominated training sets of IP into EP. At this point, the algorithm evolution progresses per iterations. In each iteration, it deletes the current content of IP, and repeats the following steps until N (the size of IP) new training sets have been generated: 1) the algorithm selects two training sets from EP and produces a single training set via crossover, then it mutates this training set just built up; 2) the algorithm evaluates each training sets of IP and, then, it incorporates the non-dominated training sets of IP into EP. At the end of each iteration, if the termination criteria are not reached, the new created archive is used to continue the algorithm evolution, otherwise the algorithm stops and returns as results the training sets contained in EP. These training sets are undergone to the proposed decision-making mechanism to select a single training set to be used for the classification task.

4.6. Time complexity of ParetoTSS

As all Pareto-based multi-objective optimization algorithms, the time complexity of our approach is dominated by the complexity of the fitness function and the complexity of the procedure aimed at selecting the non-dominated set of solutions. In our approach, the fitness function is represented by the function named EVALUATE. This function takes as input a candidate solution (a reduced training dataset) and compute the objectives described in Section 3.2. In particular, one of these objectives is the accuracy obtained by the SVM when it is trained by means of the reduced dataset. As a consequence, the time complexity of the EVALUATE function is strongly depending on the time complexity of the SVM, which is known to be $O(M^2)$ where M is the number of training instances. Hence, for each generation, the time complexity related to the EVALUATE function is $O(NM^2)$ where N is the number of solutions (reduced training datasets). Moreover, the procedure used by PESA-II to select the non-dominated set at each generation has a complexity $O(mN)$ [45], where m is the number of objectives. Consequently, for each generation, the time complexity of our approach is $O(NM^2 + mN)$.

5. Experimental study

The improvements provided by ParetoTSS have been evaluated by means of a set of experiments.

5.1. Experimental setup

The experiments involve thirty-six well-known datasets from the UCI Machine Learning Database Repository including thirty-four datasets with a low/medium size and two with a larger size (namely banana and phoneme). The main features of all datasets are summarized in Table 1. For each dataset, the name, the number of instances, the number of attributes and the number of classes are given. By analyzing the features of the exploited datasets, it is possible to note that there are not big datasets used in the experiments. This is due to the fact

that managing big data needs specialized distributed techniques for TSS that are beyond the scope of this paper.

In this work, we use John Platt's Sequential Minimal Optimization (SMO) algorithm [46] to construct the SVM classifier because it is one of the fastest and most regularly used optimization algorithms in the context of SVMs. Precisely, the implementation of SMO used is contained in WEKA [47]. As for the kernel function, we use the Radial Basis Function (RBF) kernel with the parameter $\gamma = \frac{1}{2\sigma^2}$ equal to 0.01 and the cost parameter C set to 1. We use the pairwise coupling setting to handle multi-class problems. These parameters represent a default SVM configuration already used in other similar works [23]. The choice not to tune these parameters is due to our goal of highlighting the benefits yielded by the proposed TSS algorithm in terms of itself and without considering any improvements in classification accuracy due to a given configuration of the SVM used in its fitness function.

The configuration setting of ParetoTSS is the same in all experiments and reported in Table 2. These parameters were empirically chosen. The used implementation is extracted by jMetal library [48]. The comparison between ParetoTSS and state-of-the-art approaches involves two methods: GGA [23] and SE [37]. These methods represent the state-of-the-art: GGA is the best TSS technique for SVM known so far among the evolutionary approaches[4] and SE is the best TSS performer among non-evolutionary ones. The implementation of GGA is that provided by the KEEL Software [49] where the exploitation of the KNN classifier has been replaced by the SMO contained in WEKA. As for the SE approach, the code made available by its authors has been used. The parameters of all compared TSS methods are reported in Table 2. For the state-of-the-art approaches, the used parameter settings are those specified by their respective authors as we assume that their choices are optimally made. However, it is worth noting that an optimal value for the reduction rate, considered as a parameter in SE (see Section 2), is not specified by the authors. Therefore, SE has been executed with a range of reduction rates (from 0.01 to 0.99) and the results characterized by the best trade-off between accuracy and

Table 1: Datasets characteristics

Dataset name	**Number of instances**	**Number of attributes**	**Number of classes**
appendicitis	106	7	2
australian	690	14	2
automobile	159	25	6
balance	625	4	3
banana	5,300	2	2
bands	365	19	2
breast	277	9	2
bupa	345	6	2
cleveland	297	13	5
crx	653	15	2
dermatology	358	34	6
ecoli	336	7	8
german	1,000	20	2
glass	214	9	7
haberman	306	3	2
heart	270	13	2
hepatitis	80	19	2
housevotes	232	16	2
iris	150	4	3
led7digit	500	7	10
lymphography	148	18	4
mammographic	830	5	2
monk-2	432	6	2
newthyroid	215	5	3
phoneme	5,404	5	2
pima	768	8	2
saheart	462	9	2
sonar	208	60	2
spectfheart	267	44	2
tae	151	5	3
tic-tac-toe	958	9	2
vehicle	846	18	4
vowel	990	13	11
wine	178	13	3
wisconsin	683	9	2
zoo	101	16	7

Table 2: Parameters used by all compared TSS methods

Algorithm	Parameters
ParetoTSS	$Evaluations$ = 10,000, N = 50, $\overline{N}$ = 40, p_c = 1.0, $p_{inclusion}$ = 0.15, p_{m0} = 0.05, p_{m1}=0.001, p_r = 0.15
GGA	$Population\ size$ = 51, $Evaluations$ = 10,000, p_c = 0.6, p_{m0} = 0.01, p_{m1}=0.001, $Equilibrate\ factor$ α = 0.5
SE	$\lambda = 0.8$, $\delta = 0.01$

reduction have been compared for each dataset. Besides, in order to perform a fair comparison, ParetoTSS and GGA have been run by using the same number of fitness evaluations as termination criterion. This value (denoted as simply *Evaluations* in Table 2) is set to 10,000 since it is reasonable in terms of the computational complexity and, at the same time, it allows achieving good performance. However, it is worth noting that the increasing of the number of the fitness evaluations can lead to an improvement of both evolutionary algorithms, but the study of the convergence is out of the scope of this work. All algorithms are run on a computer HP, Intel Core i7 4.0GHz, 1 GB RAM. All datasets and implementation codes are available to enable a full reproducibility of the experiments[1].

