Journal Pre-proof

SI(FS)^2: Fast simultaneous instance and feature selection for datasets with many features

Nicolás García-Pedrajas, Juan A. Romero del Castillo, Gonzalo Cerruela-García

PII: S0031-3203(20)30526-4
DOI: https://doi.org/10.1016/j.patcog.2020.107723
Reference: PR 107723

To appear in: *Pattern Recognition*

Received date: 6 May 2020
Revised date: 10 August 2020
Accepted date: 23 October 2020

Please cite this article as: Nicolás García-Pedrajas, Juan A. Romero del Castillo, Gonzalo Cerruela-García, SI(FS)^2: Fast simultaneous instance and feature selection for datasets with many features, *Pattern Recognition* (2020), doi: https://doi.org/10.1016/j.patcog.2020.107723

Highlights

- We propose a new simultaneous instance and feature selection algorithm.
- The method achieves better storage reduction and testing error than previous approaches.
- The method is scalable to datasets with millions of features.

SI(FS)^2: Fast simultaneous instance and feature selection for datasets with many features

Nicolás García-Pedrajas[a,*], Juan A. Romero del Castillo[a], Gonzalo Cerruela-García[a]

[a] *University of Córdoba, Campus de Rabanales, 14011 Córdoba (Spain)*

Abstract

Data reduction is becoming increasingly relevant due to the enormous amounts of data that are constantly being produced in many fields of research. Instance selection is one of the most widely used methods for this task. At the same time, most recent pattern recognition problems involve highly complex datasets with a large number of possible explanatory variables. For many reasons, this abundance of variables significantly hinders classification and recognition tasks. There are efficiency issues, too, because the speed of many classification algorithms is greatly improved when the complexity of the data is reduced. Thus, feature selection is also a widely used method for data reduction and for gaining an understanding of feature information.

Although most methods address instance and feature selection separately, the two problems are interwoven, and benefits are expected from performing these two tasks jointly. However, few algorithms have been proposed for simultaneously addressing the tasks of instance and feature selection. Furthermore, most of those methods are based on complex heuristics that are very difficult to scale up even to moderately large datasets. This paper proposes a new algorithm for dealing with many instances and many features simultaneously by

☆This work was supported in part by grant PID2019-109481GB-I00 of the Spanish Ministry of Science and Innovation and grant UCO-1264182 of the Junta de Andalucía Excellence in Research program and FEDER funds.

*Corresponding author

Email addresses: npedrajas@uco.es (Nicolás García-Pedrajas), aromero@uco.es (Juan A. Romero del Castillo), gcerruela@uco.es (Gonzalo Cerruela-García)

performing joint instance and feature selection using a simple heuristic search and several scaling-up mechanisms that can be successfully applied to datasets with millions of features and instances.

In the proposed method, a forward selection search is performed in the feature space jointly with the application of standard instance selection in a constructive subspace built stepwise. Several simplifications are adopted in the search to obtain a scalable method. An extensive comparison using 95 large datasets shows the usefulness of our method and its ability to deal with millions of instances and features simultaneously. The method is able to obtain better classification performance results than state-of-the-art approaches while achieving considerable data reduction.

Keywords: Instance selection, Feature selection, Evolutionary algorithms, k nearest neighbor rule.

---

1. Introduction

The overwhelming amounts of data that are available today in many fields of research pose new problems for data mining and knowledge discovery methods. These enormous volumes of data cause most existing algorithms to be inapplicable to many real-world problems. Two approaches have been used to address this problem: scaling up data mining algorithms and data reduction. However, scaling up a given algorithm is not always feasible. Data reduction consists of removing missing data, redundant data, information-poor data and/or erroneous data to obtain a tractable problem size. Among the different methods that can be used for data reduction, instance and feature selection are arguably the most widely used.

Instance selection consists of choosing a subset of the total available data to achieve the original purpose of the data mining application as successfully as this purpose would have been achieved with the whole dataset. Multiple variants of instance selection exist. Moreover, in real-world situations, the relevant features are often unknown *a priori*. Therefore, many candidate features are in-

troduced to better represent the domain. Unfortunately, many of these features are either partially or completely irrelevant or redundant with respect to the target concept. Furthermore, many current applications involve large number of features that can represent tens or hundreds of thousands of input variables.

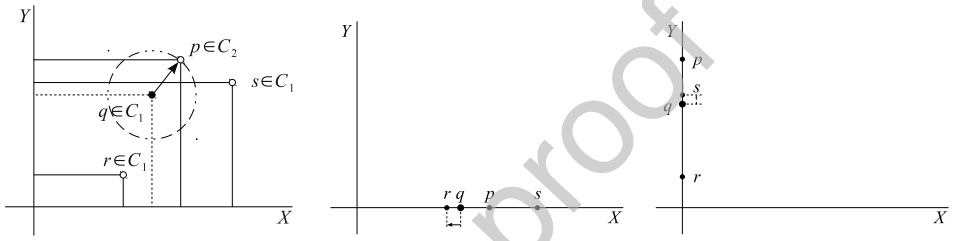
Data mining, as a multidisciplinary joint effort involving databases, machine learning, and statistics, is currently being used to turn mountains of data into nuggets. To use data mining tools effectively, data preprocessing is essential. Feature selection is one of the most important and frequently used techniques in data preprocessing for data mining. In contrast to other dimensionality reduction techniques, feature selection preserves the original semantics of the variables and thus offers the advantage of interpretability by a domain expert[1].

Feature selection can be defined as the selection of a subset Φ' of features from a set Φ of features, where $|\Phi'| < |\Phi|$, such that the value of a criterion function is optimized over all subsets of size $|\Phi'|$[2]. The objectives of feature selection are manifold, with the most important ones being as follows[1]:

- To avoid overfitting and improve model performance, i.e., achieve better prediction performance in the case of supervised classification and better cluster detection in the case of clustering.
- To provide faster and more cost-effective models.
- To gain deeper insight into the underlying processes through which data are generated.

Although most proposed methods for instance selection or feature selection address one of these problems but not both, feature and instance selection are closely related. Depending on the subset of instances considered, the relevant features might change. Conversely, different subsets of features might yield different subsets of relevant instances. Figure 1 presents an illustration of this idea. The instances or features selected to classify query instance q depend on each other, as different instances will be nearest to the query instance in different subspaces. Thus, jointly searching for a subset of relevant instances

and features may be beneficial for the overall classification performance of the obtained subset. It has also been established experimentally[3] that the simultaneous selection of features and instances yields better results than sequential selection. Furthermore, because such algorithms can remove instances as well as features, their data reduction capability is increased. The problem of feature and instance selection is further complicated in the case of many features but few instances[4].



(a) 1-NN rule using inputs x and y.

(b) 1-NN rule using input x.

(c) 1-NN rule using input y.

Figure 1: Relationship between instance and feature selection. We consider a test instance, q, and three training instances, p, s and r, belonging to classes 2, 1 and 1, respectively. Using the 1-Nearest neighbor rule, to classify q correctly, we need to select different instances depending on the features selected. If we select feature x, we need to select instance r, and if we select feature y, we need to select instance s. If both features are selected, then q is misclassified.

Most previous attempts to deal with the simultaneous selection of features and instances have been based on evolutionary algorithms[5, 6, 4] or similar swarm-based procedures[7]. Although these methods achieve very good results on small to medium datasets, it is very difficult to apply them to large datasets. Methods used for scaling up instance selection algorithms, such as stratification sampling for evolutionary instance selection[8], are not readily applicable to simultaneous instance and feature selection. Other common ways of speeding up evolutionary algorithms such as using a cache of distances cannot be used for this task as the number of possible subspaces grows exponentially. This means that, even when several methods of scaling up are used the time needed for large

datasets is prohibitive. As an example, García-Pedrajas *et al.*[5] developed a memetic algorithm for simultaneous instance and feature selection. Although several modifications were carried out to make it scalable, in the worst case the algorithm took more than 100 hours to evolve a solution for a dataset with 5,620 instances and 64 features. It is evident that such method would be infeasible for the datasets used in this paper.

