Applied Soft Computing 38 (2016) 10–22

Contents lists available at ScienceDirect

Applied Soft Computing

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

Evolutionary wrapper approaches for training set selection as preprocessing mechanism for support vector machines: Experimental evaluation and support vector analysis

Nele Verbiest[a,*], Joaquín Derrac[b], Chris Cornelis[a,c], Salvador García[c,d], Francisco Herrera[c]

[a] Department of Applied Mathematics and Computer Science, Ghent University, Belgium
[b] Affectv Limited, London, United Kingdom
[c] Department of Computer Science and AI, Research Center on Information and Communications Technology (CITIC-UGR), University of Granada, Spain
[d] Department of Information Systems, Faculty of Computing and Information Technology, King Abdulaziz University, Jeddah, Saudi Arabia

ARTICLE INFO

Article history:
Received 31 January 2014
Received in revised form 2 September 2015
Accepted 3 September 2015
Available online 30 September 2015

Keywords:
Support vector machines
Training set selection
Data reduction

ABSTRACT

One of the most powerful, popular and accurate classification techniques is support vector machines (SVMs). In this work, we want to evaluate whether the accuracy of SVMs can be further improved using training set selection (TSS), where only a subset of training instances is used to build the SVM model. By contrast to existing approaches, we focus on wrapper TSS techniques, where candidate subsets of training instances are evaluated using the SVM training accuracy. We consider five wrapper TSS strategies and show that those based on evolutionary approaches can significantly improve the accuracy of SVMs.

1. Introduction

In many real-world applications, datasets can contain noisy or wrong information. Even the best classifiers might not be able to deal with these datasets. Training Set Selection (TSS, [1–3]) is a good way to alleviate this problem. It is a preprocessing technique that only selects relevant instances before applying the classifier. The objective of TSS is twofold: on the one hand, the accuracy of the classifier can be improved, while on the other hand, the efficiency can be enhanced.

TSS has mainly been investigated for the K Nearest Neighbor (KNN) classifier [4], in that context it is referred to as Prototype Selection (PS, [5]). There are two main groups of PS techniques. Wrapper techniques use the KNN classifier to evaluate entire candidate subsets of instances, while filter techniques do not make use of the KNN classifier or only use it to carry out partial evaluations.

In this work we want to study if TSS techniques can also improve the accuracy of Support Vector Machines (SVMs). As wrapper PS techniques explicitly use the KNN classifier, they cannot be applied meaningfully to improve SVM classification. It is clear, however, that filter PS techniques can be directly applied to SVMs as they are less dependent on the KNN classifier. On the other hand, filter methods are in general less suited to improve the accuracy of a classifier.

To the best of our knowledge, only two filter TSS techniques have been proposed to specifically improve SVMs. In [6], the Multi-Class Instance Selection (MCIS) method is proposed, which selects instances near the boundary between one and the other classes of datasets. This method focuses on reduction of the dataset to improve the efficiency of the SVMs. Another approach is presented in [7], where only training instances that are likely to become support vectors are selected. This Sparsifying Neural Gas (SNG) algorithm was developed to improve the efficiency of the SVM while maintaining or slightly improving the accuracy.

Unfortunately, these filter TSS techniques are unable to improve the accuracy of the SVMs. Therefore we attempt to improve the accuracy of SVMs using wrapper approaches. We adapt the five most important wrapper TSS techniques by plugging the SVM into the TSS methods. The wrapper techniques we consider evaluate candidate subsets based on the so-called training accuracy, which is the accuracy obtained when building the classifier at hand based on the candidate subset and using this model to classify the entire training data. In our case we use SVMs to calculate the training

* Corresponding author at: Krijgslaan 281 (S9), 9000 Gent, Belgium. Tel.: +32 92644770; fax: +32 92644995.
E-mail address: Nele.Verbiest@UGent.be (N. Verbiest).

http://dx.doi.org/10.1016/j.asoc.2015.09.006
1568-4946/

accuracy, and as a result subsets with a high training accuracy will be well-suited for SVM classification.

The remainder of this paper is organized as follows: In Section 2 we provide the necessary background on SVMs. In Section 3 we present existing filter TSS techniques and in Section 4 we present the design of the wrapper TSS techniques for SVMs. Then, we set up an experimental framework to evaluate the approaches' performance in Section 5. By means of an experimental evaluation on 43 real-life datasets, we show that wrapper TSS techniques can indeed significantly improve SVM classification. Evolutionary approaches, and the Generational Genetic Algorithm (GGA,[8,9]) in particular, seem to be especially well-suited for our purpose. In order to get more insight into the operation of the evolutionary wrappers, we provide a more detailed analysis for the latter, investigating the effect of TSS on the SVM's support vectors, and illustrating their behavior graphically on a two-dimensional artificial dataset. Finally, we conclude in Section 6.

2. Preliminaries

In this subsection we provide a general background on SVMs to make the paper self-contained. We denote instances by their feature vectors x. For now, we consider two-class problems, the class of an instance is either -1 or 1. At the end of this section we discuss the multi-class case.

The most basic form of SVMs are separating hyperplanes, where one aims to separate the two classes linearly by a hyperplane. The hyperplane can be represented by a linear function $f(x)$ that is optimized such that the distance from the hyperplane to the closest instances from both classes is maximal. To classify a new instance t, the value $f(t)$ is calculated. When $f(t) > 0$, t is classified to class 1 and else to the negative class -1.

In practice, the data is often not linearly separable, which led to the introduction of support vector classifiers, which allow for overlap between the classes. The idea is to find a hyperplane that maximizes the margin between the two classes, but allows for some data points to fall on the wrong side of the margin. Of course the number of misclassified training points is bounded. It can be shown that the resulting hyperplane is a linear combination of a set of instances, these lie on the classification margin and are called support vectors.

To allow for even more flexibility, kernel-based SVMs were introduced. Before constructing the separating hyperplane, the feature space is enlarged using a function h such that for two feature vectors x_1 and x_2

$$h(x_1)^t h(x_2) = K(x_1, x_2) \tag{1}$$

where K is a kernel function. A well-known example is the Radial Basis Function (RBF) kernel (x_1, x_2 feature vectors):

$$K(x_1, x_2) = \exp\left(-\frac{||x_1 - x_2||^2}{2\sigma^2}\right). \tag{2}$$

The separating hyperplanes are represented by a function f that takes values in $(-\infty, \infty)$. However, it is more useful to obtain probabilities. Therefore, a sigmoid model can be used to calculate the probabilities $P(y = 1|f)$. This scaling, referred to as Platt's scaling [10], can also be seen as training the model to find a better threshold: instead of using the standard 0 as threshold to classify test instances, we can train the model based on the class probabilities to find a better threshold.

The discussed methods apply to two-class problems. A traditional approach to handle multi-class problems is pairwise coupling [11–13], where the multi-class problem is decomposed in all possible two-class problems and the majority voting principle is applied. For instance, when there are K classes, for each pair of classes i and j with $i, j \leq K$ and $i \neq j$, the binary SVM is constructed. A new instance is classified by all classifiers, and each class gets a vote if the new instance is classified to that class. The class with the highest number of votes is the final class returned for that instance. Another approach is the so-called one-versus-all technique. In this case, K training datasets are considered, where in each dataset one class is the positive class and the remaining classes form the negative class. The SVM is trained on each of these training datasets and the target instance t is classified by each SVM. Each SVM returns a probability value p expressing the confidence that t should be classified to the positive class. Finally, t is classified to the class for which this probability is maximal.

3. Related work: filter TSS techniques for SVMs

In [5], a comprehensive overview of TSS techniques is provided. Apart from the already mentioned distinction between wrapper and filter approaches, TSS techniques can also be categorized as edition, condensation or hybrid methods: while edition (or editing) methods remove noisy instances in order to increase classifier accuracy, condensation methods compute a training set consistent subset, removing superfluous instances that will not affect the classification accuracy of the training set. Finally, methods that eliminate both noisy and superfluous are called hybrid ones.

As we are mainly interested in improving the accuracy of SVM classification, we only consider the nine editing filter techniques discussed in [5]. They are reviewed in Section 3.1.

In addition to the nine TSS techniques from [5], which were originally developed to improve KNN, we also consider two filter TSS techniques that were specifically developed for SVMs. These are discussed in Section 3.2.

3.1. Editing filter TSS methods

A basic method is the Edited Nearest Neighbor (ENN, [14]) algorithm, which considers every instance in the training set and removes it whenever the nearest neighbor rule classifies it incorrectly using the remaining instances as training data. Many methods are derived from ENN, including:

- ENN with Estimation of Probabilities of Threshold (ENNTh, [15]): proceeds like ENN, except that the removal criterion is based on probabilities.
- All-KNN ([16]): applies ENN for different numbers of neighbors and removes an instance whenever any of the ENN runs marked an instance for removal.
- Modified ENN (MENN, [17]): takes into account the fact that multiple instances can be at the same distance from the target instance.
- Nearest Centroid Neighborhood Edition (NCNEdit, [18]): is very similar to ENN, but uses an alternative definition to determine the neighbors of an element based on centroids.
- Multi-Edit [19]: randomly divides the training data in blocks, applies ENN to them and merges the resulting sets.

We also consider three methods that are not derived from ENN:

- Relative Neighborhood Graph (RNG, [20]): constructs a proximity graph and removes instances that are misclassified by the neighbors in the graph.
- Model Class Selection (MOCS, [21]): uses a feedback system to incorporate knowledge about the dataset in a tree-based classifier.

- Edited Normalized Radial Basis Function (ENRBF, [22]): calculates for each instance the probability that it belongs to each class. Instances for which the actual class does not correspond to the class with the highest probability are removed.

3.2. Filter TSS methods designed for SVM

The first SVM-specific technique is Multi-Class Instance Selection (MCIS, [6]), which can only be used in a one-versus-all setting. When the number of classes is K, the one-versus-all scheme considers K problems, where the i-th problem considers the i-th class as positive and the remaining classes as negative. For each of these problems, a subset of instances S is selected, and the SVM is trained on S instead of on the entire training set. The MCIS algorithm clusters only the positive class and then removes instances of the positive class that are close to the centers of the clusters and selects instances of the negative class that are closest to the centers of the clusters. In this way, instances near the boundary between the positive and negative class are selected.