5.2. Methods and Metrics for the experimental evaluation

The analysis of the performance of the proposed approach, ParetoTSS, is carried out though a validation procedure that takes care of the interpolation as well as the extrapolation. It consists in dividing a dataset into two parts with a ratio 80-20%. The first part of data represents the training set and it is used to perform the classical tenfold cross-validation. Briefly, the tenfold cross-validation separates data in 10 folds and, every time, a single fold is used as test data and the remaining ones are used as training data. Due to the non-deterministic nature of the our proposal, in each one of the ten steps of

[1] `http://quasar.unina.it/research/publication_extras/journals/KBS2017/publishedKBSworkspace.zip`

the cross-validation, ParetoTSS is executed three times. Therefore, the average result over the three trials and the ten steps are considered in the evaluation.

The second part of the dataset (the 20% of the original dataset) is used as the validation set. Each one of the SVM models obtained by the ten steps of the cross-validation procedure is tested on the validation set and the average result over the ten steps is considered. The overall validation procedure allows us to evaluate the performance of the proposed approach doubly. Firstly, the evaluation of the performance of ParetoTSS on the ten test-folds of the cross-validation ensures that the proposed method works with different combinations of the training set and takes care of the interpolation. Secondly, the evaluation of the performance of ParetoTSS on the validation set makes sure that even the extrapolation is taken care of since the validation set is never seen by the proposed model at all. This validation procedure has been chosen because it allows to give an evaluation of performance of the proposed method as realistic as possible.

The performance of ParetoTSS is evaluated through two measures: the *accuracy* (ACC) and the *reduction rate* (RED). In general, the first measure represents the number of successful hits relative to the total number of classifications performed by the SVM. As for ParetoTSS, it represents the accuracy of the SVM trained using the reduced set computed by ParetoTSS. This measure has been chosen because it is the most commonly used metric for assessing the performance of classifiers [50]. The reduction rate, instead, represents the number of instances in the training set selected by ParetoTSS to be removed relative to the total number of instances contained in the original training set. It is a mandatory measure to be considered during the evaluation of TSS methods.

Apart from the evaluation of the performance of the ParetoTSS, the experiments aim at comparing the proposed approach with SVM without preprocessing and with other TSS techniques representing the state-of-the-art. Both comparisons have been carried out by performing the ten-fold cross-validation discussed above. This choice has been made to perform comparisons that are not depending on the data contained in the training set. Indeed, as described,

the cross-validation allows to run each method with different combinations of training and test data sets. As for the evaluation measures, the accuracy is used during both comparisons; instead, the reduction rate is used only in the comparison between ParetoTSS and the state-of-the-art approaches because it is not computable for the SVM without pre-processing. Apart from the accuracy and the reduction rate, the comparison between ParetoTSS and the state-of-the-art approaches involves also a trade-off measure denoted as *Accuracy-Reduction balance* (*ACC*RED*). It represents the product of accuracy and reduction rate. As described in [6], this product is a fair estimator of how good a TSS method is considering a trade-off of reduction and success rate of classification, and as a consequence, it is useful for comparing TSS methods overall.

Finally, to verify the significance of the obtained results, a set of non-parametric tests has been used as recommended in [51]. In particular, we use the well-known Wilcoxon's signed rank [52] test for pairwise comparisons and the Friedman Aligned Ranks test [53] followed by the Finner's test [54] for the multiple comparisons. Among the several statistical tests useful for multiple comparisons, we have chosen the Friedman Aligned Ranks test as it is more suitable when comparisons involve a low number of algorithms (no more than 4 or 5) [55]. As for the Finner's test, the use is recommended in [55] for its good trade-off between performance and ease of understanding. Details about the exploited statistical tests can be found in [55][56].

In the next sections, details about the results of the performed experiments are discussed.

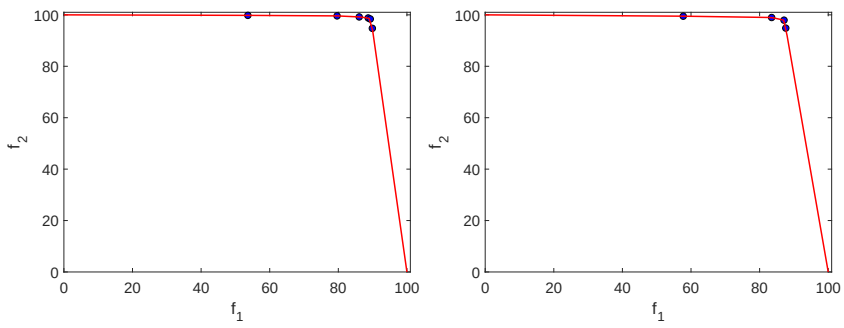
5.3. Experimental session I: evaluating ParetoTSS performance

When ParetoTSS is run, it computes a sub-optimal Pareto front for each one of the considered datasets. Figs. 4(a)-(f) shows Pareto fronts obtained by a single run on some of the considered datasets (Pareto fronts for all datasets

can be downloaded online[2]). Then, by means of the proposed decision-making mechanism, ParetoTSS selects a single solution from the obtained Pareto front. This solution represents a reduced training set for the SVM. In order to evaluate the quality of the obtained reduced training set, and, as a consequence, the performance of ParetoTSS, we have performed the validation procedure described in the previous section. Table 3 shows the results of the validation procedure in terms of the accuracy and the reduction rate obtained by ParetoTSS. In particular, we denote with *accuracy on training*, the average of the accuracy values obtained by the SVM trained with the reduced set computed by ParetoTSS and used to classify the training sets related to the performed ten-fold cross-validation. Instead, the *accuracy on testing* represents the average of the accuracy values obtained by the SVM trained with the reduced set computed by ParetoTSS and used to classify the testing sets related to the performed ten-fold cross-validation. Finally, *accuracy on validation* represents the average of the accuracy values obtained by the SVM trained with the reduced set computed by ParetoTSS and used to classify the validation set.

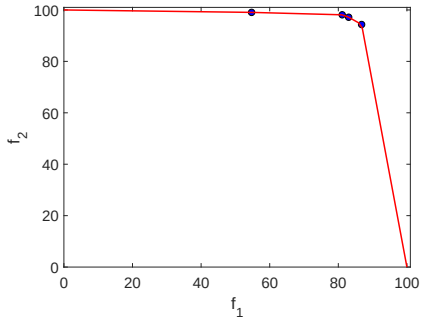
By analyzing Table 3, it is possible to make some considerations. Firstly, our proposal does not lead to the overfitting since there is a low difference (about the 4% on average) between the accuracy values on training set and those on testing data. Secondly, our approach for TSS works in a good way also on data never seen before. Indeed, the difference between the accuracy values on testing set and those on validation set is not significant at the 99% confidence level (after applying the Wilcoxon's signed rank test). Finally, the proposed approach is characterized by a very high reduction rate value, more than 97% on average.