In this paper, we aim to design a method for the simultaneous selection of instances and features that can be scaled up to problems with millions of features and instances. We place special emphasis on problems with many features, as the presence of many features poses a greater challenge for existing methods. To meet the requirement of scalability, we resort to two well-known techniques for scaling up learning algorithms. First, we use simple search heuristics, which are better suited to large datasets. Second, we limit the search space with certain simplifications that reduce the number of possible solutions.

Our proposal is called **S**imultaneous **I**nstance and **F**eature **S**election using sequential **F**orward **S**earch (SI(FS)^2). The algorithm searches for the optimal combined subset of instances and features using a simple heuristic search, namely, sequential forward selection (SFS), in the feature space together with an incremental instance selection process. Our method is intended for application to datasets with many features[4].

This paper is organized as follows: Section 2 reviews some related work; Section 3 describes our proposal; Section 4 describes the experimental setup; Section 5 presents the results of our experiments; and Section 6 summarizes the conclusions of our work.

2. Related work

A major problem with simultaneous instance and feature selection is the absence of a formal theory that can be readily applied, as is the case for either feature or instance selection separately. For feature selection, methods based on mutual information or various statistical tests have been developed[9]. However,

it is very difficult to develop similar methods for selecting instances and features at the same time. Thus, most of the methods developed so far for simultaneous instance and feature selection are based on heuristics such as evolutionary algorithms[6], memetic algorithms[5] or particle swarm optimization[7]. Multi-objective optimization has also been used. Intelligent Multiobjective Evolutionary Algorithm (IMOEA)[10] is a multi-objective evolutionary algorithm, which considers both instance and feature selection. The algorithm has two objectives, maximization of training accuracy and minimization of the number of instances and features selected. The multi-objective algorithm used is based on Pareto dominance because the approach is common in multi-objective algorithms. As in other evolutionary methods scalability is a serious issue.

The first simultaneous method, RMHC-PF1, for instance and feature selection was proposed by Skalak[11]. The method consists of a simple random mutation hill climbing search where instances and features are equally considered during the search. Dasarathy and Sánchez[12] developed a method based on combining both tasks using an heuristic search. However, this method has scalability problems and can also remove entire classes[13]. Babu *et al.*[14] developed another method but it was only usable for binary features.

Zhang et al.[15] proposed a unified criterion for feature and instance selection (UFI) to simultaneously identify the most informative features and instances that minimize the trace of the parameter covariance matrix. A greedy algorithm was introduced to efficiently solve the optimization problem. Experimental results were presented for two small datasets.

Tsai *et al.*[16] combined instance and feature selection by means of a genetic algorithm for large datasets. However, the two processes were carried out sequentially rather than simultaneously. The order of selection was decided depending on the characteristics of the datasets.

Derrac *et al.*[17] developed a cooperative coevolutionary approach based on the use of three populations: the first population performed an instance selection process, the second population performed a feature selection process, and the third population performed a joint instance and feature selection process.

The method was advantageous compared with other evolutionary approaches performing instance and feature selection either separately or jointly.

Villuendas-Rey *et al.*[13] used genetic algorithms and rough sets for instance and feature selection to improve the classification of families having children with affective-behavioral maladies.

An objective similar to simultaneous instance and feature selection was addressed by Suganthi and Karunakaran[18], who applied a feature selection process using cuttlefish optimization followed by principal component analysis. In the same way, Zhang *et al.*[19] proposed an algorithm based on fuzzy rough sets to select representative instances and then perform feature selection.

However, one major problem with these previous approaches is their scalability. These previous papers addressed small- to medium-sized problems. The number of features was always low. Table 1 shows the maximum numbers of instances and features addressed in those previous papers. In this paper, we consider a large dataset with more than 2 million instances and 3 million features, representing a clear distinction from previous works, which considered much smaller problems. Problems with millions of instances are quite common in many research fields such as Bioinformatics, Physycs, Chemistry and many other scientific areas. Datasets with thousands and even millions of features are common in data extracted from web pages or when bag of words or n-grams are used to represent text documents[20].

Table 1: Maximum numbers of features and instances in previous works.

Paper	Largest datasets for instances		Largest datasets for features	
	Instances	Features	Instances	Features
Proposal	**2,396,130**	**3,231,961**	**2,396,130**	**3,231,961**
Derrac *et al.* (2010) [17]	6,435	36	360	90
Ahmad and Pedrycz (2011) [7]	5,875	26	5,875	26
Zhang *et al.* (2012) [15]	1,500	241	400	1,024
Tsai et al. (2013) [16]	581,012	54	779	2,197
Villuendas-Rey *et al.* (2013) [21]	1,728	6	798	38
García-Pedrajas *et al.* (2014) [5]	6,435	36	279	452
Zhang *et al.* (2018) [19]	7,195	21	126	310
Suganthi and Karunakaran (2019) [18]	1,025,010	11	581,012	54

3. SI(FS)^2: Fast simultaneous instance and feature selection for datasets with many features

As stated in the previous section, our aim is the development of a new method for the simultaneous selection of instances and features, especially one that is scalable to datasets with a large number of features. This aim precludes the use of previous approaches based on evolutionary algorithms or similar metaheuristics due to their difficulties in dealing with large datasets for this task[1].

Thus, our first decision is that our method must be based on simple search heuristics, one of the most straightforward ways of scaling up learning methods[22]. The second decision is related to the size of the search space. Searching for the global optimum in both the feature and instance spaces is infeasible. Thus, we opt for two interwoven steps that aim to optimize both selections but in a feasible way. Based on these two premises, we design, as the basic architecture of our proposal, a SFS search process in the feature space in which every time a new feature is considered to be added to the subset of selected features, an instance selection algorithm is carried out in the corresponding subspace. To describe the method more formally, consider a training set $S = \{(\mathbf{x}_1, y_1), \ldots, (\mathbf{x}_N, y_N)\}$ with a feature set $\Phi = \{\phi_1, \phi_2, \ldots, \phi_M\}$. Our task is to obtain a subset of instances, $S' \subset S$, and features, $\Phi' \subset \Phi$, such that $S' \ll S$ and $\Phi' \ll \Phi$. Based on the premises stated above, we develop the first basic model shown in Algorithm 1. The result of the algorithm is the subset of selected features, Φ', and the subset of selected instances, S'.

Thus, our method is based on combining a simple heuristic search in the feature space with an instance selection algorithm in the instance space. However, both algorithms would be too computationally expensive if we were to consider the whole search space in every step. In a search space consisting of tens of thousands of features and hundreds of thousands of instances, the procedure

[1] We must emphasize that we are not stating that evolutionary algorithms cannot deal with large problems in general but rather that for the task of simultaneous instance and feature selection in particular, their scalability is problematic.

Algorithm 1: Basic skeleton for our proposed method of simultaneous instance and feature selection.

Data: Training set $S = \{(\mathbf{x}_1, y_1), \ldots, (\mathbf{x}_n, y_n)\}$ with feature set $\Phi = \{\phi_1, \phi_2, \ldots, \phi_M\}$ and instance selection algorithm IS.

Result: Subset of selected features $\Phi' \subset \Phi$ and instances $S' \subset S$.