Another technique developed to improve SVMs is the Sparsifying Neural Gas (SNG, [7]) algorithm, which is restricted to two-class problems. The intention of SNG is to only select instances that will likely become support vectors in the final SVM classification. To this goal, a combination of learning vector quantization techniques and the growing neural gas algorithm is used.

Note that neither MCIS nor SNG was developed to improve the accuracy of SVM classification. Instead, the algorithms aim to improve the efficiency of SVM maintaining a good accuracy rate. However, as MCIS and SNG were specifically developed for SVM classification we do include them in our experimental study to get a complete understanding of TSS for SVM.

4. Wrapper TSS techniques for SVMs

In this section, we present our approach to use wrapper TSS techniques for SVMs. Recall that wrapper TSS methods depend on the classifier used, in our case SVM. One component that all presented wrapper TSS techniques have in common is that they evaluate candidate subsets of instances based on their training accuracy. When the training set is given by X and subset $S \subseteq X$ is a candidate subset of instances, its training accuracy, denoted by $acc(S)$, is calculated as Algorithm 1 depicts.

Algorithm 1. Wrapper TSS for SVM
Construct the SVM based on the instances in S
For each instance in X (including instances in S)
  Classify it using the SVM obtained in the previous step
count = number of correctly classified instances in X
Return $count/|X|$ as training accuracy.

Next, we plug this function *acc* into five wrapper TSS techniques. Below, we summarize the procedures that these techniques follow. In particular, Section 4.1 considers three evolutionary approaches, while two non-evolutionary approaches (one hybrid and one editing method) are reviewed in Section 4.2. For more technical details of each of the described methods, we refer to the corresponding papers.

4.1. Evolutionary wrapper TSS methods

The main concept of the evolutionary methods we consider in this section is that they maintain a population of individuals, which are subsets of instances in the TSS case. The algorithms initialize the population randomly and then repeat the steps enumerated in Algorithm 2.

Algorithm 2. Cycle followed by evolutionary algorithms
Repeat until a specified number of Generations is reached
  Select the best individuals
  Generate new individuals from the selected, using cross-over and mutation
  Evaluate the fitness of the new individuals
  Survivor selection: replace the worst individuals in the population

The iteration is stopped when a fixed number of evaluations is reached and the prototype subset in the final population with the best fitness is returned.

The fitness of an individual S is based on the value $acc(S)$ defined above on the one hand and on the reduction $red(S)$ on the other hand, where $red(S) = \frac{|X|-|S|}{|X|}$ when X is the original training set. These two components are balanced as follows:

$$\text{fitness}(S) = \alpha acc(S) + (1-\alpha) red(S), \tag{3}$$

with $\alpha \in [0, 1]$ a user-defined variable.

In particular, we consider three evolutionary algorithms that follow the above general scheme, and whose particular characteristics are summarized below:

- Generational Genetic Algorithm (GGA, [8,9]): GGA follows the general scheme of evolutionary algorithms. The population is initialized randomly. Parent selection happens stochastically, that is, individuals with a higher fitness have a higher chance of being selected, but also individuals with a low fitness value can be selected as parent. Once the parents are selected, parents are matched randomly and offspring is generated using two-point crossover. Mutation only happens with a small probability, and the probability of a 0 to 1 mutation is smaller than the probability of a 1 to 0 mutation in order to force the algorithm to obtain higher reduction rates. Survivor selection is done by selecting the entire generated offspring and adding the fittest individual from the previous population, this is also referred to as elitism.
- CHC evolutionary algorithm ([23,24]): by contrast to GGA, CHC only matches parents that differ enough in order to prevent incest: if the similarity between two parents is larger than a certain threshold, no crossover takes place. The threshold is dynamic, that is, if there are nearly no parent pairs left, the threshold can be increased. Survivor selection happens by merging the old population with the new one and selecting the fittest individuals amongst them. CHC does not use mutations to introduce variation but re-initializes the population if it converges, by selecting the fittest individual found so far and changing a fixed percentage of randomly selected genes from 1 to 0.
- Steady State Genetic Algorithm (SSGA, [23]): SSGA follows the same procedure as GGA, but instead of selecting multiple parents in each step, only two parents are selected to generate offspring, and they always replace the two worst individuals in the population.

4.2. Other wrapper TSS methods

Random Mutation Hill Climbing (RMHC, [25]) is a hybrid TSS technique described in Algorithm 3.

Algorithm 3. Random Mutation Hill Climbing
Initialize a random subset of fixed number of instances S
Repeat a predefined number of times
  N = neighbor solution of S: replace one instance in S by one in X.
  If $acc(N) \geq acc(S)$,
    Replace S by N

Finally, the Fuzzy Rough Prototype Selection (FRPS, [26]) algorithm is an editing wrapper algorithm based on fuzzy rough set theory [27] that proceeds as illustrated in Algorithm 4.

Table 1
Description of the 43 datasets used in the experimental study: number of attributes, number of instances and number of classes.

NAME	#atts	#inst	#classes	NAME	#atts	#inst	#classes
appendicitis	7	106	2	iris	4	150	3
australian	14	690	2	led7digit	7	500	10
automobile	25	150	6	lymphography	18	148	4
balance	4	625	3	mammographic	5	830	2
bands	19	365	2	monk-2	6	432	2
breast	9	277	2	movement libras	90	360	15
bupa	6	345	2	new thyroid	5	215	3
car	6	1728	4	pima	8	768	2
cleveland	13	297	5	postoperative	8	87	3
contraceptive	9	1473	3	saheart	9	462	2
crx	15	653	2	sonar	60	208	2
dermatology	34	358	6	spectfheart	44	267	2
ecoli	7	336	8	tae	5	151	3
flare	11	1066	6	tic-tac-toe	9	958	2
german	20	1000	2	vehicle	18	846	4
glass	9	214	7	vowel	13	990	11
haberman	3	306	2	wdbc	30	569	2
hayesroth	4	160	3	wine	13	178	3
heart	13	270	2	wisconsin	9	683	2
hepatitis	19	80	2	yeast	8	1484	10
housevotees	16	232	2	zoo	16	101	7
ionosphere	33	351	2				

Algorithm 4. Fuzzy Rough Prototype Selection
For all instances
  Measure the quality q based on fuzzy rough set theory
Order and rename the instances such that $q(i_1) \geq q(i_2) \geq \ldots \geq q(i_{|S|})$
Select the subset S_i among $S_1 = \{i_1\}$, $S_2 = \{i_1, i_2\}$, $S_3 = \{i_1, i_2, i_3\}$,. . .,
  $S_{|S|} = \{i_1, i_2, \ldots, i_{|S|}\}$ for which $acc(S_i)$ is highest.

The idea is that only high-quality instances should be selected; the threshold for the quality measure is determined using the training accuracy.

5. Experimental evaluation

In this section, we evaluate if TSS algorithms can improve the accuracy of SVM classification. We first discuss the experimental set-up of our evaluation in Section 5.1, the results are presented and discussed in Section 5.2.

5.1. Experimental set-up

We use 43 datasets from the Keel[1] and UCI [28] dataset repository. The properties of these datasets are described in Table 1. The number of instances and attributes is limited, as larger datasets might need specialized distributed techniques for TSS [29–33] beyond the scope of this paper.

In this study we use the Sequential Minimal Optimization (SMO, [34]) algorithm to construct the SVM, as it is one of the fastest and most regularly used optimization algorithms in the context of SVMs. It divides the optimization problem in several smaller problems and solves them analytically. We choose to use Platt's scaling [10] after building the SVM. We use the Radial Basis Function (RBF) kernel with $\sigma = 0.01$ and set the cost parameter $C = 1$. These parameters could be tuned, but as we want to study the net effect of TSS, we fix the parameters in our work. We use the pairwise coupling setting to handle multi-class problems, except for the MCIS algorithm which can only be used in combination with the one-versus-all strategy.

The parameters of the TSS methods, as proposed in [5,26] are described in Table 2.

We use a 10 fold cross validation procedure, that is, we divide the data in 10 folds and use each fold once as test data, and the remaining folds as train data. We apply the TSS technique to the 10 training datasets, this results in 10 subsets S. We build the SVM on each S and classify the instances in the corresponding test set using this SVM model.

To contrast the algorithms among each other we use Friedman's aligned-ranks test [35–38]. This procedure calculates the average aligned-ranks of each algorithm, obtained by computing the difference between the performance of the algorithm and the mean performance of all algorithms for each data set. When significant differences are detected, we use Holm's post-hoc procedure [39] to test if the algorithm with the best Friedman aligned-rank significantly outperforms the others. We report the adjusted p-values, which represent the lowest level of significance of a hypothesis that results in rejection.

Table 2
Parameters used for the TSS algorithms in the experimental evaluation.

	Parameters
ENN	Number of neighbors: 3
Multi-Edit	Number of neighbors: 1
	Number of sub-blocks: 3
RNG	Order of the graph: 1
MENN	Number of neighbors: 3
NCNEdit	Number of neighbors: 3
ENRBF	Sigma: 1
	Alpha: 1
ENNTh	Number of neighbors: 3
	Noise threshold: 0.7
AllKNN	Number of neighbors: 3
GGA	Number of iterations: 10 000
	Population size: 51
	Crossover probability: 0.6
	Mutation probability: 0.01
	Balancing parameter α : 0.5
CHC	Number of iterations: 10 000
	Population size: 50
	Balancing parameter α : 0.5
	Percentage of change in restart: %35
SSGA	Number of iterations: 10 000
	Population size: 50
	Mutation probability: 0.01
	Balancing parameter α : 0.5
RMHC	Number of iterations: 10 000
	Number of selected instances: 10%
FRPS	OWA-weights used as in [26]
	Version FRPS-4 of the algorithm in [26]

[1] http://www.keel.es/.

Table 3
Average accuracy, reduction rate and running time of the evolutionary TSS wrappers over all datasets.

	GGA	CHC	SSGA	NO TSS
Accuracy	81.21	79.14	79.67	77.04
Reduction	0.9246	0.9055	0.6540	0
Running time (in s)	3198.7	7514	3841.4	0

5.2. Results

In this Section we evaluate our proposals and compare them to the state of the art. In Section 5.2.1 we compare the evolutionary approaches and select the best one, which is then compared to the remaining TSS methods in Section 5.2.2.