[2] `http://quasar.unina.it/research/publication_extras/journals/KBS2017/paretoFrontPlots.zip`

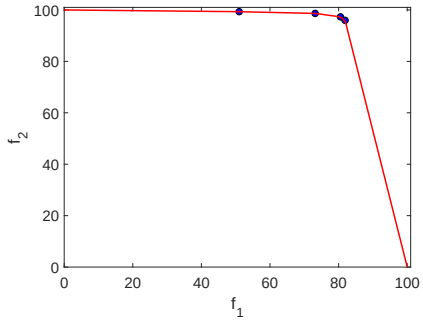


(a) Dataset australian

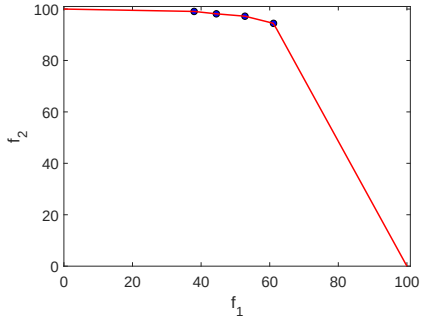
(b) Dataset heart



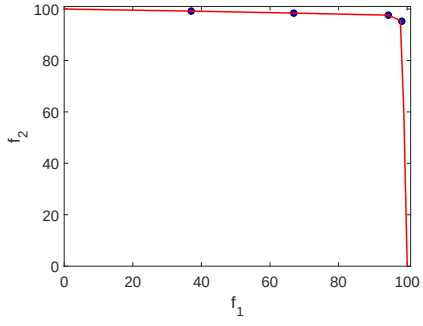
(c) Dataset lymphography



(d) Dataset sonar



(e) Dataset tae



(f) Dataset wine

Figure 4: Examples of Pareto Fronts obtained by ParetoTSS for some of the considered datasets

Table 3: Performance of the proposed approach. The accuracy and the reduction rate are reported in percentage.

	Accuracy on training	*Accuracy on testing*	*Accuracy on validation*	*Reduction rate*
appendicitis	89.241	87.083	91.364	97.354
australian	88.298	85.811	85.29	98.235
automobile	54.999	43.077	53.125	96.035
balance	88.17	87.067	88.747	99.104
banana	62.416	61.95	62.638	99.749
bands	69.876	65.126	61.644	98.364
breast	77.728	73.814	77.917	98.576
bupa	70.746	68.311	70.097	98.43
cleveland	62.339	54.42	57.556	98.453
crx	88.023	85.379	82.901	98.07
dermatology	94.833	92.41	90.509	97.682
ecoli	78.453	72.953	78.873	97.637
german	76.069	72.292	70.417	98.657
glass	60.621	53.693	59.07	97.336
haberman	76.032	72.122	70.968	99.18
heart	87.055	84.286	80.679	97.634
hepatitis	89.515	83.175	82.292	95.434
housevotes	96.957	93.304	90.638	97.417
iris	97.006	95.556	94.111	96.389
led7digit	72.583	69.25	67.833	97.324
lymphography	84.086	75.051	85.444	95.701
mammographic	85.196	84.233	84.779	99.297
monk-2	87.912	86.675	85.594	98.594
newthyroid	95.586	94.978	93.721	98.062
phoneme	77.93	77.193	79.019	99.813
pima	78.025	75.805	73.636	99.294
saheart	73.742	69.269	71.434	98.695
sonar	80.811	73.664	74.921	95.36
spectfheart	80.299	79.098	77.654	99.339
tae	56.512	50	49.785	95.833
tic-tac-toe	73.934	70.926	68.472	97.776
vehicle	56.355	52.703	59.137	99.069
vowel	33.974	27.91	30.135	98.719
wine	96.662	92.508	95.093	96.4
wisconsin	97.741	96.819	96.569	99.566
zoo	92.685	90	90.476	91.62
Average	78.678	74.942	75.904	97.783

5.4. Experimental session II: evaluating ParetoTSS improvements on SVM classification

In this section, we show improvements to the SVM classification provided by the application of ParetoTSS as TSS technique. In detail, as already described, the benefits of our approach are: 1) the computational complexity of SVM classification is decreased thanks to the reduction of the instances in the training data and 2) the accuracy of SVM can be improved thanks to the removal of wrong information in training data. To verify the first benefit, we compare the time of classification of a new instance when SVM is trained on the reduced training set computed by ParetoTSS (denoted simply ParetoTSS from now on) and when SVM is trained with the original dataset (denoted as simply SVM from now on). To verify the second benefit, instead, we compare the accuracy values between ParetoTSS and SVM. Table 4 presents the results obtained by the SVM and ParetoTSS over all the considered datasets in terms of both time of classification and accuracy. By analysing this table, one can to note that ParetoTSS improves the accuracy of SVM in 31 out of 36 datasets and outperforms SVM in all datasets in terms of the time of classification. Moreover, ParetoTSS is characterized by the highest average accuracy with a relative improvement on SVM of about 15% and by the lowest average time of the classification with a relative improvement on SVM of about 97%. In order to verify the significance of these results, we performed two statistical experimental sessions using the Wilcoxon's signed rank test by considering as samples, in the first one, the accuracy values and, in the second one, the times of the classification reported in Table 4. The p-values computed by the Wilcoxon's test are equal to $2.3290 \cdot 10^{-6}$ and $8.766 \cdot 10^{-8}$, respectively. Therefore, it is possible to state that ParetoTSS statistically improves SVM classification time and accuracy at the 99% confidence level.

Table 4: Comparison between SVM trained on the reduced training data produced by ParetoTSS (denoted simply as ParetoTSS) and SVM trained on the original training data (denoted simply as SVM). The accuracy is in percentage and the time of the classification is reported in seconds.