1 $\Phi' = \emptyset$

2 **do**

  /* Forward step */

3 **forall the** $\phi \in (\Phi - \Phi')$ **do**

4 $F = \Phi' \cup \phi$

5 Perform instance selection on S with subspace F: $s' = \text{IS}(S, F)$

  /* IS(S, F) perform IS algorithm on S using subspace F. */

6 Evaluate 1-NN classification performance using (S', F'); if current best, $\phi_{\text{best}} = \phi$ and $S'_{\text{best}} = s'$

7 **end**

  /* ϕ_{best} and S'_{best} are the best feature and best subset of instances obtained in the preceding loop respectively */

8 $\Phi' = \Phi' + \phi_{\text{best}}$

9 $S' = S'_{\text{best}}$

10 **while** *Convergence criterion not met*

11 Return S', Φ'

described in Algorithm 1 could not be executed efficiently.

Therefore, several modifications are introduced to obtain a fast and scalable method. A first approach to reducing the time needed for every round of the algorithm is to consider only a subset of features for addition in every step of the procedure. Furthermore, when the number of features is very large, many of these features will be useless. To avoid repeatedly considering useless features for addition, we use a roulette selection scheme in which the probability of selecting each feature depends on its relevancy. Many ranking methods for scoring the importance of a feature have been developed. Among them, methods based on mutual information both offer good performance and are fast[23] and thus the reason why we use them. However, the particular feature ranking method used does not have a significant impact on the performance of SI(FS)^2, provided that it is fast enough to avoid compromising the scalability of our method. Thus, for every feature ϕ_i, we obtain the mutual information between that feature and the target class and assign that score to a vector of weights to be used for the roulette selection of features: $\mathbf{w}_i = \text{MI}(\phi_i, Y)$. Then, the number of features considered for addition each round is set to a fixed small number, this value is named f_{step} (see Algorithm 2 line 6). Through experiments, we will show that

considering only 100 features is sufficient for datasets with millions of features.

The second way to scale up our method is central to our algorithm. It consists of a constructive scheme for selecting instances. Instance selection can be a costly procedure[24]. This procedure must be carried out many times in our method, thus precluding its application to even moderately large datasets. To circumvent this problem, we modify the way the instances are selected using the following procedure:

- Initially, all instances are removed from the selected subset S'.
- Each time the instance selection algorithm is carried out for a certain subspace, it is performed on a small stratified sample from the whole dataset, $s \subset S$, to obtain a subset of selected instances, $s' \subset s$ (see Algorithm 2 step 5). As will be shown in the experimental results, the size of this subset can be as small as 100 instances. The evaluation of the current subspace and the subset of selected instances is also carried out considering only the instances in s. There is not a fixed bound for the size of s, as it depends on the available resources and the time constraints. However, as a rule of thumb, in the reported experiments a size of 1,000 instances was used with very good results. That means that although larger samples could be used for datasets with many instances a good performance is already achieved with this small sampling size.
- Once the best feature to add to the subset of selected features, ϕ_{best}, is obtained, the selection associated with it, s_{best}, is used to update the current subset of instances S' as follows: $S' = S' + \{\mathbf{x} \in s'\}$. That is, all selected instances in s' are added to the selected subset S'. If an instance was already in S', its status is not affected.
- Regarding the unselected instances in s, i.e., $s - s'$, two versions of the algorithm have been implemented. In the first version, after the selected instances are added, the instances in s that have not been selected are also removed, i.e., $S' = S' - \{\mathbf{x} \in (s - s')\}$. This is the *standard* version of

our proposed SI(FS)^2 method. In the alternative version, called the *OR* version or our proposed SI(FS)$^2\vee$ method, the unselected instances do not affect the set S'; only the newly selected instances are added to S', i.e., $S' = S' \cup s'$.

The last method used to improve the speed of SI(FS)^2 is applied in the evaluation of the current solution, which must be carried out in every round of the algorithm. For large datasets, evaluating the 1-NN classification performance may be a costly process. In our method, we instead evaluate the classification performance using a stratified partition of the dataset. The training data are divided into T strata of fixed size, which was 10,000 instances in the experiments reported here, using a stratified partition that preserves the class distribution of the instances, and then, the 1-NN classification performance is evaluated in each stratum independently. The overall classification performance value is the average over all the strata. As will be shown in the reported experiments, with this procedure, it is possible to obtain very small subsets of features and instances that provide a performance comparable to that achieved using the whole dataset in a moderate amount of time.

To evaluate the current solution, we consider both the classification performance and the data reduction ability. Both factors are weighted equally. The quality of a certain selection of instances, S', and features, Φ', is given by

$$f(S', \Phi') = \text{classification performance}_{1-NN}(S', \Phi') + r(S', \Phi'), \qquad (1)$$

where $\text{classification performance}_{1-NN}(S', \Phi')$ is the classification performance of the subset of instances and features obtained using the stratified approach described above and $r(S', \Phi')$ is the reduction of the dataset, defined as $r = 1 - \frac{|S'|}{|S|}\frac{|\Phi'|}{|\Phi|}$. For the classification performance we can consider any metric of our choice.

The stopping criterion is based on the average value of this fitness measure throughout the execution of the algorithm. A certain number of consecutive rounds with worsening average values must be reached to consider that no fur-

ther improvement might be expected. We use the average value instead of the current fitness because it is a more stable measure. The result of our algorithm is always the subset of features and instances that achieved the best classification performance during the search process, as better classification performance is usually preferred over greater data reduction. The length of this "losing streak", l_s, must be sufficiently long to avoid premature convergence, but since the final result is always the best-performing selection, it has no major impact on the ultimate performance of SI(FS)^2. In our experiments, this criterion was fixed to $l_s = 10$. The final SI(FS)^2 procedure is described in Algorithm 2. As in the previous algorithm, the result is the subset of selected features, Φ', and the subset of selected instances, S'.

Algorithm 2: **S**imultaneous **I**nstance and **F**eature **S**election using sequential **F**orward **S**earch (SI(FS)^2).

Data: Training set $S = \{(\mathbf{x}_1, y_1), \ldots, (\mathbf{x}_n, y_n)\}$ with feature set $\Phi = \{\phi_1, \phi_2, \ldots, \phi_M\}$.
Result: Subset of selected features $\Phi' \subset \Phi$ and instances $S' \subset S$.

$\mathbf{w}_i = \mathrm{MI}(\Phi_i, \mathbf{y}), \forall \phi_i \in \Phi$
$\Phi' = \emptyset$
$S' = \emptyset$
do
  /* Forward step */
  s, a random stratified sample of S
  for $i = 1$ *to* f_{step} **do**
    ϕ = Roulette selection of next instance
    $F = \Phi' \cup \phi$
    Perform instance selection on s with subspace F: $s' = \mathrm{IS}(s, F)$
      /* IS(S, F) perform IS algorithm on S using subspace F. */
    Evaluate 1-NN classification performance using (s', F'); if current best, $\phi_{\text{best}} = \phi$ and $S'_{\text{best}} = s'$
  end
  /* ϕ_{best} and S'_{best} are the best feature and best subset of instances obtained in the preceding loop respectively */
  $\Phi' = \Phi' + \phi_{\text{best}}$
  $S' = S' \cup S'_{\text{best}}$
  $S' = S' - s \setminus S_{\text{best}}$
    /* Not applied to OR version of SI(FS)^2 */
while *Convergence criterion not met*
Return S', Φ'

4. Experimental setup

To ensure a fair comparison between standard instance and feature selection algorithms and our proposed approach, we selected a set of 95 datasets with

wide-ranging numbers of patterns, features and classes. A summary of these datasets is provided in Table 1 of the supplementary material. Because this paper focuses on datasets with many features, we selected problems with at least 500 features. To estimate the data reduction and classifier performance, we used 10-fold cross-validation. `rcv1` has two versions, consisting of the original dataset and a stratified 10% sample. The numbers of features of these two versions are different because the sampling caused some of the features to be constant, and thus, they were removed.