5.2.1. Comparison among evolutionary approaches

We proposed three evolutionary TSS techniques for SVMs: GGA, CHC and SSGA. In the following, we first compare them with respect to classification accuracy, reduction rate and running time.

- **Accuracy**
  The goal of our work is to find a TSS method that improves the accuracy of SVMs. Therefore, our first selection criterion to select a good TSS method is accuracy. In Table 3, the average accuracy over all datasets of the evolutionary TSS methods can be found, along with the result obtained with the SVM classifier without preprocessing. The full accuracy results can be found in Table A.1 in Appendix A. GGA is the most accurate evolutionary TSS method on average, its accuracy is about 2 percent better than that of CHC and SSGA, and about 4 percent better than that obtained without TSS. We also remark that while CHC and SSGA have comparable accuracy, CHC only manages to improve on the baseline SVM classifier in 22 datasets, while for SSGA a net improvement occurs in 36 cases, only one less than for GGA.
  To test if the differences between the evolutionary methods are significant, we carry out the Friedman test. The p-value of this test is 0.000069, meaning that there are indeed significant differences detected between the three evolutionary methods. The rankings of the methods are given in Table 4, GGA gets the best (i.e. the lowest) ranking and is followed by SSGA and CHC. To test if GGA significantly outperforms CHC and SSGA we carry out the Holm post-hoc procedure. In the last column of Table 4 we show the adjusted p-values for the comparison between GGA and the two other evolutionary TSS methods CHC and SSGA. Both adjusted p-values are very low, which means that GGA outperforms the other evolutionary approaches significantly.
- **Reduction**
  Another important aspect of TSS methods is reduction, which we define as the percentage of removed instances. In Table 3, we show the reduction rate of each evolutionary wrapper TSS method, full results can be found in Table A.2 in Appendix A. Note that all evolutionary methods remove an important part of the data; indeed, as hybrid methods they combine editing and condensation capabilities, and thus remove both noisy and superfluous instances. SSGA removes the least instances on average, about 65 percent. GGA and CHC both remove about 90 percent of the instances on average.

Table 4
Aligned Friedman rankings of the evolutionary TSS algorithms and adjusted p-values resulting from the Holm post-hoc procedure comparing GGA to CHC and SSGA.

	Friedman ranking	Adjusted p-value
CHC	2.4186	0.000032
SSGA	2.0930	0.005053
GGA	1.4884	–

- **Running time**
  A third aspect that should be taken into account is running time. In Table A.3 in Appendix A we show the average running time of the wrapper TSS methods over the 10 folds. This only includes the running time of the TSS method, not the running time of the following training of the SVM and the testing phase. The average results over all datasets can be found in Table 3. As expected, these running times are rather long: SSGA and GGA take about one hour per dataset on average and CHC two hours.

Based on the above analysis, GGA is the most suited evolutionary TSS method for SVMs: it is the most accurate method among the evolutionary ones, improves on baseline SVM classification in 37 out of 43 selected datasets, removes about 90 percent of all instances and has a reasonable running time. Therefore, we will select GGA as our proposed TSS method for SVMs. In the next subsection, we will compare it to other TSS methods.

Before proceeding with this comparison, however, we want to obtain some more insight into the operation of evolutionary TSS algorithms, and better understand their interaction with the SVM classifier. In particular, we want to illustrate and evaluate the effect each of the TSS methods has on the selection of support vectors within the SVM algorithm. To this aim, we first apply them to a dataset that is easy to visualize, namely the two-dimensional artificial *banana* dataset used in [5] to compare PS methods. Next, we focus again on the 43 datasets of our experimental study: specifically, we study if there exists a correlation between the accuracy improvement by TSS and the fraction of support vectors in the reduced datasets that were also support vectors on the original datasets.

- **Illustration on Banana Dataset**
  The banana dataset is an artificial data set with 5,300 data points and two classes composed of three well-defined clusters of instances of the class -1 and two clusters of the class 1. Although the borders are clear among the clusters, there is a high overlap between both classes. The complete data set is illustrated in Fig. 1. We consider four types of data points:
  – SV-CLASS1: data points in class 1 that are support vectors when applying an SVM to it
  – NOSV-CLASS1: data points in class 1 that are no support vectors when applying an SVM to it
  – SV-CLASSMIN1: data points in class -1 that are support vectors when applying an SVM to it
  – NOSV-CLASSMIN1: data points in class -1 that are no support vectors when applying an SVM to it.

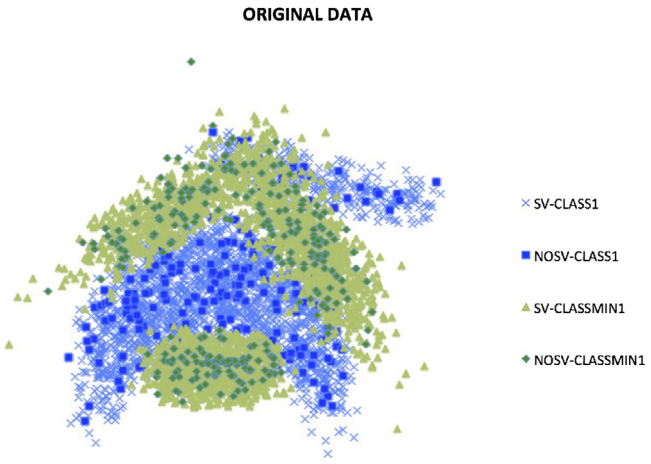

Fig. 1. Banana dataset.

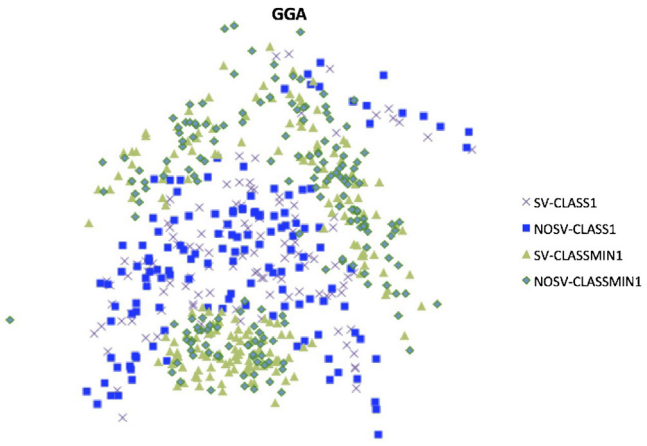

Fig. 2. Banana dataset after GGA preprocessing.

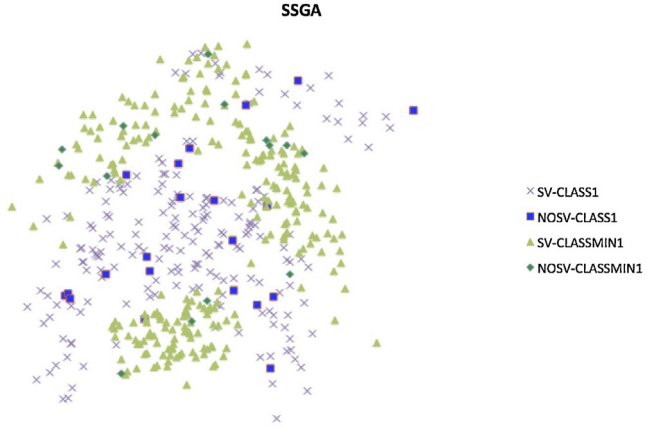

Fig. 3. Banana dataset after SSGA preprocessing.

We analyze what happens when evolutionary wrapper TSS methods are used before the SVM is applied to the data. The results for GGA, SSGA and CHC are shown in Figs. 2–4. Each graph contains the same four types of data points as the original dataset, but this time the support vectors are selected based on the reduced training set.

Although the results are similar, some differences between the TSS methods can be found. The first one is that there are few data points that are no support vectors after SSGA was applied. This is different for GGA and CHC: there are many data points selected that are not used as support vectors. When we look at the differences between CHC and GGA, we observe that the borders between the two classes are cleaner for GGA, that is, GGA is better at removing points that are in overlapping regions than CHC. When we look at the difference between GGA and SSGA, we see that the gaps between the two classes are larger for SSGA. This means that for this dataset, GGA is better at detecting border points, but does not remove too many points in the sense that there are no large gaps between the two classes.

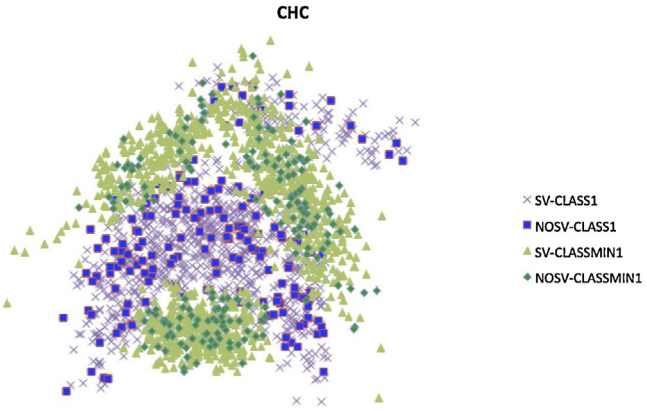

Fig. 4. Banana dataset after CHC preprocessing.

- **Analysis on KEEL and UCI Datasets**

As seen in the example of the banana dataset, TSS affects the problem landscape in terms of the selection of support vectors, that is: after TSS, the SVM algorithm builds a separating hyperplane using support vectors from a reduced training set, and in general these support vectors will not coincide with those obtained from the original training set. In order to investigate this effect, we considered, for each of the 43 datasets of the experimental study, and each of GGA, SSGA and CHC, the following fraction:

$$f = \frac{|\text{support vectors in } S \text{ that are also support vectors in } X|}{|\text{support vectors in } S|}$$

where X and S represent the original and reduced dataset, respectively. In particular, we want to evaluate if there exists a correlation between f and the improvement in terms of accuracy that can be obtained by applying the SVM algorithm to the reduced dataset. To this aim, Fig. 5 provides a scatter plot that shows, for each dataset and each TSS method, the corresponding f value (horizontal axis), and the difference in accuracy obtained with the reduced and unreduced datasets (vertical axis). That is, the higher a point appears in the graph, the higher is the improvement over the original data obtained by the corresponding TSS method. The graph also shows the average f values of SSGA, GGA and CHC.