	ParetoTSS		SVM	
Dataset	Accuracy	Classification time	Accuracy	Classification time
appendicitis	87.083	0.00833	78.194	0.02222
australian	85.811	0.00302	85.87	0.07974
automobile	43.077	0.0265	31.731	0.35385
balance	87.067	0.00267	86.2	0.07
banana	61.95	0.00189	55.071	0.33184
bands	65.126	0.00226	65.08	0.04793
breast	73.814	0.00758	71.522	0.08142
bupa	68.311	0.00123	56.548	0.03267
cleveland	54.42	0	52.88	0.09312
crx	85.379	0.00893	86.963	0.16665
dermatology	92.41	0.01285	81.441	0.35665
ecoli	72.953	0.00494	42.536	0.09316
german	72.292	0.00917	69.75	0.37
glass	53.693	0	37.516	0.06438
haberman	72.122	0	74.533	0.0245
heart	84.286	0.00317	83.853	0.03247
hepatitis	83.175	0	84.762	0.03333
housevotes	93.304	0.00185	92.398	0.02164
iris	95.556	0.00556	56.667	0.05
led7digit	69.25	0.0125	55.75	0.4425
lymphography	75.051	0	74.621	0.04318
mammographic	84.233	0.001	80.265	0.05421
monk-2	86.675	0.00193	80.891	0.04336
newthyroid	94.978	0	70.948	0.00588
phoneme	77.193	0.0017	70.923	0.30048
pima	75.805	0.00055	65.299	0.06354
saheart	69.269	0.0027	64.767	0.04339
sonar	73.664	0.01029	66.25	0.09596
spectfheart	79.098	0.00317	79.892	0.05152
tae	50	0.00278	30	0.025
tic-tac-toe	70.926	0.00478	66.965	0.18667
vehicle	52.703	0.00693	36.396	0.48516
vowel	27.91	0.01684	7.195	1.51389
wine	92.508	0.0119	38.667	0.04238
wisconsin	96.819	0.00122	95.96	0.03667
zoo	90	0.00833	63.75	0.075
Average	74.942	0.0049	65.057	0.1621

5.5. Experimental session III: comparing ParetoTSS with state-of-the-art approaches

In this section, we show the results of the comparison of the proposed approach with GGA and SE. The comparison is carried out in terms of accuracy, reduction rate and accuracy-reduction balance. For sake of completeness, also the running time, i.e, the average elapsed time in seconds to complete a run of a TSS method, is considered in the comparison. However, it is worth noting that the first three measures are the most relevant to evaluate how good is a TSS algorithm. Indeed, the training set selection is a task performed a single time on a training set, therefore, the running time of TSS algorithms has a limited effect. Instead, the classification task using the reduced training set is an operation repeated indefinitely. Hence, the accuracy, the reduction rate and the accuracy-reduction balance are measures more relevant since they directly affect the SVM classification in terms of quality and time complexity.

Table 5 presents the results obtained by the compared TSS methods over the considered datasets for the first three metrics, whereas, Table 6 reports the running time values.

By analyzing the results reported in Table 5 related to the accuracy, ParetoTSS outperforms GGA in 26 out of 36 datasets and SE in 33 out of 36. As for the average accuracy, ParetoTSS is characterized by the highest average accuracy. Precisely, ParetoTSS improves the accuracy of GGA by 4.5% and the one of SE by 26.60%.

From this empirical analysis, ParetoTSS emerges to be the best performer in terms of accuracy. In order to statistically verify this result, we perform a statistical multiple comparison procedure composed of the Friedman Aligned Rank test and the Finner's test as described above. The samples used in the statistical analysis are the accuracy values obtained by all TSS techniques over all datasets. Table 7 shows the ranking obtained by all compared TSS techniques during the Aligned Friedman's test. As highlighted in bold, the algorithm with the best ranking (the lowest value) for the accuracy metric is ParetoTSS. The computed Aligned Friedman's statistic is 43.84. Since it is greater than the

Table 5: Results obtained by all TSS methods over all considered datasets. *RED* denotes reduction rate achieved, *ACC* denote the accuracy obtained in test data, *ACC*RED* correspond to the accuracy-reduction balance.

Dataset	ParetoTSS			GGA			SE		
	ACC (%)	RED (%)	ACC*RED	ACC (%)	RED (%)	ACC*RED	ACC (%)	RED (%)	ACC*RED
appendicitis	87.083	97.354	0.847788	85.556	97.091	0.830672	78.194	94.84	0.741592
australian	85.811	98.235	0.842964	86.773	88.125	0.764687	83.133	70.028	0.582164
automobile	43.077	96.035	0.41369	48.974	91.779	0.449478	25.962	93.786	0.243487
balance	87.067	99.104	0.862869	86.2	96.467	0.831546	76.2	95.578	0.728304
banana	61.95	99.749	0.617945	55.071	93.158	0.51303	55.071	99.921	0.550275
bands	65.126	98.364	0.640605	59.264	99.277	0.588355	65.08	98.097	0.638415
breast	73.814	98.576	0.727629	71.976	98.391	0.708179	71.522	97.034	0.694007
bupa	68.311	98.43	0.672385	69.193	98.712	0.683018	56.548	92.029	0.520406
cleveland	54.42	98.453	0.535781	53.768	97.234	0.522808	52.88	97.75	0.516902
crx	85.379	98.07	0.837312	83.135	90.379	0.751366	81.789	72.073	0.589478
dermatology	92.41	97.682	0.902679	65.69	89.398	0.587255	45.505	97.708	0.44462
ecoli	72.953	97.637	0.712291	72.308	96.808	0.699999	47.365	93.615	0.443407
german	72.292	98.657	0.713211	70	95.847	0.670929	69.75	99.722	0.695561
glass	53.693	97.336	0.522626	53.627	94.738	0.508051	36.928	95.127	0.351285
haberman	72.122	99.18	0.715306	73.75	99.089	0.730781	74.533	99.135	0.738883
heart	84.286	97.634	0.822918	81.926	96.346	0.789324	60.26	94.65	0.570361
hepatitis	83.175	95.434	0.793772	84.762	98.264	0.832905	84.762	97.402	0.825599
housevotes	93.304	97.417	0.90894	91.404	96.696	0.88384	71.959	90.45	0.650869
iris	95.556	96.389	0.921055	94.167	94.167	0.886742	50	77.87	0.38935
led7digit	69.25	97.324	0.673969	46.75	93.028	0.434906	34.5	21.833	0.075324
lymphography	75.051	95.701	0.718246	72.803	95.108	0.692415	51.97	96.516	0.501594
mammographic	84.233	99.297	0.836408	82.834	90.076	0.746136	62.775	92.888	0.583104
monk-2	86.675	98.594	0.854563	77.387	97.487	0.754423	58.563	99.388	0.582046
newthyroid	94.978	98.062	0.931373	92.516	98.062	0.90723	70.948	98.256	0.697107
phoneme	77.193	99.813	0.770486	70.923	93.459	0.662839	70.923	98.913	0.701521
pima	75.805	99.294	0.752698	73.096	98.335	0.71879	65.299	98.896	0.645781
saheart	69.269	98.695	0.68365	71.539	98.555	0.705053	64.767	96.929	0.62778
sonar	73.664	95.36	0.70246	69.191	95.045	0.657626	53.566	96.52	0.517019
spectfheart	79.098	99.339	0.785752	81.169	99.008	0.803638	79.892	99.322	0.793503
tae	50	95.833	0.479165	49.167	93.426	0.459348	40.833	90.833	0.370898
tic-tac-toe	70.926	97.776	0.693486	72.312	93.328	0.674873	66.965	39.676	0.26569
vehicle	52.703	99.069	0.522123	35.823	92.735	0.332205	28.244	82.89	0.234115
vowel	27.91	98.719	0.275525	19.959	93.715	0.187046	10.079	93.35	0.094087
wine	92.508	96.4	0.891777	96.476	94.366	0.910405	48.667	88.108	0.428795
wisconsin	96.819	99.566	0.963988	97.431	93.529	0.911262	94.323	70.9	0.66875
zoo	90	91.62	0.82458	85	90	0.765	41.25	90.417	0.37297
Average	74.942	97.783	0.733	71.72	95.034	0.682	59.195	89.235	0.530

Table 6: Running time in seconds for all considered TSS techniques over all datasets.