We used the Wilcoxon test[25] as the main statistical test for comparing pairs of algorithms. This test was chosen because it assumes limited commensurability and is safer than parametric tests because it does not assume normal distributions or homogeneity of the variance. Thus, this test can be applied to both error ratios and data storage requirements. Furthermore, empirical results[25] show that this test is stronger than other tests.

In our experiments, we also compared groups of methods. In such cases, it is not advisable to use pairwise statistical tests such as the Wilcoxon test. Instead, we first applied the Iman-Davenport test to ascertain whether there were significant differences among the methods. The Iman-Davenport test is based on the χ_F^2 Friedman test, which compares the average ranks of k algorithms, but the former is more powerful. After applying the Iman-Davenport test, we can use any of the general procedures available for controlling the familywise error in multiple hypothesis testing. One of the simplest methods of this type is Holm's procedure[25], which is the one used in our experiments.

In Holm's procedure, the best-performing algorithm in terms of Friedman's ranks is compared in a stepwise manner against the other methods. The test statistic for comparing the i-th and j-th methods is

$$z = \frac{(R_i - R_j)}{\sqrt{k(k+1)/6N}}. \tag{2}$$

The z value is used to find the corresponding probability from the normal distribution table, which is then compared against an appropriate α. The tests

differ in how the α value is adjusted to compensate for multiple comparisons. The ordered p-values are denoted by $p_1, p_2, \ldots$, such that $p_1 \leq p_2 \leq \ldots \leq p_{k-1}$. Holm's step-down procedure starts with the most significant p-value. If p_1 is less than $\alpha/(k-1)$, then the corresponding hypothesis is rejected, and p_2 with $\alpha/(k-2)$ can be tested. If the second hypothesis is rejected, then the test proceeds to the third, and so on. As soon as a certain null hypothesis cannot be rejected, all remaining null hypotheses are also retained.

When the Iman-Davenport test rejects a null hypothesis, we can also proceed with a post hoc Nemenyi test, which compares groups of methods. The performances of two classifiers are considered significantly different if the corresponding average ranks differ by at least the following critical difference:

$$CD = q_\alpha \sqrt{k(k+1)\frac{6}{N}}, \tag{3}$$

where the critical value q_α is based on the studentized range statistic divided by $\sqrt{2}$, N is the number of datasets, and k is the number of compared methods. As a graphical representation of the Nemenyi test, we use the plots described by Demšar[25]. When comparing the algorithms against one another, we connect each group of algorithms that are not significantly different with a horizontal line. We also show the critical difference above the graph. For all statistical tests, we use a significance level of 0.05.

The source codes, written in C and licensed under the GNU General Public License, for all methods as well as the partitioning of the datasets are freely available from the authors upon request.

4.1. Evaluation measures

The data reduction is measured as the percentage of instances and features removed by an algorithm. Thus, if we have N instances and M features and a certain algorithm selects n instances and m features, the reduction is $r = 1 - (n/N)(m/M)$. We can use the standard measure of accuracy to quantify the percentage of instances that are correctly classified. However, recent works have shown that misclassification rates may be biased because they contain

substantial randomness[26]. Furthermore, when the numbers of patterns are not evenly distributed among the classes, the accuracy may yield a poor evaluation of the performance, as the performance on classes with only a few patterns is almost entirely disregarded if we calculate the accuracy over the entire test set. Thus, as a performance measure, we use Cohen's κ measure, which compensates for random hits. Its original purpose was to measure the degree of agreement. However, κ can also be adapted to measure classification accuracy, and its use is recommended because it accounts for random successes[26]. For a classification task, the value of κ can be computed from the confusion matrix as follows:

$$\kappa = \frac{n \sum_{i=1}^{C} x_{ii} - \sum_{i=1}^{C} x_{i\cdot} x_{\cdot i}}{n^2 - \sum_{i=1}^{C} x_{i\cdot} x_{\cdot i}}, \tag{4}$$

where x_{ii} is the cell count on the main diagonal, n is the number of examples, C is the number of classes, and $x_{i\cdot}$ and $x_{\cdot i}$ are the total column and row counts, respectively. The value of κ ranges from -1 (total disagreement) to 1 (perfect agreement). For multiclass problems, κ is a very useful yet simple metric for measuring the accuracy of a classifier while compensating for random successes.

5. Experimental results

As stated in the previous sections, the actual instance selection algorithm used in our search is a variable setting of our method. In the first step of our experiments, we performed a comparison of 14 different instance selection algorithms to determine the one that is best suited for our method using the datasets listed in Table 1 of the supplementary material and the two versions of our proposed method. The tested instance selection algorithms were BBIS[27], CHC[24], CNN[28], DROP3[29], GCNN[30], HMNEI[31], IB3[28], ICF[32], MSS[33], PSA[34], PSRCG[35], RMHC[28], RNGE[28] and RNN[28]. For the OR version of our method, the name of the instance selection algorithm used is marked as NAME$\vee$.

Figures 2 and 3 show the results of the Nemenyi test for data reduction and classification performance, measured using κ, as obtained with the 14 different

instance selection methods and the two versions of SI(FS)^2. Detailed numerical comparisons are shown in Tables 4, 5, 6 and 7 of the supplementary material. The Iman-Davenport test yielded a p-value of 0.0000 for three compared factors of the algorithms: classification performance, data reduction and run time.

In terms of data reduction ability (see Figure 2), the first interesting result is the impressive ability of our method to achieve large reductions. Even the worst method in terms of the Friedman ranks, SI(FS)$^2\vee$ using RNGE, achieved an average reduction of over 96%. Several algorithms achieved an average reduction of over 99%. Although RMHC was the best overall method, the differences were small. RNGE also showed very good performance in terms of data reduction but at the cost of the worst classification performance. As is clear from the algorithm design, the OR version of SI(FS)^2 reduced fewer instances in all cases than its standard counterpart.

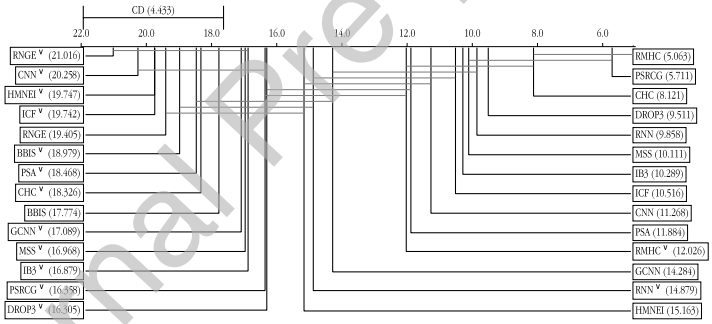


Figure 2: Nemenyi test for reduction results for the 14 different instance selection algorithms used to implement SI(FS)^2.

In terms of κ, CHC$\vee$ was better than the remaining methods, although the Nemenyi test did not find any significant differences among the best 22 methods. Overall, there were four algorithms that achieved good classification performance for both versions of the proposed method: CHC, RMHC, HMNEI and BBIS. Among these algorithms, we selected HMNEI as the representative setting of SI(FS)^2 for the remaining experiments due to its very good combined behavior in terms of classification performance and data reduction. Thus, HM-

NEI was used as the representative instance selection method of our approach for the subsequent experiments.

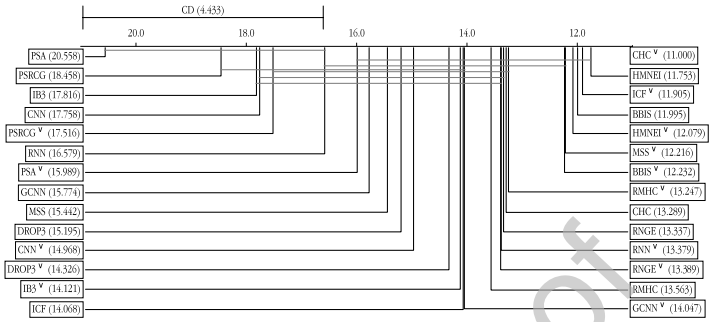


Figure 3: Nemenyi test for κ results for the 14 different instance selection algorithms used to implement SI(FS)^2.