Several observations can be made from this graph. First, on average, TSS by means of CHC leads to maintaining considerably less original support vectors ($f = 0.7080$) than TSS by SSGA ($f = 0.7859$) and GGA ($f = 0.7729$). On the other hand, when we look at the datasets for which the improvement is greatest (at least 5%), we can see that in the case of CHC, they mostly yield an f value lower than the average. By contrast, for SSGA and GGA, the best results are obtained with relatively high f values.

These observations may explain why there is much more variation in the accuracy results when CHC is used for reducing the training data: indeed, the separating hyperplane that SVM creates after TSS will differ more from the original one when CHC is used instead of GGA or SSGA. This may be attributed to the fact that CHC, because of the way it operates, is imposing a lot of variation on its candidate solutions. SSGA, by contrast, modifies the candidate solutions more cautiously and as a consequence the results show a much more stable behavior in terms of accuracy. A possible explanation why SSGA performs inferiorly to GGA (which is also introducing more variation in its candidate solutions than SSGA) may lie in the fact that SSGA suffers from overfitting: indeed, as the evaluation of a data point in a candidate solution happens using a model that was built using that same data point (amongst others, of course), there is a larger likelihood of overfitting than there is for instance for PS. GGA and CHC, by construction, check more varied solutions and thus avoid overfitting of a certain solution.

Summarizing, it may be hypothesized that GGA strikes the right balance between introducing variation and respecting the original support vectors of the considered classification problems.

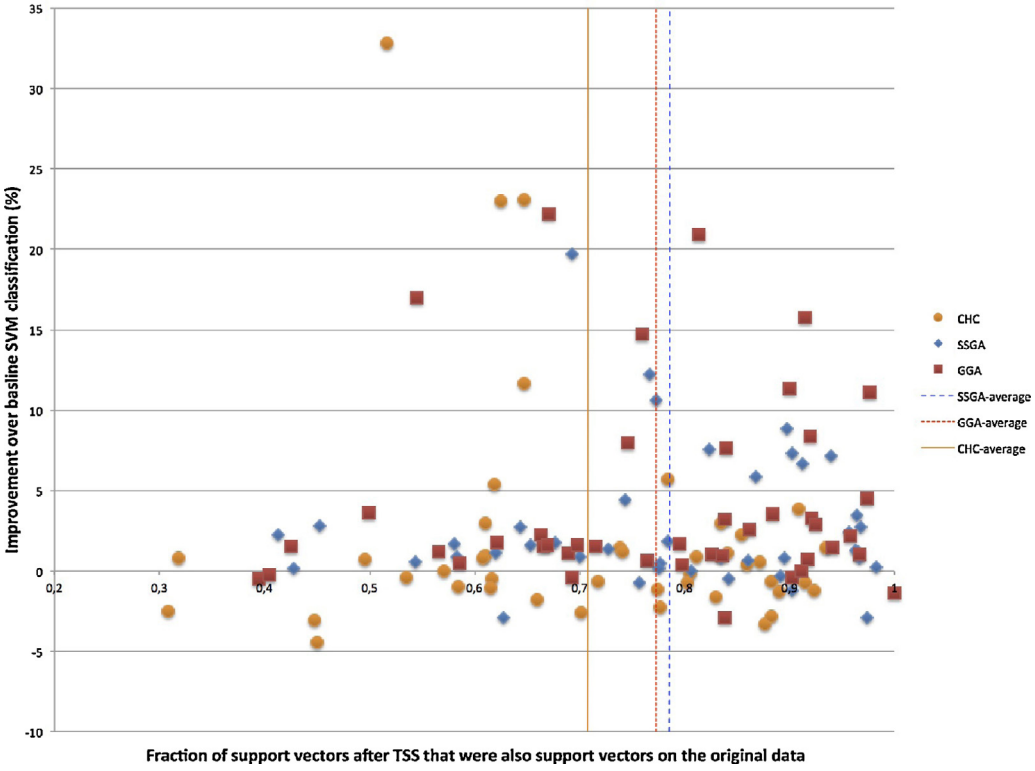

Fig. 5. Effect of support vector selection on SVM accuracy.

Table 5
Average accuracy, reduction rate and running time of GGA and the not evolutionary wrapper TSS Techniques.

	GGA	FRPS	RMHC
Accuracy	81.21	77.23	70.04
Reduction	0.9246	0.0959	0.9017
Running time (in s)	3198.7	205.9	2250.8

5.2.2. Comparison of GGA with other TSS techniques

In this part we compare our proposal, GGA, with the other TSS techniques. We first compare GGA against the other wrapper approaches FRPS and RMHC, and then compare GGA with the baseline and filter TSS techniques.

5.2.2.1. Comparison of GGA with other wrapper TSS techniques. We first compare the best evolutionary approach, GGA, with the other wrapper TSS techniques, FRPS and RMHC.

- **Accuracy**
  In Table 5 we show the average accuracy of GGA, FRPS and RMHC over all datasets, full results are in Table A.1 in Appendix A for reference. GGA clearly outperforms the other wrapper TSS approaches, especially RMHC performs poorly compared to GGA. This is confirmed by the Friedman test, the p-value is smaller than 0.000001 indicating that there are significant differences between the approaches. The rankings of the Friedman test are listed in Table 6, GGA gets the best ranking, RMHC the worst. We continue the evaluation with the Holm post-hoc procedure, the adjusted p-values are listed in the last column of Table 6. The low values show that GGA significantly outperforms both RMHC and FRPS.
- **Reduction**
  In Table 5 the average reduction rates of GGA, RMHC and FRPS are compared in the second row. Full results can be found in Table A.2 in Appendix A. The reduction rate of RMHC is 90 percent, which is a logical result as the percentage of removed instances is fixed to 90 percent for RMHC. This reduction rate for RMHC is about the same as for GGA, but GGA clearly outperforms RMHC with respect to accuracy, which means that GGA is better able to select the right 10 percent of instances. The reduction rate of FRPS is low, only 10 percent of the instances is removed.
- **Running time** The last row in Table 5 shows the average running times of the GGA, RMHC and FRPS, the running times per dataset are listed in Table A.3. GGA is the slowest method, FRPS is much faster and RMHC is intermediate.

Table 6
Aligned Friedman rankings of GGA, FRPS and RMHC and adjusted p-values resulting from the Holm post-hoc procedure comparing GGA to FRPS and RMHC.

	Friedman ranking	Adjusted p-value
RMHC	2.9302	<0.000001
FRPS	1.8605	0.002533
GGA	1.2093	–

5.2.2.2. Comparison of GGA with state-of-the-art filter TSS techniques. In this part we will compare our proposal, GGA for SVM, with the state of the art filter TSS techniques and with the baseline method where no TSS preprocessing is applied. Again, there are three aspects that should be taken into account.

- **Accuracy**
  In Table 7, the average accuracy over all datasets can be found for the baseline, our proposal GGA and the filter TSS methods, full results are listed in Table A.4 in Appendix A. Our proposal GGA clearly outperforms the other methods. It is clear that none of the filter TSS methods outperforms the baseline on average. Some of the filter methods, like MOCS and NCNEdit do not deteriorate the accuracy too much. The two methods SNG and MCIS that were designed to preprocess SVM have lower accuracy rates, which is a logical result as these methods were mainly designed to improve the efficiency of SVMs rather than to improve their accuracy.

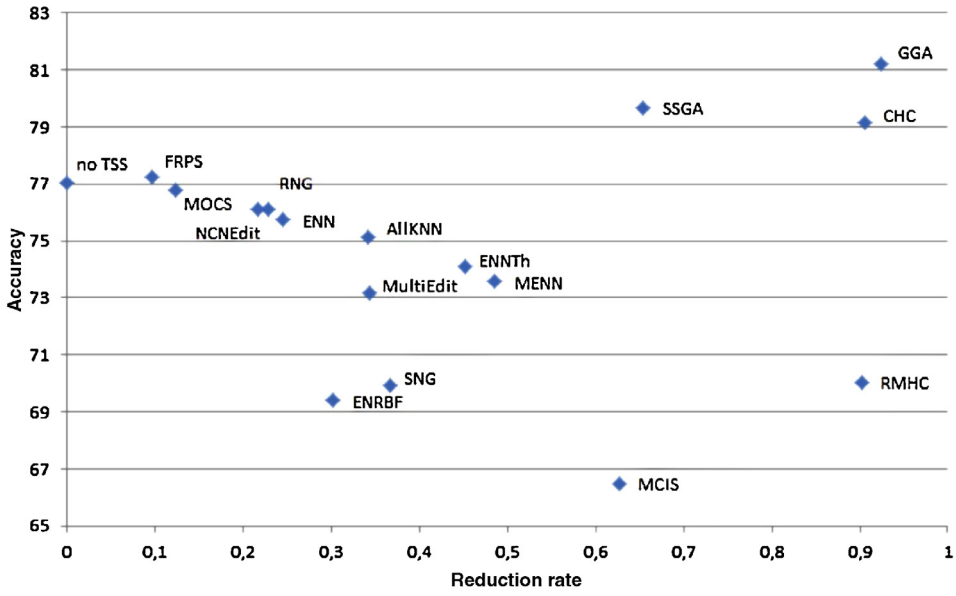

Fig. 6. Accuracy of the TSS filter methods in function of the reduction rate.

Table 7
Average accuracy, reduction rate and running time of the baseline (NO TSS), our proposal GGA and filter TSS methods.