Dataset name	ParetoTSS	GGA	SE
appendicitis	18.037	33.073	0.01
australian	84.324	174.12	0.109
automobile	129.407	390.209	0.055
balance	92.491	219.445	0.084
banana	909.426	4099.035	0.043
bands	40.534	47.421	0.057
breast	36.073	52.559	0.038
bupa	35.838	48.022	0.047
cleveland	95.669	247.269	0.069
crx	146.955	280.827	0.077
dermatology	291.715	814.669	0.051
ecoli	206.011	481.249	0.056
german	209.865	588.593	0.078
glass	123.59	303.14	0.022
haberman	22.768	32.446	0.038
heart	37.346	61.065	0.05
hepatitis	20.384	5.355	0.019
housevotes	36.118	53.148	0.028
iris	64.949	123.407	0.011
led7digit	575.629	2166.615	0.005
lymphography	56.192	95.136	0.031
mammographic	57.688	160.615	0.071
monk-2	45.581	71.531	0.06
newthyroid	56.785	109.613	0.038
phoneme	1364.14	6336.406	0.73
pima	57.286	110.02	0.064
saheart	43.113	65.396	0.091
sonar	50.039	71.322	0.147
spectfheart	20.54	35.271	0.126
tae	67.454	119.538	0.011
tic-tac-toe	183.271	384.517	27.563
vehicle	275.894	915.387	0.137
vowel	1346.284	4508.354	0.05
wine	73.789	131.027	0.02
wisconsin	53.366	111.057	0.075
zoo	185.7	455.924	0.009
Average	197.618	663.967	0.838

Table 7: Friedman's test ranking for accuracy, reduction rate and accuracy-reduction balance.

algorithm	Rank for ACC	Rank for RED	Rank for ACC*RED
ParetoTSS	**31.49**	**33.14**	**29.4**
GGA	45.14	56.28	46.5
SE	86.88	74.08	87.6

Table 8: Finner's test for accuracy metric. The control algorithm is ParetoTSS

i	algorithm	z value	unadjusted p-value	$1-(1-\alpha)^{\frac{i}{k-1}}$
2	GGA	1.8494	0.0644	0.1
1	SE	7.5028	$6.2460 \cdot 10^{-14}$	0.05132

critical value for two degrees of freedom 4.605 (to be considered being three the number of compared algorithms), the null hypothesis is rejected and it is possible to assess that there is a significant difference between at least two of the compared algorithms in terms of accuracy. Attending to this result, a post-hoc statistical analysis, the Finner's test, is needed to conduct pairwise comparisons to detect concrete differences among compared algorithms. ParetoTSS is chosen as control algorithm in the Finner's test having obtained the best ranking in the Aligned Friedman's test. All data computed by the Finner's test for the accuracy measure are displayed in Table 8. By analyzing this table, it is possible to state that ParetoTSS statistically outperforms GGA and SE at the 90% confidence level in terms of accuracy.

As for the reduction rate, from the results reported in Table 5, it emerges that ParetoTSS outperforms GGA in 32 out of 36 datasets and SE in 28 out of 36. As for the average, ParetoTSS is characterized by the highest average reduction rate. Precisely, ParetoTSS improves the reduction rate of GGA by 2.9% and of SE by 9.6%.

Therefore, from this empirical analysis, ParetoTSS is proven to be the best performer also in terms of reduction rate. To verify this empirical result, we performed a statistical multiple comparison procedure also for the reduction rate metric. The samples used in the statistical analysis are the reduction rate values obtained by all TSS techniques over all datasets shown in the Table 5.

Table 9: Finner's test for reduction rate metric. The control algorithm is ParetoTSS.

i	algorithm	z value	unadjusted p-value	$1-(1-\alpha)^{\frac{i}{k-1}}$
2	GGA	3.1343	$1.7225 \cdot 10^{-3}$	0.1
1	SE	5.5462	$2.9192 \cdot 10^{-4}$	0.05132

Table 7 shows the ranking obtained by all compared TSS techniques during the Aligned Friedman's test. As highlighted in bold, the algorithm with the best ranking (the lowest value) for reduction rate metric is ParetoTSS. The computed Aligned Friedman's statistic is 22.34. Since it is greater than the critical value 4.605, the null hypothesis is rejected and it is possible to assess that there is a significant difference between at least two of the compared algorithms in terms of reduction rate. With arguments similar to the one before, we performed the Finner's test also for the reduction rate measure. By analyzing data in Table 9, it results that the Finner's test rejects all the hypotheses. Hence, ParetoTSS statistically outperforms GGA and SE at the 90% confidence level in terms of reduction rate.

Finally, as for the accuracy-reduction balance, ParetoTSS outperforms GGA in 29 out of 36 datasets and SE in 33 out of 36 datasets. Moreover, ParetoTSS improves the average accuracy-reduction balance of GGA by 7.48% and of SE by 38.3%.

To complete this comparison, we performed the statistical multiple comparison procedure. The samples used in the statistical analysis are the accuracy-reduction balance values obtained by all TSS techniques over all datasets. Table 7 shows the ranking obtained by all compared TSS techniques during the Aligned Friedman's test. As highlighted in bold, the algorithm with the lowest ranking for the accuracy-reduction balance is ParetoTSS. The computed Aligned Friedman's statistic is 46.97. Since it is greater than the critical value 4.605, we performed the Finner's test. ParetoTSS is chosen as control algorithm in the Finner's test having obtained the best ranking in the Aligned Friedman's test. Table 10 shows all data computed by the Finner's test for the accuracy-reduction balance measure. By analyzing this data, the Finner's procedure rejects all the

Table 10: Finner's test for the accuracy-reduction balance metric. The control algorithm is ParetoTSS.

i	algorithm	z value	unadjusted p-value	$1-(1-\alpha)^{\frac{i}{k-1}}$
2	GGA	2.3159	0.0206	0.1
1	SE	7.8829	$3.2 \cdot 10^{-15}$	0.05132

hypotheses. Therefore, it is possible to state that ParetoTSS statistically outperforms GGA and SE at the 90% confidence level in terms of trade-off between accuracy and reduction rate.