Finally, a run-time comparison is shown in Figure 1 of the supplementary material. In most cases, the OR version ran for a shorter time. This is explained by its worse data reduction ability. Because our algorithm stops when the average fitness has continuously decreased for a certain number of rounds, a worse reduction ability means that the classification performance must continuously improve to avoid convergence. Thus, the OR version usually met the stopping criterion earlier. However, there is room for differences, as its better classification performance compensated for this effect on many datasets.

Previous results have shown that evolutionary algorithms outperform other methods[17]. The differences between the different evolutionary methods are usually small and due to the large computational cost involved comparing with many different algorithms is infeasible. Thus, we compared SI(FS)^2 against a memetic algorithm that showed the best performance compared with other evolutionary methods in a comparison on many datasets[5]. We also compared both algorithms against the classification performance of the 1-NN rule without instance or feature selection.

Due to the size of some of the datasets and to ensure that comparisons were carried out on similar grounds for all methods, we set an arbitrary time

limit of 10 hours for all algorithms. When this time was reached, the best result so far was returned as the final output of the algorithm. In the tables, some of the methods are reported to have longer run times because the last iteration/generation of every algorithm was always allowed to reach completion. Table 2 of the supplementary material shows the κ results for 1-NN, a memetic algorithm performing instance and feature selection, and the two versions of our method with HMNEI as the instance selection algorithm.

The four methods are compared in Table 2. For κ, the Iman-Davenport test yielded a p-value of 0.0109. The Wilcoxon test showed that both versions of SI(FS)^2 achieved significantly better performance not only than the memetic algorithm but also than the use of the whole dataset while achieving dramatic data reduction. For κ, no significant differences were found between the two versions of our method.

Table 2: Comparison of results for 1-NN, a memetic algorithm and our method in terms of the κ measure.

		1-NN	Memetic	SI(FS)^2	SI(FS)$^{2\vee}$
Mean		0.5318	0.5246	0.5678	0.5663
Ranks		2.7316	2.7053	2.3158	2.2474
1-NN	w/l		47/45	50/36	53/25
	p		0.7921	0.0284	0.0027
	R^+/R^-		2351.0/2209.0	2870.5/1689.5	3088.5/1471.5
Memetic	w/l			57/35	55/36
	p			0.0325	0.0227
	R^+/R^-			2856.0/1704.0	2894.0/1666.0
SI(FS)^2	w/l				42/41
	p				0.8035
	R^+/R^-				2213.0/2347.0

The results for data reduction are shown in Table 3 of the supplementary material, and the corresponding comparison is shown in Table 3. For data reduction, the Iman-Davenport test yielded a p-value of 0.0000. The differences between our proposed method and the memetic algorithm are dramatic. Both versions of our method achieved greater reduction at the 99% confidence level. As should be expected, the standard version of SI(FS)^2 outperformed the OR version. Both tables show the efficiency of the memetic algorithm and SI(FS)^2 in terms of data reduction. The memetic algorithm achieved an average reduction greater than 90%.

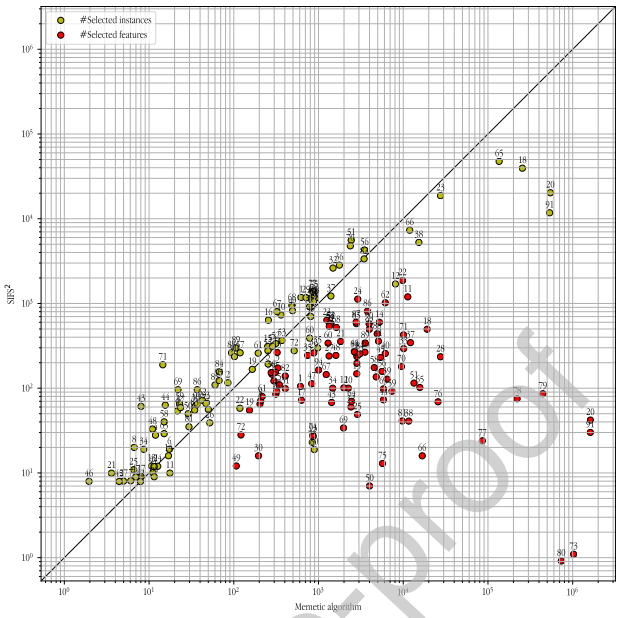
Table 3: Comparison of the results for the memetic algorithm and our method in terms of data reduction.

		Memetic	$SI(FS)^2$	$SI(FS)^{2\vee}$
Mean		0.9050	0.9801	0.9594
Ranks		2.8632	1.3053	1.8316
Memetic	w/l		93/2	84/11
	p		0.0000	0.0000
	R^+/R^-		4557.0/3.0	4159.0/401.0
$SI(FS)^2$	w/l			25/66
	p			0.0000
	R^+/R^-			921.5/3638.5

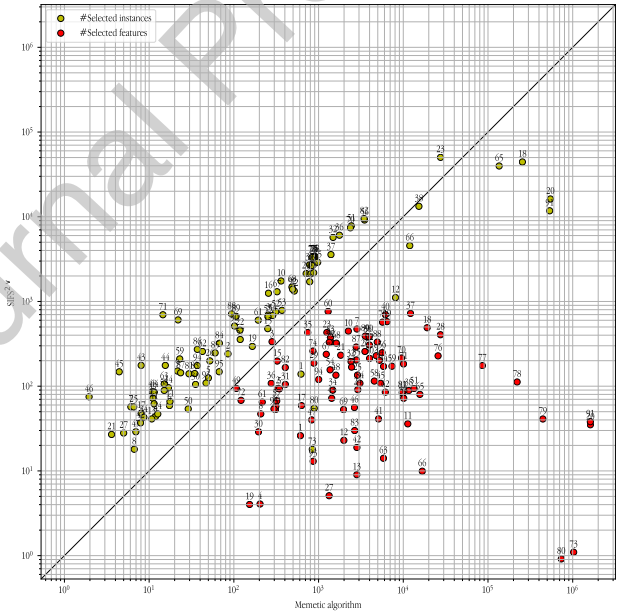
However, the reduction ability was mainly focused on instance selection; only approximately 50% of the features were removed, whereas the removed instances constituted more than 90%. On the other hand, the data reduction ability of $SI(FS)^2$ showed impressive results, with average reductions of 98.0% and 95.9% for the standard and OR versions of our algorithm, respectively. On average, only 43.8 instances and 95.5 features were selected. A comparison of the memetic algorithm and $SI(FS)^2$ in terms of the numbers of instances and features selected is shown in Figure 4.

To better illustrate the results, Figure 4 shows the instances and features selected by each method for all datasets. The actual results are shown in Table 8 and 9 of the supplementary material for the instances and features, respectively. In general, $SI(FS)^2$ retained more instances, although the differences were small. However, its ability to remove features was far superior to that of the memetic algorithm. As an example, for the datasets with the largest number of features, `day` and `url-rep`, both with 2,396,130 features, our proposed method retained only 16,211.9 and 11,763.9 features on average, respectively, while 534,496.9 and 545,199.9 features, respectively, were retained on average by the memetic algorithm. Furthermore, in both cases, the classification performance of $SI(FS)^2$ was better than that of the memetic algorithm.