Algorithm	Accuracy	Reduction	Running time
NO TSS	77.04	0	0
GGA	81.21	0.9246	3198.7
RNG	76.14	0.2276	0.6587
MOCS	76.80	0.1222	0.6874
ENN	75.73	0.2446	0.6372
MultiEdit	73.19	0.3440	0.7591
MENN	73.57	0.4857	1.0714
NCNEdit	76.10	0.2163	2.1109
ENRBF	69.40	0.3021	1.3644
ENNTh	74.12	0.4517	0.8556
AllKNN	75.13	0.3410	1.0947
SNG	69.93	0.3594	0.1700
MCIS	66.50	0.6266	0.6971

To test if these differences are significant, we carry out the Friedman test. The p-value is smaller than 0.000001, meaning that there are indeed significant differences between the considered methods. The Friedman aligned ranks are listed in Table 8. GGA has the highest ranking, followed by NCNEdit and the baseline method. We carry out the Holm post-hoc procedure to verify if GGA significantly outperforms the other methods. The corresponding adjusted p-values are listed in the last column of Table 8. The low values show that GGA outperforms all other approaches significantly with respect to accuracy, and that it significantly improves SVMs.

Table 8
Average Friedman rankings of the filter TSS algorithms, the baseline and our proposal GGA, and adjusted p-values resulting from the Holm post-hoc procedure comparing GGA to the remaining algorithms.

Algorithm	Ranking	Adjusted p-value
MCIS	10.0233	<0.000001
ENRBF	9.6744	<0.000001
SNG	8	<0.000001
MultiEdit	7.8605	<0.000001
ENNTh	7.8256	<0.000001
MENN	7.814	<0.000001
AllKNN	6.9302	<0.000001
RNG	6.6047	<0.000001
ENN	6.5814	<0.000001
MOCS	5.8721	0.000023
NO TSS	5.8721	0.000023
NCNEdit	5.8256	0.000023
GGA	2.1163	–

- **Reduction**
  The second aspect to evaluate is reduction. The average reduction rates of the filter methods and the proposal GGA is listed in Table 7, full results are available in Table A.5 in the Appendix A. The reduction rates of the filter methods are rather low, only MCIS manages to remove 65 percent of the instances. In Fig. 6 we plot the accuracy rate of the TSS methods in function of the reduction rate. It is remarkable that our proposal GGA achieves high accuracy rates and high reduction rates at the same time. For the filter methods, a downwards trend can be observed: the more instances removed, the lower the accuracy. This suggests that these basic filter approaches are not able to select the right instances for removal.
- **Running time**
  From the analysis above we learn that GGA is superior with respect to accuracy and reduction. Unfortunately this comes with a higher computational time. In Table 7 we show the average running time of the state-of-the-art methods and our proposal over all datasets, full results can be found in Table A.6. The GGA algorithm is remarkably slower than the filter methods.

We can conclude that our proposal GGA is able to improve the accuracy of the baseline and the state-of-the-art TSS methods significantly. It removes about 90 percent of the data and improves the accuracy of SVMs by 4 percent, unfortunately this comes at a higher computational cost.

6. Conclusion

In this paper, we improve the accuracy of SVMs by means of wrapper TSS methods. To this aim, we have adapted five wrapper TSS techniques that were originally proposed for KNN classification for SVMs. Instead of evaluating candidate subsets using the KNN classifier, we use SVMs to evaluate them. By means of an experimental evaluation we select the best wrapper approach, which is the evolutionary method GGA, and show that it significantly outperforms not only the remaining wrappers, but also the original SVM algorithm as well as the state-of-the-art TSS techniques. We hypothesize that the good behavior of GGA is due to a clever combination of introducing sufficient variation in candidate solutions,

and at the same time sufficiently respecting the original structure, defined by the SVM's support vectors, of the classification problems.

Acknowledgements

This work was partially supported by the Spanish Ministry of Science and Technology under the project TIN2011-28488 and the Andalusian Research Plans P12-TIC-2958, P11-TIC-7765 and P10-TIC-6858, and by project PYR-2014-8 of the Genil Program of CEI BioTic GRANADA.

Appendix A.

See Tables A.1–A.6.

Table A.1
Accuracy of the wrapper TSS methods.

	NO TSS	GGA	CHC	SSGA	RMHC	FRPS
appendicitis	87.82	87.32	83.36	87.95	79.77	87.82
australian	85.51	87.26	85.51	87.23	83.74	85.80
automobile	63.07	74.21	66.93	70.23	46.75	62.49
balance	90.25	89.85	90.08	89.01	69.64	90.25
bands	68.81	69.19	67.06	69.28	60.37	69.10
breast	74.73	76.25	74.12	76.53	72.60	73.99
bupa	66.93	64.06	65.82	66.47	61.16	66.88
car	90.05	91.15	93.00	91.69	82.34	90.28
cleveland	56.99	65.32	56.23	62.85	56.72	56.64
contraceptive	48.27	51.54	48.67	50.98	45.09	49.69
crx	86.30	86.80	85.87	87.41	84.45	86.92
dermatology	96.62	97.30	93.29	97.42	77.41	96.34
ecoli	78.00	80.92	78.89	80.42	68.32	78.30
flare	75.24	76.66	74.48	76.56	73.14	74.39
german	74.80	77.04	73.80	77.57	70.51	75.00
glass	65.30	67.50	63.03	62.41	48.49	63.77
haberman	74.17	75.74	75.12	75.05	70.52	74.17
hayesroth	81.88	86.39	80.63	82.15	55.56	81.25
heart	82.59	83.21	80.00	82.76	79.01	82.22
hepatitis	86.08	87.64	83.59	88.33	83.47	87.90
housevotes	94.54	94.30	95.31	91.67	87.69	94.54
ionosphere	89.17	90.85	90.63	91.04	76.89	89.17
iris	93.33	96.52	92.67	94.00	94.30	92.67
led7digit	73.40	76.93	71.80	76.82	65.00	73.40
lymphography	79.08	90.39	81.34	86.41	72.75	79.08
mammographic	82.90	83.94	84.13	82.90	82.37	82.28
monk2	97.27	97.30	98.40	96.94	88.79	97.72
movementlibras	56.94	71.67	68.61	69.14	44.91	57.50
newthyroid	96.80	96.43	96.30	96.07	90.65	96.32
pima	75.80	77.29	76.57	77.17	74.10	75.93
postoperative	64.58	72.54	70.00	68.97	58.75	67.92
saheart	71.64	73.26	70.57	72.51	69.02	72.29
sonar	70.69	78.31	76.43	78.26	68.70	70.69
spectfheart	79.49	83.10	76.44	82.31	76.65	80.23
tae	49.71	65.49	51.13	56.36	42.68	50.33
tictactoe	74.94	97.11	97.91	94.64	70.34	74.52
vehicle	49.67	70.55	72.80	58.52	46.15	49.78
vowel	34.75	51.76	67.58	45.34	26.58	36.77
wdbc	94.37	96.97	97.36	95.14	92.99	94.72
wine	96.08	97.07	96.67	96.88	89.20	97.19
wisconsin	96.37	97.59	97.09	96.89	95.82	96.80
yeast	59.30	60.33	58.02	60.53	55.71	59.50
zoo	98.50	97.14	95.67	94.94	72.50	98.50
average	**77.04**	**81.21**	**79.14**	**79.67**	**70.04**	**77.23**

Table A.2
Reduction of the wrapper TSS methods.

	GGA	CHC	SSGA	RMHC	FRPS
appendicitis	0.9528	0.9780	0.6792	0.9057	0.2275
australian	0.9504	0.9338	0.6654	0.9002	0.2432
automobile	0.8288	0.7876	0.6094	0.9022	0.0175
balance	0.8562	0.8030	0.6016	0.9004	0.0009
bands	0.9382	0.9202	0.6259	0.9014	0.0521
breast	0.9515	0.9426	0.6486	0.9017	0.2643
bupa	0.9327	0.9195	0.6097	0.9002	0.0000
car	0.9002	0.7717	0.6208	0.9003	0.0000
cleveland	0.9285	0.9330	0.6293	0.9027	0.0580
contraceptive	0.9155	0.7723	0.6154	0.9004	0.0875
crx	0.9456	0.9496	0.6835	0.9010	0.1661
dermatology	0.9348	0.9308	0.6735	0.9007	0.0146
ecoli	0.9322	0.9418	0.6455	0.9008	0.0331
flare	0.9392	0.8771	0.6572	0.9006	0.3817
german	0.9226	0.8707	0.6663	0.9000	0.0199
glass	0.9065	0.9335	0.6106	0.9013	0.0047

Table A.2 (*Continued*)

	GGA	CHC	SSGA	RMHC	FRPS
haberman	0.9622	0.9706	0.6877	0.9020	0.4147
hayesroth	0.8694	0.9174	0.6347	0.9028	0.1479
heart	0.9539	0.9654	0.7251	0.9012	0.2807
hepatitis	0.9292	0.9417	0.6347	0.9056	0.1306
housevotes	0.9569	0.9655	0.6461	0.9018	0.0038
ionosphere	0.9437	0.9326	0.6879	0.9019	0.0199
iris	0.9452	0.9756	0.7393	0.9037	0.0304
led7digit	0.9422	0.9289	0.6638	0.9000	0.0000
lymphography	0.9017	0.8904	0.6674	0.9024	0.0601
mammographic	0.9554	0.9124	0.6862	0.9008	0.2681
monk2	0.9324	0.9434	0.6708	0.9020	0.2132
movementlibras	0.8627	0.8074	0.5957	0.9012	0.0151
newthyroid	0.9504	0.9695	0.7142	0.9018	0.0150
pima	0.9420	0.9261	0.6545	0.9002	0.0165
postoperative	0.9515	0.9553	0.6386	0.9106	0.1903
saheart	0.9456	0.9473	0.6159	0.9014	0.0031
sonar	0.9119	0.9327	0.6341	0.9038	0.0027
spectfheart	0.9459	0.9347	0.6259	0.9001	0.0017
tae	0.8521	0.8543	0.6350	0.9043	0.1038
tictactoe	0.9138	0.9019	0.6737	0.9003	0.0658
vehicle	0.9023	0.8060	0.5956	0.9002	0.0537
vowel	0.8536	0.6311	0.5338	0.9001	0.0000
wdbc	0.9598	0.9551	0.7184	0.9004	0.2000
wine	0.9438	0.9607	0.7141	0.9001	0.0799
wisconsin	0.9658	0.9684	0.7108	0.9008	0.2151
yeast	0.9248	0.8590	0.6822	0.9004	0.0180
zoo	0.9054	0.9087	0.6931	0.9043	0.0033
average	**0.9246**	**0.9053**	**0.6540**	**0.9017**	**0.0959**

Table A.3
Time of the wrapper TSS methods in seconds.