From the above analysis ParetoTSS results to be superior to all state-of-the-art approaches with respect to the accuracy-reduction balance. Unfortunately this comes with a higher computational time with respect to the not-evolutionary algorithm. Indeed, as shown in Table 6, ParetoTSS algorithm is remarkably slower than SE. This computational time difference is mainly due to the fact that SE reduces on the original training set by means of an iterative approach whose computational time is proportional to the number of instances multiplied by the number of attributes, whereas, ParetoTSS acts on a collection of potential reduced training sets and, for each algorithm evolution, it performs a computationally complex set of operations, i.e., crossover, mutation and fitness evaluation. However, ParetoTSS succeeds to be faster than GGA. This superiority in computational time performance is mainly due to the better performance obtained by ParetoTSS in computing fitness function evaluations than GGA. For both algorithms, each fitness function evaluation generates an SVM model starting from a reduced version of the original training set and, successively, it tests this model on the original training set. As aforementioned, the computational effort necessary to generate and test the SVM model strongly depends upon the size of the training set. As a consequence, if an evolutionary algorithm is able to quickly reduce the size of the training set in the solution pool, it will be able to quickly compute its fitness function evaluations. Now, as highlighted by the Table, ParetoTSS yields better reduction rate performance than GGA and, as a consequence, its fitness evaluations will be faster than those performed

by GGA. In order to statistically validate the best performance of ParetoTSS in running time with respect to GGA, we have performed a Wilcoxon's signed rank test by considering as sample data the running time values over all datasets held by only these algorithms. The computed p-value is $1.3361 \cdot 10^{-7}$. Hence, it is possible to state that ParetoTSS is statistically faster than GGA at 99% confidence level.

6. Conclusions

This paper presents ParetoTSS, a TSS technique for the SVM mainly based, for the first time, on a Pareto-based multi-objective evolutionary approach such as PESA-II. After a set of experiments, we can conclude that the proposed approach is characterized by a high reduction rate, a good capability to work with data never seen before and the ability of not leading to the overfitting. Moreover, its application allows enhancing SVM classification accuracy and efficiency and improving on state-of-the-art TSS techniques.

The good performance shown by ParetoTSS opens new research activities in the field of the application of Pareto-based approaches to the TSS problem. As further research we are working to analyze the performance of different Pareto-based multi-objective approaches in terms of measures such as hypervolume and Δ-index [57] specific for comparisons among Pareto-based approaches. Besides, another interesting study will be to investigate different decision-making mechanisms for selecting a solution in Pareto fronts obtained by the studied multi-objective approaches.

Acknowledgments

The paper has been partially supported by the Spanish project identified by the grant number TIN2017-89517-P, funded by Ministerio de Economía, Industria y Competitividad.

Appendix A. Support Vector Machine

SVM is a very effective method for classification and regression. Different from unsupervised techniques [58][59][60], SVM works by building a model based on training data which predicts the class for new data (called test data). In order to achieve this goal, SVM uses a discriminant hyperplane. The selected hyperplane is the one that maximizes the distance (the margin) from the so-called *support vectors*, i.e., the data points closest to the hyperplane (see Fig. A.5) [61]. Therefore, the solution to the classification problem is based only on these data points at the margin. An SVM classification using linear decision boundaries is known as linear SVM. However, the data is often not linearly separable [4]. In order to overcome this problem, a non-linear extension of SVM can be achieved by using the *kernel trick* [62]. This mechanism consists in implicitly mapping the data to another space, generally of much higher dimensionality, using a kernel function $K(x, y)$. Well-known kernels are:

- Linear kernel:

$$K(x, y) = x * y$$

  where x and y are vectors in the input space, i.e. vectors of features;

- Polynomial kernel:

$$K(x, y) = (x * y + 1)^p$$

  where x and y are vectors in the input space, i.e. vectors of features, and p is the degree of the polynomial;

- Radial Basis Function (RBF) kernel:

$$K(x, y) = e^{-\frac{||x-y||^2}{2\sigma^2}}$$

  where x and y are vectors in the input space, i.e. vectors of features, and σ is a free parameter.

Apart from the parameters related to the kernel functions, SVM design requires the setting of a regularization parameter, named C, used as a trade-off between

allowing training errors and forcing rigid margins. The choice of this parameter is crucial because large values for C lead to few training errors, but, on the other hand, it may create a model overfitting the training data.

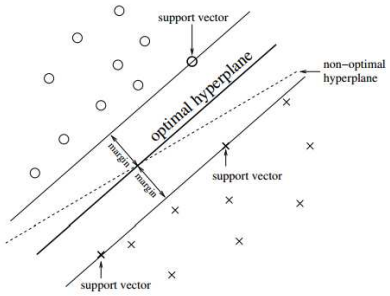


Figure A.5: SVM optimal hyperplane description [61]

Natively, SVM deals with binary problems. In the literature, some methodologies have been introduced to allow SVM to address multi-class problems. A traditional approach is the pairwise coupling [63][64], where the multi-class problem is decomposed in all possible two-class problems and the majority voting principle is applied [4].

References

[1] C. Cortes, V. Vapnik, Support-vector networks, Machine learning 20 (3) (1995) 273–297.

[2] L. Wang, Preface, in: L. Wang (Ed.), Support vector machines: theory and applications, Vol. 177, Springer, Berlin, Heidelberg, 2005, pp. V–VII.

[3] K. Zhu, H. Wang, H. Bai, J. Li, Z. Qiu, H. Cui, E. Y. Chang, Parallelizing support vector machines on distributed computers, in: Advances in Neural Information Processing Systems, 2008, pp. 257–264.

[4] N. Verbiest, J. Derrac, C. Cornelis, S. García, F. Herrera, Evolutionary wrapper approaches for training set selection as preprocessing mechanism

for support vector machines: experimental evaluation and support vector analysis, Applied Soft Computing 38 (2016) 10–22.

[5] S. García, A. Fernández, F. Herrera, Enhancing the effectiveness and interpretability of decision tree and rule induction classifiers with evolutionary training set selection over imbalanced problems, Applied Soft Computing 9 (4) (2009) 1304–1314.

[6] S. Garcia, J. Derrac, J. Cano, F. Herrera, Prototype selection for nearest neighbor classification: Taxonomy and empirical study, IEEE Transactions on Pattern Analysis and Machine Intelligence 34 (3) (2012) 417–435.