In fact, for datasets with more than 10,000 instances, the evolutionary algorithm performed too few iterations in the assigned time, which resulted in very poor performance. The capability of faster execution is an interesting characteristic of our approach when dealing with large datasets. To further illustrate



(a) Memetic algorithm and SI(FS)^2



(b) Memetic algorithm and SI(FS)$^2\vee$

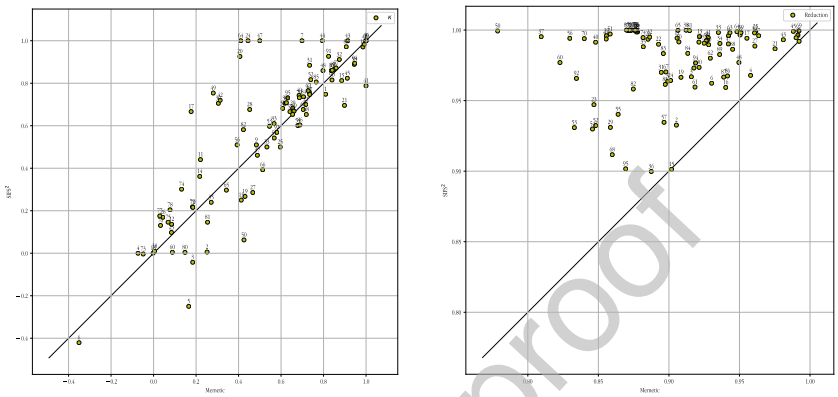
Figure 4: Comparison of the numbers of instances and features selected by the memetic algorithm (on the x-axis) and SI(FS)^2 with HMNEI (on the y-axis). Both axes are shown on a logarithmic scale.

the results, Figure 5 shows the results for the memetic algorithm and SI(FS)^2 in terms of κ, data reduction, run time and the average between κ and reduction as a combined fitness metric. We selected the standard version of our method for this comparison because it matched the classification performance of the OR version with significantly better data reduction.

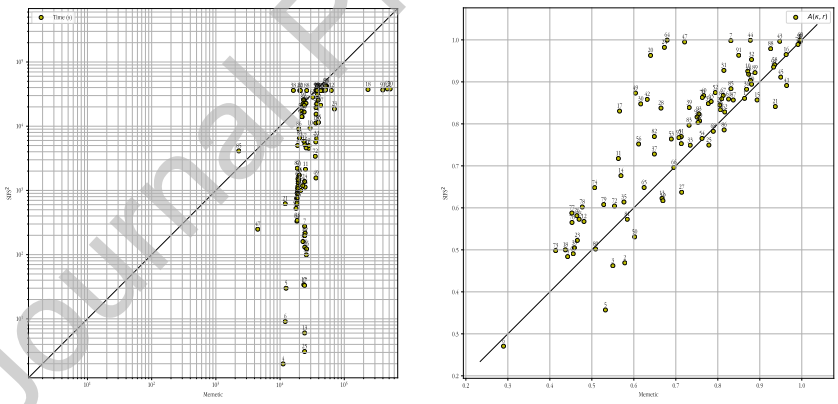
Figure 5(a) shows κ, Figure 5(b) shows the data reduction, Figure 5(c) shows the run time, and Figure 5(d) shows the average of κ and data reduction. In these figures, points above the main diagonal line $y = x$ indicate better performance of SI(FS)^2. The results for the κ measure show the overall advantage of our method, as corroborated by the Wilcoxon test. More points are located above the diagonal, and these points tend to be more separated from the $y = x$ line. The differences in data reduction ability shown in Figure 5(b) are a perfect illustration of the capability of our method. With the exception of the `cnae-9` dataset, for which our method was 0.1% worse on average, for the remaining 94 datasets, SI(FS)^2 achieved a superior reduction ability.

Regarding the run time (see Figure 5(c)), we found a group of datasets for which both algorithms did not finish within the time limit of 10 hours, and thus, their run times were similar. The differences depend on the length of the last iteration of each method. However, there are a few datasets, namely, `comedy_comparisons`, `day`, `rcv1` and `url-rep`, for which just one generation of the memetic algorithm took many hours and the overall run time was much longer than that of our proposed method. Finally, to unify the performance of the two methods into one metric, we show the average of κ and the data reduction metric in Figure 5(d). This last figure shows a clear improvement of SI(FS)^2 over the memetic algorithm. Most of the points are above the main diagonal and, furthermore, are clearly more separated from it than the points below the diagonal.

The results of the two selection methods are further illustrated in Figure 6. This figure shows the results in terms of κ and data reduction. This graphical representation is based on κ-error relative movement diagrams. In such diagrams, an arrow is used to represent the results of both methods applied



(a) κ (b) Data reduction



(c) Time (logarithmic scale) (c) Average of κ and data reduction

Figure 5: Comparison in terms of classification performance, data reduction and run time for the memetic algorithm and SI(FS)^2 using HMNEI.

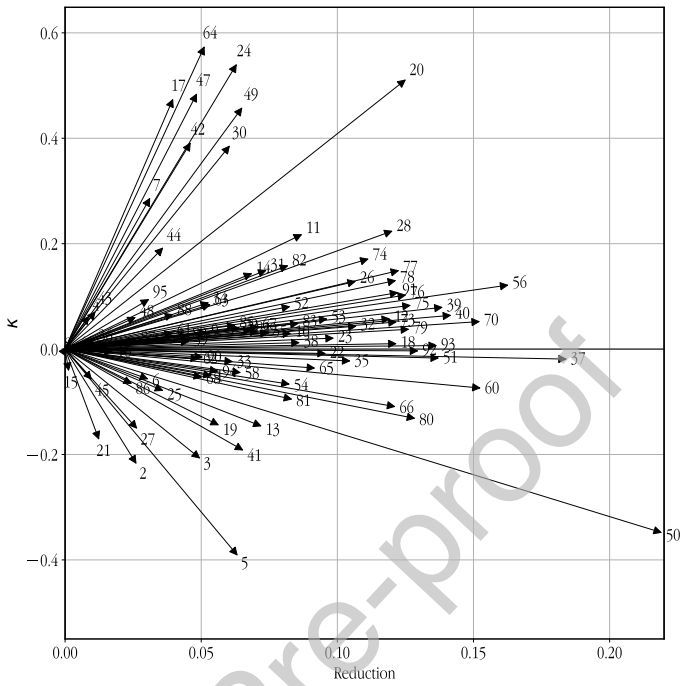


Figure 6: Relative movement diagram for CHC and SI(FS)^2 in terms of κ and data reduction.

to the same dataset. The arrow starts at the coordinate origin, and the coordinates of the tip of the arrow represent the differences between the κ and data reduction metrics of our boosting method and those of the standard algorithm alone. These graphs are a convenient way of summarizing the results. A positive value for either data reduction or κ means that our method performed better. This figure shows that SI(FS)^2 achieves impressive performance in terms of data reduction while also improving the overall classification performance of the selected subset of instances and features.

As stated above, the major aim of this work was to develop a simultaneous instance and feature selection method that is able to deal with large numbers of features. To evaluate whether that objective was achieved, Table 4 shows a summary of the results for the nine datasets with more than 50,000 features. This table shows interesting results. First, SI(FS)^2 achieved an impressive data

reduction of more than 99.9% on average, in contrast with the 87.5% reduction of the memetic algorithm. In terms of the number of features selected, only 62.2 features were selected on average. For some of the largest datasets, this reduction was achieved while maintaining a classification performance close to that achieved with the use of the whole dataset. For the `day` dataset, only 42.1 features were selected from a total of 3,231,961. However, the value of κ was 0.9260, very close to the value of 0.9411 obtained using the whole dataset. The same behavior was observed for `url-rep` and `sens-10-grams`.

Table 4: Results of the memetic algorithm and our method for datasets with more than 50,000 features.