	GGA	CHC	SSGA	RMHC	FRPS
appendicitis	424.2	1151.3	410.5	395.2	4.7
australian	999.1	8501.3	1755.4	732.5	100.9
automobile	2986.5	10056.0	3188.0	2600.7	18.3
balance	1560.9	9750.9	2221.4	1109.6	81.7
bands	749.1	4443.3	867.9	510.1	19.7
breast	721.1	1900.3	945.9	429.5	11.9
bupa	474.7	3518.9	635.2	373.3	13.7
car	7127.6	14990.6	9921.8	4068.1	1215.5
cleveland	2292.2	8624.6	2018.1	1801.1	35.1
contraceptive	3633.0	13149.1	7402.0	1941.2	972.2
crx	1344.3	5477.5	1863.8	1085.3	107.2
dermatology	5317.5	12580.7	6233.9	4150.1	102.9
ecoli	5024.9	12148.9	5090.1	3220	68.3
flare	8309.9	13853.8	12145.0	5542.2	104.5
german	2094.0	11150.8	3403.5	1251.7	460.1
glass	2957.8	10588.5	2873.1	1713.5	21.5
haberman	553.8	952.5	638.9	400.6	8.2
hayesroth	936.3	3257.8	898.9	649.0	3.2
heart	436.8	3412.2	647.0	404.7	10.3
hepatitis	368.8	2666.5	436.0	224.9	1.6
housevotes	585.5	1745.5	685.9	443.9	4.4
ionosphere	756.9	2809.4	965.4	665.9	24.6
iris	902.8	2689.0	862.9	662.9	5.4
led7digit	9689.9	13830.7	8977.7	9410.2	19.0
lymphography	1209.9	5644.3	1353.2	798.5	8.6
mammographic	1121.3	6454.8	1556.1	806.5	51.2
monk2	633.9	4327.3	823.9	461.7	25.8
movementlibras	25309.8	19817.2	23640.5	17640.1	341.8
newthyroid	865.6	4456.4	859.1	778.5	7.7
pima	1592.1	5394.5	2369.3	1222.1	141.8
postoperative	474.8	1248.5	54	321.9	2.0
saheart	605.1	2793.1	933.5	510.3	32.2
sonar	747.3	5114.9	941.8	624.1	10.9
spectfheart	636.2	5049.6	773.0	457.0	10.2
tae	795.5	6215.1	911.0	836.4	4.9
tictactoe	2439.6	9999.8	2781.2	1429.3	253.6
vehicle	3120.8	11553.8	4698.1	2144.0	393.1
vowel	18086.0	21203.1	27232.3	10838.0	1713.2
wdbc	920.8	6484.7	1196.1	768.3	62.0
wine	816.1	6704.9	771.6	717.5	7.3
wisconsin	704.9	5168.6	792.1	521.7	28.1
yeast	12999.3	12715.2	14811.4	10680.3	2338.4
zoo	4215.6	9519.9	3592.4	1440.6	5.4
average	**3198.7**	**7514.3**	**3841.4**	**2250.8**	**205.9**

Table A.4
Accuracy of the baseline (NO TSS), our proposal GGA and the state-of-the-art filter TSS methods.

	NO TSS	GGA	RNG	MOCS	ENN	MultiEdit	MENN	NCNEdit	ENRBF	ENNTh	AllKNN	SNG	MCIS
appendicitis	87.82	87.32	87.82	87.82	87.82	87.74	85.85	88.68	80.19	85.85	87.74	90.00	65.09
australian	85.51	87.26	85.65	85.36	86.23	87.10	87.54	85.22	86.52	87.54	87.25	85.59	88.68
automobile	63.07	74.21	58.50	60.14	46.46	44.65	49.69	51.57	43.40	44.03	44.65	75.00	70.87
balance	90.25	89.85	89.76	87.35	89.60	87.52	87.20	87.68	87.68	87.04	86.88	75.16	92.28
bands	68.81	69.19	67.52	69.25	67.42	66.58	67.67	69.32	63.01	67.95	66.30	70.28	40.00
breast	74.73	76.25	73.39	73.65	74.02	75.09	74.37	72.92	74.01	74.37	75.09	74.07	68.23
bupa	66.93	64.06	63.49	64.23	64.66	62.03	60.00	66.96	57.97	60.00	64.64	65.29	42.32
car	90.05	91.15	89.06	89.29	90.22	87.21	86.40	90.22	70.02	87.15	87.27	38.67	77.22
cleveland	56.99	65.32	56.36	56.34	55.91	56.23	55.22	58.25	57.24	52.19	54.21	47.33	57.98
contraceptive	48.27	51.54	47.05	48.13	48.06	46.84	47.59	50.10	43.72	46.98	48.34	59.80	69.05
crx	86.30	86.80	86.81	86.30	86.93	86.52	86.37	86.98	86.83	86.37	87.29	86.41	66.31
dermatology	96.62	97.30	96.34	96.35	96.90	96.65	95.25	97.21	96.09	95.25	96.37	77.50	95.92
ecoli	78.00	80.92	78.89	79.51	80.11	74.40	75.60	78.27	42.56	76.19	78.57	53.24	60.69
are	75.24	76.66	75.42	74.77	71.39	67.26	60.51	72.80	60.13	71.86	69.79	44.11	72.21
german	74.80	77.04	74.40	74.70	73.30	73.20	74.20	74.40	70.00	74.20	73.90	73.10	70.90
glass	65.30	67.50	57.07	62.47	56.20	59.35	61.21	60.28	37.38	55.61	52.80	56.36	44.20
haberman	74.17	75.74	74.43	74.83	74.16	74.84	71.90	73.53	73.53	72.55	73.86	73.33	26.47
hayesroth	81.88	86.39	68.13	85.00	65.63	47.50	65.00	50.00	38.75	56.88	61.25	58.13	51.18
heart	82.59	83.21	81.85	82.22	81.48	82.96	83.33	83.33	82.96	82.59	82.59	79.26	71.11
hepatitis	86.08	87.64	82.99	79.92	82.65	88.75	76.25	85.00	82.50	77.50	82.50	87.14	75.00
housevotes	94.54	94.30	92.35	93.95	92.01	91.81	91.38	93.10	91.81	91.38	91.81	95.00	88.79
ionosphere	89.17	90.85	88.60	88.90	87.17	83.19	88.32	88.60	79.20	88.32	87.75	72.35	86.04
iris	93.33	96.52	90.00	93.33	92.00	91.33	91.33	91.33	89.33	91.33	91.33	45.00	68.75
led7digit	73.40	76.93	71.40	73.60	56.60	27.40	17.60	72.20	74.60	37.80	51.00	85.40	53.14
lymphography	79.08	90.39	81.87	82.50	81.20	81.76	78.38	80.41	82.43	80.41	80.41	50.67	62.03
mammographic	82.90	83.94	83.50	82.17	82.66	79.28	79.88	79.52	79.16	79.52	79.28	76.71	57.47
monk2	97.27	97.30	97.27	97.27	97.27	82.41	79.17	86.11	78.94	76.39	84.49	80.95	74.54
movementlibras	56.94	71.67	59.44	59.17	61.94	48.89	57.50	58.06	59.72	58.06	62.78	85.28	23.51
newthyroid	96.80	96.43	97.25	96.80	95.87	96.28	95.35	97.67	69.77	95.35	96.28	46.36	98.67
pima	75.80	77.29	76.31	75.80	75.80	76.04	76.04	76.04	65.10	76.04	76.43	76.32	65.10
postoperative	64.58	72.54	71.25	70.14	67.64	70.11	71.26	68.97	71.26	71.26	70.11	52.22	67.01
saheart	71.64	73.26	73.17	73.17	71.87	72.08	73.16	72.29	65.37	73.16	71.65	70.87	65.37
sonar	70.69	78.31	68.81	70.71	70.71	70.19	72.60	69.71	69.23	74.52	71.63	67.50	59.62
spectfheart	79.49	83.10	79.46	79.84	75.30	75.28	77.53	79.03	79.40	77.53	77.53	85.77	71.91
tae	49.71	65.49	45.67	46.37	47.67	48.34	48.34	54.97	48.34	50.33	55.63	66.00	60.87
tictactoe	74.94	97.11	74.84	70.66	88.41	73.80	70.35	79.44	72.96	79.33	81.84	69.37	65.34
vehicle	49.67	70.55	56.27	50.37	55.10	54.37	56.38	54.96	45.04	56.50	56.74	66.07	67.29
vowel	34.75	51.76	37.68	36.36	36.77	38.08	51.21	45.56	27.07	50.30	44.75	83.50	27.70
wdbc	94.37	96.97	94.37	94.55	94.72	94.73	94.55	94.38	87.35	94.55	94.55	92.86	86.64
wine	96.08	97.07	94.97	96.08	96.60	97.75	96.63	96.07	95.51	96.63	96.63	32.94	74.47
wisconsin	96.37	97.59	97.09	97.25	97.24	96.93	97.36	97.22	92.68	97.36	96.78	94.18	89.90
yeast	59.30	60.33	58.69	59.84	59.37	56.74	58.83	58.96	31.27	58.83	58.76	67.52	48.59
zoo	98.50	97.14	88.72	96.14	93.25	88.12	89.11	95.05	94.06	90.10	91.09	74.55	90.99
average	**77.04**	**81.21**	**76.14**	**76.80**	**75.73**	**73.19**	**73.57**	**76.10**	**69.40**	**74.12**	**75.13**	**69.93**	**66.50**

Table A.5
Reduction of our proposal GGA and the state-of-the-art filter TSS methods.