[7] G. Acampora, U. Kaymak, V. Loia, A. Vitiello, Applying NSGA-II for solving the ontology alignment problem, in: 2013 IEEE International Conference on Systems, Man, and Cybernetics, 2013, pp. 1098–1103.

[8] A. Messac, C. A. Mattson, Generating well-distributed sets of pareto points for engineering design using physical programming, Optimization and Engineering 3 (4) (2002) 431–450.

[9] I. Das, J. E. Dennis, A closer look at drawbacks of minimizing weighted sums of objectives for pareto set generation in multicriteria optimization problems, Structural optimization 14 (1) (1997) 63–69.

[10] K. Deb, A. Pratap, S. Agarwal, T. Meyarivan, A fast and elitist multiobjective genetic algorithm: NSGA-II, IEEE Transactions on Evolutionary Computation 6 (2) (2002) 182–197.

[11] S. Mitra, H. Banka, Multi-objective evolutionary biclustering of gene expression data, Pattern Recognition 39 (12) (2006) 2464 – 2477.

[12] R. Sarkhel, N. Das, A. K. Saha, M. Nasipuri, A multi-objective approach towards cost effective isolated handwritten bangla character and digit recognition, Pattern Recognition 58 (2016) 172 – 189.

[13] G. Acampora, H. Ishibuchi, A. Vitiello, A comparison of multi-objective evolutionary algorithms for the ontology meta-matching problem, in: 2014 IEEE Congress on Evolutionary Computation, 2014, pp. 413–420.

[14] D. W. Corne, N. R. Jerram, J. D. Knowles, M. J. Oates, et al., PESA-II: Region-based selection in evolutionary multiobjective optimization, in: Proceedings of the Genetic and Evolutionary Computation Conference, Morgan Kaufmann Publishers, 2001, pp. 283–290.

[15] M. Li, S. Yang, X. Liu, K. Wang, IPESA-II: Improved pareto envelope-based selection algorithm ii, in: International Conference on Evolutionary Multi-Criterion Optimization, Springer, 2013, pp. 143–155.

[16] M. Lichman, UCI machine learning repository (2013).
URL `http://archive.ics.uci.edu/ml`

[17] F. O. Catak, M. E. Balaban, CloudSVM: training an SVM classifier in cloud computing systems, in: Joint International Conference on Pervasive Computing and the Networked World, Springer, 2012, pp. 57–68.

[18] J. Derrac, I. Triguero, S. Garcia, F. Herrera, Integrating instance selection, instance weighting, and feature weighting for nearest neighbor classifiers by coevolutionary algorithms, IEEE Transactions on Systems, Man, and Cybernetics, Part B (Cybernetics) 42 (5) (2012) 1383–1397.

[19] S. García, J. R. Cano, F. Herrera, A memetic algorithm for evolutionary prototype selection: A scaling up approach, Pattern Recognition 41 (8) (2008) 2693–2709.

[20] S. Vluymans, I. Triguero, C. Cornelis, Y. Saeys, EPRENNID: An evolutionary prototype reduction based ensemble for nearest neighbor classification of imbalanced data, Neurocomputing 216 (Supplement C) (2016) 596 – 610.

[21] G. Acampora, G. Tortora, A. Vitiello, Applying SPEA-2 to prototype selection for nearest neighbor classification, in: 2016 IEEE International Confer-

ence on Systems, Man, and Cybernetics (SMC), 2016, pp. 003924–003929. doi:10.1109/SMC.2016.7844847.

[22] K. jae Kim, Artificial neural networks with evolutionary instance selection for financial forecasting, Expert Systems with Applications 30 (3) (2006) 519 – 526, intelligent Information Systems for Financial Engineering.

[23] L. I. Kuncheva, Editing for the k-nearest neighbors rule by a genetic algorithm, Pattern Recognition Letters 16 (8) (1995) 809–814.

[24] J. R. Cano, F. Herrera, M. Lozano, Using evolutionary algorithms as instance selection for data reduction in KDD: an experimental study, IEEE Transactions on Evolutionary Computation 7 (6) (2003) 561–575.

[25] D. L. Wilson, Asymptotic properties of nearest neighbor rules using edited data, IEEE Transactions on Systems, Man, and Cybernetics (3) (1972) 408–421.

[26] J. Chen, C. Zhang, X. Xue, C.-L. Liu, Fast instance selection for speeding up support vector machines, Knowledge-Based Systems 45 (2013) 1–7.

[27] A. Bouraoui, Y. Ben Ayed, S. Jamoussi, A multi-objective genetic algorithm for model selection for support vector machines, in: D.-N. Pham, S.-B. Park (Eds.), PRICAI 2014: Trends in Artificial Intelligence: 13th Pacific Rim International Conference on Artificial Intelligence, Gold Coast, QLD, Australia, December 1-5, 2014. Proceedings, Springer International Publishing, Cham, 2014, pp. 809–819.

[28] L. Xu, C. Li, Multi-objective parameters selection for SVM classification using NSGA-II, in: P. Perner (Ed.), Advances in Data Mining. Applications in Medicine, Web Mining, Marketing, Image and Signal Mining: 6th Industrial Conference on Data Mining, ICDM 2006, Leipzig, Germany, July 14-15, 2006. Proceedings, Springer Berlin Heidelberg, Berlin, Heidelberg, 2006, pp. 365–376.

[29] P. B. C. Miranda, R. B. C. Prudncio, A. C. P. L. F. de Carvalho, C. Soares, Multi-objective optimization and meta-learning for SVM parameter selection, in: The 2012 International Joint Conference on Neural Networks, 2012, pp. 1–8.

[30] M. F. A. Hady, W. Herbawi, M. Weber, F. Schwenker, A multi-objective genetic algorithm for pruning support vector machines, in: 2011 IEEE 23rd International Conference on Tools with Artificial Intelligence, IEEE, 2011, pp. 269–275.

[31] R. Pighetti, D. Pallez, F. Precioso, Improving SVM training sample selection using multi-objective evolutionary algorithm and LSH, in: 2015 IEEE Symposium Series on Computational Intelligence, IEEE, 2015, pp. 1383–1390.

[32] A. Rosales-Prez, S. Garca, J. A. Gonzalez, C. A. C. Coello, F. Herrera, An evolutionary multiobjective model and instance selection for support vector machines with pareto-based ensembles, IEEE Transactions on Evolutionary Computation 21 (6) (2017) 863–877.