Dataset	Features	1-NN	Memetic				
		κ	Red.	#Instances	#Features	κ	Time(s)
day	3,231,961	0.9411	0.8735	545199.9	1616872.1	0.4062	518134.3
farm-ads	54,877	0.6547	0.8769	916.0	27499.1	0.4525	37850.4
sens-10-gram	2,052,063	-0.0499	0.8766	843.9	1025693.1	-0.0489	53219.6
sens-5-gram	51,238	0.0374	0.8728	866.0	25746.1	0.0327	39258.4
sens-6-gram	172,109	0.1059	0.8758	851.9	85860.9	0.0292	42982.7
sens-7-gram	441,251	0.0209	0.8769	842.0	220778.1	0.0775	49212.1
sens-8-gram	888,837	0.0590	0.8721	874.9	444590.0	0.1848	51234.2
sens-9-gram	1,465,215	0.0258	0.8699	888.9	733666.1	0.1479	51234.3
url-rep	3,231,961	0.9427	0.8761	534496.9	1616021.9	0.8236	398156.4
Average		0.3042	0.8745	120642.1	644080.6	0.2340	137919.5

Dataset	Features	1-NN	$SI(FS)^2$				
		κ	Red.	#Instances	#Features	κ	Time(s)
day	3,231,961	0.9411	1.0000	20263.9	42.1	0.9260	38092.0
farm-ads	54,877	0.6547	0.9987	1148.0	235.1	0.6764	36017.3
sens-10-gram	2,052,063	-0.0499	1.0000	22.9	1.1	-0.0037	41931.5
sens-5-gram	51,238	0.0374	0.9996	1034.0	69.1	0.1308	25491.1
sens-6-gram	172,109	0.1059	0.9999	1383.9	23.9	0.1754	21218.2
sens-7-gram	441,251	0.0209	0.9999	1075.0	75.0	0.2043	36170.4
sens-8-gram	888,837	0.0590	1.0000	1045.9	85.9	0.2157	36679.5
sens-9-gram	1,465,215	0.0258	1.0000	18.9	1.8	0.0040	42574.5
url-rep	3,231,961	0.9427	1.0000	11756.9	30.0	0.9267	36714.6
Average		0.3042	0.9998	4194.3	62.6	0.3617	34987.3

Regarding the comparison with the memetic algorithm, $SI(FS)^2$ showed the ability to deal with large datasets, a situation in which the memetic algorithm encountered serious problems in achieving a useful solution. The scalability problems of the latter are clearly shown: although a significantly longer time was used, the achieved solution was inferior to that of $SI(FS)^2$ in terms of both data reduction and κ. In terms of both measures, our approach obtained results that were improved by more than 10% compared with those of the memetic

algorithm.

Among the non-evolutionary methods previously proposed we also considered the method of Dasarathy and Sánchez[12]. This method consists of a sequential backward selection search in tge feature space where at each step two instance selection algorithms are applied one after another, RNGE and MCS[36] procedures. This method is not scalable even to medium datasets, due to the large computational cost of each step and thus cannot be compared with our proposal. Skalak[11] proposed a random mutation hill climbing method to select prototypes and features (RMHC-PF1) that it is able to deal with large datasets. Table 5 shows the comparison of SI(FS)^2 and RMHC-PF1 in terms of reduction and classification performance.

Table 5: Comparison of the results of RMHC-PF1 and our method in terms of κ and data reduction.

		Reduction	
		RMHC-PF1	SI(FS)^2
Mean		0.9650	0.9801
Ranks		1.6421	1.3579
RMHC-PF1	w/l		61/34
	p		0.0007
	R^+/R^-		3189.0/1371.0

		κ	
		RMHC-PF1	SI(FS)^2
Mean		0.4749	0.5678
Ranks		1.7316	1.2684
RMHC-PF1	w/l		65/21
	p		0.0000
	R^+/R^-		3604.5/955.5

The table shows the advantage of our method when compared with RMHC-PF1. SI(FS)^2 significantly outperformed RMHC-PF1 in terms of both reduction and classification performance measured using κ.

Instance and feature selection when performed together are usually applied to nearest neighbor classifier, however the selected subset of instances and features can also be applied to other classification models. To further assure the viability of our proposal we also evaluated its results using three additional classifiers: decision trees (DTs), support vector machines (SVMs) and random forests (RFs). We have selected these methods because they are some of the

most commonly used and are typically top performing when classification models are compared. To set the hyperparameters of the SVMs, we utilized a linear kernel with $C \in \{0.1, 1, 10\}$, and a Gaussian kernel with $C \in \{0.1, 1, 10\}$ and $\gamma \in \{0.0001, 0.001, 0.01, 0.1, 1, 10\}$, and tested all 21 possible combinations using 10-fold cross-validation on the training set. Although this method does not ensure an optimal set of hyperparameters, it guarantees that a satisfactory set of hyperparameters will be obtained in a reasonable amount of time. For DTs, we used Gini as the impurity measure. For RFs, we used forests of 100 member trees.

Tables 6, 7 and 8 show the comparison of RMHC-PF1, the memetic algorithm and our proposal, SI(FS)^2 version, using as classification model DTs, RFs and SVMs respectively in terms of κ measure. Reduction results were already reported in Table 3. A first interesting result is the best overall performance of RFs and SVMs against DTs. The average κ values of RFs and SVMs are clearly above that of the DTs. Regarding the behavior of SI(FS)^2 it kept its better performance using the three different classifiers. SI(FS)^2 beat both the memetic algorithm and RMHC-PF1 using either DTs, RFs or SVMs.

Table 6: Comparison of the results of RMHC-PF1, a memetic algorithm and our method in terms of κ using a decision tree as classifier.

		Memetic	RMHC-PF1	SI(FS)^2
Mean		0.2121	0.2636	0.3380
Ranks		2.2526	2.0842	1.6632
Memetic	w/l		42/33	61/22
	p		0.1568	0.0000
	R^+/R^-		2661.0/1899.0	3407.0/1153.0
RMHC-PF1	w/l			56/31
	p			0.0120
	R^+/R^-			2957.0/1603.0

The results of the previous tables are illustrated in Figures 7(a) and 7(b) for the memetic algorithm and RMHC-PF1 respectively. The figures show the performance of our method and the corresponding standard approach for the three classifiers. Points above the main diagonal are datasets for which SI(FS)^2 performed better than the standard approach. Both plots show most of the points above of the main diagonal. Furthermore, there is a large separation

Table 7: Comparison of the results of RMHC-PF1, a memetic algorithm and our method in terms of κ using a random forest as classifier.

		Memetic	RMHC-PF1	$SI(FS)^2$
Mean		0.5250	0.5001	0.5734
Ranks		2.1053	2.0947	1.8000
Memetic	w/l		42/42	48/28
	p		0.5801	0.0171
	R^+/R^-		2131.0/2429.0	2922.0/1638.0
RMHC-PF1	w/l			53/35
	p			0.0207
	R^+/R^-			2903.0/1657.0

Table 8: Comparison of the results of RMHC-PF1, a memetic algorithm and our method in terms of κ using a support vector machine as classifier.