	GGA	RNG	ModelCS	ENN	MultiEdit	MENN	NCNEdit	ENRBF	ENNTh	AllKNN	SNG	MCIS
appendicitis	0.9528	0.1624	0.1038	0.1593	0.2390	0.3459	0.1834	0.1981	0.3459	0.2495	0.3742	0.2642
australian	0.9504	0.1599	0.1032	0.1514	0.2163	0.3597	0.1818	0.1398	0.3572	0.2707	0.0989	0.4905
automobile	0.8288	0.3328	0.1181	0.3845	0.5388	0.6639	0.2460	0.4948	0.5542	0.4598	0.0678	0.6088
balance	0.8562	0.1284	0.1483	0.1559	0.2004	0.4123	0.1598	0.1150	0.3755	0.2706	0.1755	0.4669
bands	0.9382	0.3281	0.1294	0.2909	0.4463	0.6642	0.3020	0.3699	0.6627	0.4661	0.0827	0.6122
breast	0.9515	0.2535	0.1737	0.3181	0.3971	0.6237	0.3101	0.2631	0.5977	0.4188	0.0958	0.6462
bupa	0.9327	0.3855	0.1952	0.3775	0.5121	0.7823	0.3591	0.4203	0.7823	0.5620	0.0184	0.4031
car	0.9002	0.0327	0.0876	0.0775	0.1525	0.2793	0.0556	0.2998	0.2436	0.1575	0.0646	0.7081
cleveland	0.9285	0.4179	0.2836	0.4471	0.5503	0.6925	0.4280	0.4426	0.6667	0.5350	0.0346	0.5181
contraceptive	0.9155	0.5019	0.3134	0.5498	0.7011	0.8853	0.5539	0.5500	0.8759	0.7185	0.4778	0.6403
crx	0.9456	0.1513	0.0966	0.1560	0.2168	0.3582	0.1679	0.1377	0.3563	0.2421	0.5886	0.5273
dermatology	0.9348	0.0658	0.0205	0.0317	0.0624	0.1024	0.0332	0.0410	0.1018	0.0655	0.4906	0.4201
ecoli	0.9322	0.2133	0.1151	0.1958	0.3039	0.3919	0.1898	0.5744	0.3872	0.2761	0.0562	0.5141
flare	0.9392	0.2652	0.0904	0.3809	0.5427	0.6360	0.3155	0.4930	0.6021	0.4726	0.3587	0.4805
german	0.9226	0.279	0.1480	0.2983	0.3830	0.5994	0.2870	0.3000	0.5916	0.4122	0.1259	0.3424
glass	0.9065	0.3349	0.1537	0.3214	0.4678	0.5711	0.2861	0.6776	0.5587	0.4216	0.1175	0.4250
haberman	0.9622	0.358	0.1895	0.3032	0.3929	0.6020	0.3304	0.2647	0.5922	0.4873	0.2330	0.6993
hayesroth	0.8694	0.3903	0.1688	0.7313	0.7556	0.9271	0.2493	0.5854	0.8840	0.8118	0.2948	0.7124
heart	0.9539	0.2091	0.1292	0.2123	0.2584	0.4663	0.2148	0.1728	0.4531	0.3420	0.0962	0.5014
hepatitis	0.9292	0.1835	0.0876	0.1707	0.1833	0.3306	0.1792	0.1639	0.2861	0.2514	0.5963	0.6905
housevotes	0.9569	0.0771	0.0330	0.0762	0.1006	0.2275	0.0474	0.0824	0.1327	0.1068	0.6456	0.4173
ionosphere	0.9437	0.1352	0.0639	0.1421	0.1988	0.2314	0.0801	0.2953	0.2314	0.1830	0.4091	0.7072
iris	0.9452	0.0489	0.0237	0.0474	0.0489	0.0911	0.0393	0.1267	0.0911	0.0519	0.0323	0.6430

Table A.5 (*Continued*)

	GGA	RNG	ModelCS	ENN	MultiEdit	MENN	NCNEdit	ENRBF	ENNTh	AllKNN	SNG	MCIS
led7digit	0.9422	0.2742	0.0353	0.5660	0.8609	0.9953	0.3569	0.2609	0.8729	0.7093	0.5052	0.7070
lymphography	0.9017	0.2162	0.1622	0.2146	0.3281	0.4685	0.1892	0.1547	0.4467	0.3221	0.2842	0.7437
mammographic	0.9554	0.2016	0.0963	0.1988	0.2778	0.4993	0.2410	0.2063	0.4661	0.3268	0.0641	0.6342
monk2	0.9324	0.0062	0.1780	0.0411	0.2976	0.7127	0.1852	0.2060	0.5198	0.3683	0.5741	0.7613
movementlibras	0.8627	0.2373	0.0855	0.2275	0.4935	0.4855	0.1429	0.2410	0.4025	0.2923	0.6645	0.7669
newthyroid	0.9504	0.0512	0.0145	0.0579	0.0863	0.1142	0.0403	0.3023	0.1142	0.0677	0.6265	0.7889
pima	0.942	0.2783	0.1645	0.2603	0.3462	0.5642	0.2938	0.3490	0.5642	0.3915	0.5508	0.7224
postoperative	0.9515	0.3142	0.2299	0.4190	0.4700	0.9144	0.4189	0.2874	0.8519	0.5568	0.3373	0.6313
saheart	0.9456	0.3259	0.1902	0.3141	0.4173	0.6407	0.3348	0.3463	0.6407	0.4529	0.5801	0.6593
sonar	0.9119	0.1886	0.0529	0.1699	0.3114	0.3141	0.1218	0.2078	0.2933	0.2110	0.6522	0.8536
spectfheart	0.9459	0.2992	0.1365	0.2863	0.3766	0.5476	0.2684	0.2060	0.5476	0.4228	0.4930	0.6642
tae	0.8521	0.5467	0.3098	0.5857	0.7307	0.8565	0.3863	0.5335	0.8293	0.6961	0.7058	0.8959
tictactoe	0.9138	0.2416	0.0723	0.2284	0.2688	0.6439	0.2040	0.3397	0.2951	0.2810	0.6569	0.6158
vehicle	0.9023	0.2972	0.1659	0.2931	0.4363	0.5775	0.2866	0.5074	0.5774	0.4065	0.5034	0.6062
vowel	0.8536	0.1211	0.0020	0.0312	0.3031	0.1287	0.0184	0.5978	0.1219	0.0393	0.1364	0.6419
wdbc	0.9598	0.0387	0.0215	0.0307	0.0576	0.0976	0.0363	0.1621	0.0976	0.0660	0.1102	0.7238
wine	0.9438	0.0599	0.0144	0.0337	0.0593	0.0893	0.0356	0.0543	0.0893	0.0512	0.5607	0.8966
wisconsin	0.9658	0.0333	0.0241	0.0307	0.0431	0.0779	0.0327	0.0750	0.0756	0.0604	0.5654	0.6633
yeast	0.9248	0.4715	0.2830	0.4668	0.5675	0.7846	0.4613	0.6860	0.7845	0.6195	0.5493	0.8204
zoo	0.9054	0.1683	0.0384	0.0815	0.1914	0.1298	0.0473	0.0572	0.1012	0.0880	0.7973	0.7071
average	**0.9246**	**0.2276**	**0.1222**	**0.2446**	**0.3440**	**0.4857**	**0.2163**	**0.3021**	**0.4517**	**0.3410**	**0.3594**	**0.6266**

Table A.6
Running time of our proposal GGA and the state-of-the-art filter TSS methods, in seconds.