[33] M. B. De Almeida, A. de Pádua Braga, J. P. Braga, SVM-KM: speeding svms learning with a priori cluster selection and k-means, in: The sixth Brazilian Symposium on Neural Networks, IEEE, 2000, pp. 162–167.

[34] C. Shujuan, L. Xiaomao, Z. Jun, et al., Fuzzy support vector machine of dismissing margin based on the method of class-center, Computer Engineering and Applications 42 (22) (2006) 146–149.

[35] L. Yu, Y. Wende, H. Dake, L. Yu, Fast reduction for large-scale training data set, Journal of Southwest Jiaotong University 4 (2007) 15–22.

[36] Q. Li, L. C. Jiao, W. D. Zhou, Pre-extracting support vector for support vector machine based on vector projection, Chinese Journal of Computers 2 (2005) 12–20.

[37] C. Liu, W. Wang, M. Wang, F. Lv, M. Konan, An efficient instance selection algorithm to reconstruct training set for support vector machine, Knowledge-Based Systems 116 (2017) 58 – 73.

[38] R. Shang, L. Jiao, F. Liu, W. Ma, A novel immune clonal algorithm for mo problems, IEEE Transactions on Evolutionary Computation 16 (1) (2012) 35–50.

[39] L. Jiao, L. Li, R. Shang, F. Liu, R. Stolkin, A novel selection evolutionary strategy for constrained optimization, Information Sciences 239 (2013) 122–141.

[40] R. M. Jarvis, W. Rowe, N. R. Yaffe, R. OConnor, J. D. Knowles, E. W. Blanch, R. Goodacre, Multiobjective evolutionary optimisation for surface-enhanced raman scattering, Analytical and bioanalytical chemistry 397 (5) (2010) 1893–1901.

[41] F. di Pierro, S.-T. Khu, D. Savi, L. Berardi, Efficient multi-objective optimal design of water distribution networks on a budget of simulations using hybrid algorithms, Environmental Modelling & Software 24 (2) (2009) 202 – 213.

[42] J. Handl, J. Knowles, Multi-objective clustering and cluster validation, in: Y. Jin (Ed.), Multi-objective machine learning, Vol. 16, Springer-Verlag Berlin Heidelberg, 2006, Ch. 2, pp. 21–47.

[43] D. W. Corne, J. D. Knowles, M. J. Oates, The pareto envelope-based selection algorithm for multiobjective optimization, in: International Conference on Parallel Problem Solving from Nature, Springer, 2000, pp. 839–848.

[44] S. García, F. Herrera, Evolutionary undersampling for classification with imbalanced datasets: Proposals and taxonomy, Evolutionary computation 17 (3) (2009) 275–306.

[45] T. E. Koch, A. Zell, Mocs: Multi-objective clustering selection evolutionary algorithm, in: Proceedings of the 4th Annual Conference on Genetic and

Evolutionary Computation, Morgan Kaufmann Publishers Inc., 2002, pp. 423–430.

[46] J. Platt, et al., Sequential minimal optimization: A fast algorithm for training support vector machines, Tech. rep., Technical Report MSR-TR-98-14, Microsoft Research (1998).

[47] S. R. Garner, et al., Weka: The waikato environment for knowledge analysis, in: Proceedings of the New Zealand computer science research students conference, 1995, pp. 57–64.

[48] J. J. Durillo, A. J. Nebro, jmetal: A java framework for multi-objective optimization, Adv. Eng. Softw. 42 (10) (2011) 760–771.

[49] J. Alcalá-Fdez, L. Sanchez, S. Garcia, M. J. del Jesus, S. Ventura, J. M. Garrell, J. Otero, C. Romero, J. Bacardit, V. M. Rivas, et al., KEEL: a software tool to assess evolutionary algorithms for data mining problems, Soft Computing 13 (3) (2009) 307–318.

[50] I. H. Witten, E. Frank, Data Mining: Practical machine learning tools and techniques, Morgan Kaufmann, 2005.

[51] J. Demšar, Statistical comparisons of classifiers over multiple data sets, Journal of Machine learning research 7 (2006) 1–30.

[52] F. Wilcoxon, Individual comparisons by ranking methods, Biometrics bulletin 1 (6) (1945) 80–83.

[53] J. Hodges, E. Lehmann, Rank methods for combination of independent experiments in analysis of variance, The Annals of Mathematical Statistics 33 (2) (1962) 482–497.

[54] H. Finner, On a monotonicity problem in step-down multiple test procedures, Journal of the American Statistical Association 88 (423) (1993) 920–923.

[55] S. García, A. Fernández, J. Luengo, F. Herrera, Advanced nonparametric tests for multiple comparisons in the design of experiments in computational intelligence and data mining: Experimental analysis of power, Information Sciences 180 (10) (2010) 2044–2064.

[56] J. Derrac, S. García, D. Molina, F. Herrera, A practical tutorial on the use of nonparametric statistical tests as a methodology for comparing evolutionary and swarm intelligence algorithms, Swarm and Evolutionary Computation 1 (1) (2011) 3–18.

[57] T. Okabe, Y. Jin, B. Sendhoff, A critical survey of performance indices for multi-objective optimisation, in: The 2003 Congress on Evolutionary Computation, Vol. 2, IEEE, 2003, pp. 878–885.

[58] L. Jiao, F. Shang, F. Wang, Y. Liu, Fast semi-supervised clustering with enhanced spectral embedding, Pattern Recognition 45 (12) (2012) 4358–4369.

[59] R. Shang, Z. Zhang, L. Jiao, W. Wang, S. Yang, Global discriminative-based nonnegative spectral clustering, Pattern Recognition 55 (2016) 172–182.

[60] R. Shang, W. Wang, R. Stolkin, L. Jiao, Subspace learning-based graph regularized feature selection, Knowledge-Based Systems 112 (2016) 152–165.

[61] F. Lotte, M. Congedo, A. Lécuyer, F. Lamarche, B. Arnaldi, A review of classification algorithms for EEG-based brain–computer interfaces, Journal of neural engineering 4 (2) (2007) 1–24.

[62] Y.-J. Lee, Y.-R. Yeh, H.-K. Pao, Introduction to support vector machines and their applications in bankruptcy prognosis, in: Handbook of Computational Finance, Springer, 2012, pp. 731–761.

[63] M. Galar, A. Fernández, E. Barrenechea, H. Bustince, F. Herrera, An overview of ensemble methods for binary classifiers in multi-class prob-

lems: Experimental study on one-vs-one and one-vs-all schemes, Pattern Recognition 44 (8) (2011) 1761–1776.

[64] S.-H. Park, J. Fürnkranz, Efficient pairwise classification, in: European Conference on Machine Learning, Springer, 2007, pp. 658–665.