	GGA	RNG	ModelCS	ENN	MultiEdit	MENN	NCNEdit	ENRBF	ENNTh	AllKNN	SNG	MCIS
appendicitis	424.2	0.0799	0.0855	0.0802	0.0700	0.0900	0.2500	0.0900	0.2200	0.1500	0.0382	0.1260
australian	999.1	0.5217	0.5569	0.5270	2.3600	2.8300	4.1400	2.8100	2.0900	2.6500	0.1506	2.2180
automobile	2986.5	0.0955	0.1262	0.0881	0.5000	0.4600	1.1700	0.4700	0.2600	0.3500	0.0456	0.1000
balance	1560.9	0.4903	0.4791	0.4748	0.3300	0.6000	1.4800	0.8300	0.4200	0.5400	0.3300	0.2333
bands	749.1	0.2207	0.2860	0.2329	0.3000	0.2600	1.1800	0.3500	0.3100	0.4400	0.0400	0.4930
breast	721.1	0.1861	0.2060	0.1700	0.3300	0.5600	1.4100	0.5800	0.2900	0.4400	0.0347	0.4480
bupa	474.7	0.1908	0.2499	0.1933	2.3600	3.4800	7.0100	5.2000	2.6600	4.3800	0.0396	0.5220
car	7127.6	15.0440	14.1900	14.3700	0.1800	0.4700	1.1200	0.5900	0.4100	0.4600	0.2976	0.8975
cleveland	2292.2	0.1556	0.1915	0.1478	1.2100	2.2800	6.7100	3.9800	2.0300	2.6300	0.0507	0.1680
contraceptive	3633.0	0.6603	0.9102	0.5968	2.1100	2.8300	3.7800	2.1000	2.1600	2.4600	0.2977	0.6167
crx	1344.3	0.4988	0.5309	0.4960	1.6200	1.5400	2.3800	1.5400	1.5500	1.2800	0.1367	1.9570
dermatology	5317.5	0.3010	0.3156	0.3120	0.4300	0.5700	1.1600	0.8700	0.4700	0.5900	0.0791	0.3700
ecoli	5024.9	0.2379	0.2676	0.2432	2.1100	2.9600	5.4000	2.7400	2.0900	2.4900	0.1102	0.9000
flare	8309.9	0.7050	0.8727	0.5940	0.3200	0.3400	1.1100	0.6700	0.3800	0.4600	0.2281	0.3000
german	2094.0	0.6489	0.7668	0.6315	0.2200	0.3600	0.8900	0.9100	0.3000	0.4200	0.2051	2.1240
glass	2957.8	0.1281	0.1630	0.1307	0.1700	0.0800	0.3700	0.1700	0.1100	0.1200	0.0618	0.8300
haberman	553.8	0.1768	0.2232	0.1919	0.2400	0.4100	1.0900	0.4800	0.3100	0.3200	0.0351	0.2870
hayesroth	936.3	0.0878	0.1197	0.0387	0.1000	0.0900	0.3000	0.1000	0.1100	0.1500	0.0461	0.1000
heart	436.8	0.1922	0.2116	0.1914	0.1400	0.3800	1.0300	0.3200	0.2900	0.4700	0.0364	0.3200
hepatitis	368.8	0.0588	0.0657	0.0597	0.1200	0.2000	0.2500	0.1500	0.1500	0.1600	0.0337	0.6000
housevotes	585.5	0.1927	0.2019	0.1929	0.3000	0.7500	1.2300	0.8400	0.6900	0.5400	0.0549	0.1270
ionosphere	756.9	0.2732	0.2957	0.2710	0.5500	0.8800	1.5200	1.5000	0.7400	1.1100	0.1252	1.1830
iris	902.8	0.1284	0.1318	0.1286	0.6900	0.9700	2.7800	1.0900	0.6500	0.9000	0.0833	0.1100
led7digit	9689.9	0.3266	0.4341	0.1953	0.6400	1.0900	2.3600	1.4000	0.6200	0.9800	0.1711	0.4790
lymphography	1209.9	0.1044	0.1116	0.1046	0.3500	0.6200	1.5900	0.5800	0.6200	0.6800	0.0613	0.2400
mammographic	1121.3	0.5964	0.6751	0.5985	1.0600	0.7300	1.8900	0.9300	0.5900	1.0400	0.0888	0.3610
monk2	633.9	0.3864	0.3196	0.3728	1.1600	1.3100	2.7100	1.0600	1.2400	1.1500	0.1069	1.1930
movementlibras	25309.8	0.2471	0.2963	0.2503	0.1000	0.1500	0.4900	0.1400	0.1200	0.1600	0.1367	0.7100
newthyroid	865.6	0.1836	0.1907	0.1823	0.7200	1.7700	3.6300	2.3700	1.1500	1.4900	0.0675	0.1667
pima	1592.1	0.4988	0.5775	0.5113	0.1000	0.1400	0.5400	0.1100	0.1200	0.1900	0.1823	0.6850
postoperative	474.8	0.0537	0.0603	0.0455	2.1800	2.4000	4.8900	2.9300	2.0600	2.5800	2.2346	0.0800
saheart	605.1	0.2803	0.3367	0.2852	2.0400	2.5900	5.0500	3.7200	2.0600	3.2900	0.1105	0.7900
sonar	747.3	0.1519	0.1773	0.1554	0.1700	0.2400	1.0900	0.3700	0.1900	0.4200	0.0503	0.5730
spectfheart	636.2	0.1684	0.2075	0.1715	0.5800	1.4600	2.3900	1.4700	0.9500	1.1800	0.0788	0.7760
tae	795.5	0.0616	0.0938	0.0563	1.4600	2.3400	5.3900	5.0200	1.7600	2.9500	0.0402	0.1067
tictactoe	2439.6	0.6539	0.7999	0.6653	0.1100	0.0900	0.3300	0.1200	0.1000	0.1100	0.1667	0.9290
vehicle	3120.8	0.5351	0.6351	0.5382	0.4100	0.9100	1.1400	1.1400	0.5300	0.7500	0.2353	1.7850
vowel	18086.0	0.7831	0.8892	0.8632	1.6300	2.7100	2.6000	4.3400	2.3100	3.2500	0.1464	2.0191
wdbc	920.8	0.4923	0.5011	0.4964	1.1300	1.5100	1.9000	1.5800	1.3300	1.3500	0.1278	0.6040
wine	816.1	0.1506	0.1579	0.1548	0.1300	0.2000	0.8800	0.2400	0.2000	0.1700	0.0453	0.3433
wisconsin	704.9	0.5942	0.5999	0.5958	0.2100	0.5300	0.8600	0.7000	0.5300	0.3300	0.1397	1.3200
yeast	12999.3	0.7058	0.9576	0.7121	0.0400	0.1000	0.1200	0.1200	0.1100	0.1000	0.4954	1.6860
zoo	4215.6	0.0756	0.0874	0.0835	1.6600	1.7900	3.1600	1.9500	1.5100	1.3900	0.0638	0.0971
average	**3198.7**	**0.6587**	**0.6874**	**0.6372**	**0.7591**	**1.0714**	**2.1109**	**1.3644**	**0.8556**	**1.0947**	**0.1700**	**0.6971**

References

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

[2] J.R. Cano, F. Herrera, M. Lozano, Evolutionary stratified training set selection for extracting classification rules with trade-off precision-interpretability, Data Knowl. Eng. 60 (2007) 90–108.

[3] S. García, J. Luengo, F. Herrera, Data Preprocessing in Data Mining, Springer, 2015.

[4] J. Derrac, S. García, F. Herrera, Fuzzy nearest neighbor algorithms: taxonomy, experimental analysis and prospects, Inf. Sci. 260 (2014) 98–119.

[5] S. García, J. Derrac, J.R. Cano, F. Herrera, Prototype selection for nearest neighbor classification: taxonomy and empirical study, IEEE Trans. Pattern Anal. Mach. Intell. 34 (3) (2012) 417–435.

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

[7] M. Zechner, M. Granitzer, A competitive learning approach to instance selection for support vector machines, in: Knowledge Science Engineering and Management, Vol. 5914, 2009, pp. 146–157.

[8] L.I. Kuncheva, L.C. Jain, Nearest neighbor classifier: simultaneous editing and feature selection, Pattern Recogn. Lett. 20 (1999) 1149–1156.

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

[10] W. Ting-Fan, L. Chih-Jen, W.C. Ruby, Probability estimates for multi-class classification by pairwise coupling, J. Mach. Learn. Res. 5 (2004) 975–1005.

[11] M. Galar, A. Fernandez, E. Barrenechea, H. Bustince, F. Herrera, An overview of ensemble methods for binary classifiers in multi-class problems: experimental study on one-vs-one and one-vs-all schemes, Pattern Recogn. 44 (8) (2011) 1761–1776.

[12] T. Hastie, R. Tibshirani, Classification by pairwise coupling, Ann. Stat. 26 (2) (1998) 451–471.

[13] S.H. Park, J. Fürnkranz, Efficient pairwise classification, in: Proceedings of the 18th European Conference on Machine Learning, 2007, pp. 658–665.

[14] D.L. Wilson, Asymptotic properties of nearest neighbor rules using edited data, IEEE Trans. Syst. Man Cybern. 2 (3) (1972) 408–421.

[15] F. Vazquez, J. Sánchez, F. Pla, A stochastic approach to Wilsons' editing algorithm, in: 2nd Iberian Conference on Pattern Recognition and Image Analysis, 2005, pp. 35–42.

[16] I. Tomek, An experiment with the edited nearest-neighbor rule, IEEE Trans. Syst. Man Cybern. 6 (6) (1976) 448–452.

[17] K. Hattori, M. Takahashi, A new edited k-nearest neighbor rule in the pattern classification problem, Pattern Recogn. 32 (2000) 521–528.

[18] J.S. Sánchez, R. Barandela, A.I. Marqués, R. Alejo, J. Badenas, Analysis of new techniques to obtain quality training sets, Pattern Recogn. Lett. 24 (2003) 1015–1022.

[19] P. Devijver, On the editing rate of the multiedit algorithm, Pattern Recogn. Lett. 4 (1) (1986) 9–12.

[20] J.S. Sánchez, F. Pla, F.J. Ferri, Prototype selection for the nearest neighbour rule through proximity graphs, Pattern Recogn. Lett. 18 (1997) 507–513.

[21] C.E. Brodley, Recursive automatic bias selection for classifier construction, Mach. Learn. 20 (1995) 63–94.

[22] M. Grochowski, N. Jankowski, Comparison of instances selection algorithms, in: Artif. Intell. Soft Comput., 2004, pp. 598–603.

[23] J.R. Cano, F. Herrera, M. Lozano, Using evolutionary algorithms as instance selection for data reduction in KDD: an experimental study, IEEE Trans. Evol. Comput. 7 (6) (2003) 561–575.

[24] L. Eshelman, The CHC adaptive search algorithm: how to have safe search when engaging in nontraditional genetic recombination, Found. Genet. Algorithms (1991) 265–283.

[25] D.B. Skalak, Prototype and feature selection by sampling and random mutation hill climbing algorithms, in: Proceedings of the Eleventh International Conference on Machine Learning 6, 1994, pp. 293–301.

[26] N. Verbiest, C. Cornelis, F. Herrera, Frps: a fuzzy rough prototype selection method, Pattern Recogn. 46 (10) (2013) 2770–2782.

[27] D. Dubois, H. Prade, Rough fuzzy sets and fuzzy rough sets, Int. J. Gen. Syst. 17 (1990) 191–209.

[28] K. Bache, M. Lichman, UCI machine learning repository (2013). http://archive.ics.uci.edu/ml/.

[29] J. Cano, F. Herrera, M. Lozano, Stratification for scaling up evolutionary prototype selection, Pattern Recogn. Lett. 26 (2005) 953–963.

[30] N. García-Pedrajas, Constructing ensembles of classifiers by means of weighted instance selection, IEEE Trans. Neural Netw. 20 (2) (2009) 258–277.

[31] A. de Haro-García, N. García-Pedrajas, J.A.R. del Castillo, Large scale instance selection by means of federal instance selection, Data Knowl. Eng. 75 (2012) 58–77.

[32] A. de Haro-García, N. García-Pedrajas, A divide-and-conquer recursive approach for scaling up instance selection algorithms, Data Min. Knowl. Discov. 18 (3) (2009) 392–418.

[33] C. García-Osorio, A. de Haro-García, N. García-Pedrajas, Democratic instance selection: a linear complexity instance selection algorithm based on classifier ensemble concepts, Artif. Intell. 174 (2010) 410–441.

[34] J. Platt, Fast Training of Support Vector Machines Using Sequential Minimal Optimization, MIT Press, 1998, pp. 185–208.

[35] M. Friedman, The use of ranks to avoid the assumption of normality implicit in the analysis of variance, J. Am. Stat. Assoc. 32 (1937) 675–701.

[36] 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, Inf. Sci. 180 (2010) 2044–2064.

[37] S. García, F. Herrera, An extension on "statistical comparisons of classifiers over multiple data sets" for all pairwise comparisons, J. Mach. Learn. Res. 9 (2008) 2677–2694.

[38] J. Demsar, Statistical comparisons of classifiers over multiple data sets, J. Mach. Learn. Res. 7 (2006) 1–30.

[39] S. Holm, A simple sequentially rejective multiple test procedure, J. Stat. 6 (1979) 65–70.