		Memetic	RMHC-PF1	$SI(FS)^2$
Mean		0.5375	0.5039	0.6100
Ranks		1.8789	2.3947	1.7263
Memetic	w/l		21/54	42/32
	p		0.0007	0.0466
	R^+/R^-		1370.5/3189.5	2815.5/1744.5
RMHC-PF1	w/l			60/18
	p			0.0000
	R^+/R^-			3643.5/916.5

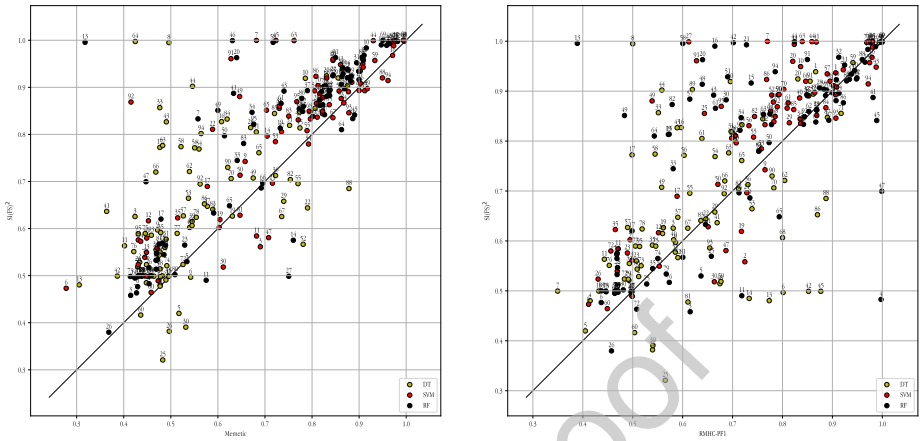
from the main diagonal for many datasets meaning a considerable classification performance.

5.1. Comparison with instance and feature selection separately

One interesting question that must be experimentally tested is whether simultaneous instance and feature selection is able to outperform instance and feature selection separately. Thus, we performed an additional experiment using memetic algorithms to separately select instances and features. Tables 10 and 11 of the supplementary material show the results for instance and feature selection performed separately respectively.

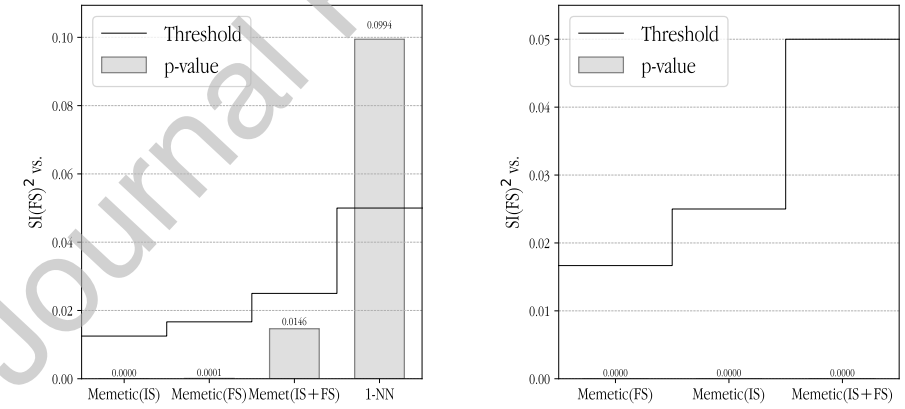
Figure 8 shows a comparison of all four methods using Holm's procedure. The Iman-Davenport test yielded a p-value of 0.0000 for both data reduction and κ.

This figure compares the best method in terms of Friedman's ranks, i.e., $SI(FS)^2$, with the other three methods. First, the figure shows that separate instance and feature selection performed worse than simultaneous instance and



(a) $SI(FS)^2$ vs. memetic algorithm (b) $SI(FS)^2$ vs. RMHC-PF1

Figure 7: Comparison of the classification performance of the memetic algorithm (a) and RMHC-pF1 (b) against $SI(FS)^2$ in terms of κ for every dataset and the three different classifiers.



(a) κ (b) Data reduction

Figure 8: Comparison using the Holm procedure in terms of classification performance and data reduction among memetic algorithms for instance selection, feature selection and simultaneous instance and feature selection and $SI(FS)^2$ with HMNEI. For classification performance, 1-NN with all instances and features is also considered.

feature selection in terms of both data reduction and classification performance. In terms of κ, SI(FS)^2 outperformed the other three selection methods and was even able to achieve a better rank than 1-NN, although the difference was not significant. In terms of data reduction, the dramatic ability of SI(FS)^2 to remove instances and features is shown by the fact that it outperformed all other selection algorithms.

5.2. Study of the hyperparameters

One of the advantages of our proposed method is that is has very few hyperparameters that must be tuned in order to apply it to a problem. Above, we have shown how different instance selection algorithms can be chosen while maintaining very good performance. The only two remaining hyperparameters are the sizes of the subsets of features and instances used for each round of SI(FS)^2. We fixed these sizes to 100 and 1,000, respectively. In this section, we report experiments in which different sizes were tested to study the behavior of our proposed method with respect to these two values.

For the size of the instance subset, we tested values from 100 to 1,000: $\{100, 200, 300, 400, 500, 600, 700, 800,\ 900, 1,000\}$. The same values were used for the size of the feature subset. Although it is evident that the two sample sizes might have a joint effect, studying the effect of both hyperparameters simultaneously would require an infeasible number of experiments.

Figure 9 shows the behavior of the average values of the data reduction, κ and run-time metrics as the sample size for instances was varied from 100 to 1,000 in intervals of 100 instances. The results of the Nemenyi test for these three metrics are shown in Figure 2 of the supplementary material. Three conclusions can be extracted from these figures. First, the data reduction ability of our method was constant regardless of the subset size, indicating a robust algorithm. Second, very good classification performance could be obtained even with very small sample sizes. In fact, the Nemenyi test indicated that only the smallest sample size of 100 instances resulted in significantly worse performance. Third, as expected, the relationship between run time and sample size is approximately

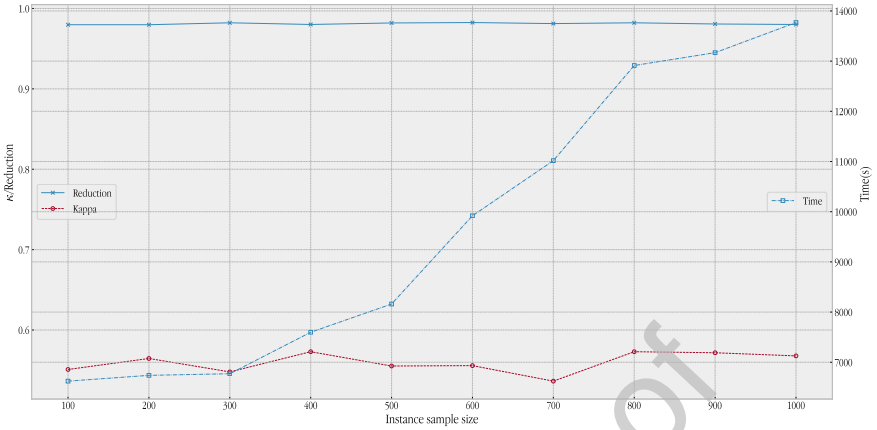


Figure 9: Behavior of SI(FS)^2 in terms of κ, data reduction and run time as a function of the instance sample size.

linear.

Figure 10 shows the behavior of the average values of the data reduction, κ and run-time metrics as the sample size for features was varied from 100 to 1,000 in intervals of 100 features. The results of the Nemenyi test for these three metrics are shown in Figure 3 of the supplementary material. As was the case for the instance sample size, our proposed method was found to be robust in terms of data reduction with respect to the size of the subset of features considered in each round. Regarding κ, we found two relevant results. First, the algorithm showed the same robustness for this hyperparameter, with almost constant behavior as the size of the feature subset was increased. Second, increasing the size of the feature subset tended to worsen the classification performance instead of improving it. The Nemenyi test showed that although there were no significant differences, smaller feature sample sizes had better Friedman's ranks.

An explanation for this effect may be found in the use of the roulette selection method for the features to be considered. Increasing the sample size increases the possibility that worse features will be considered and added to the subset of selected features. Because our method does not include a backward step, those worse features will remain in the subset of selected features, hindering

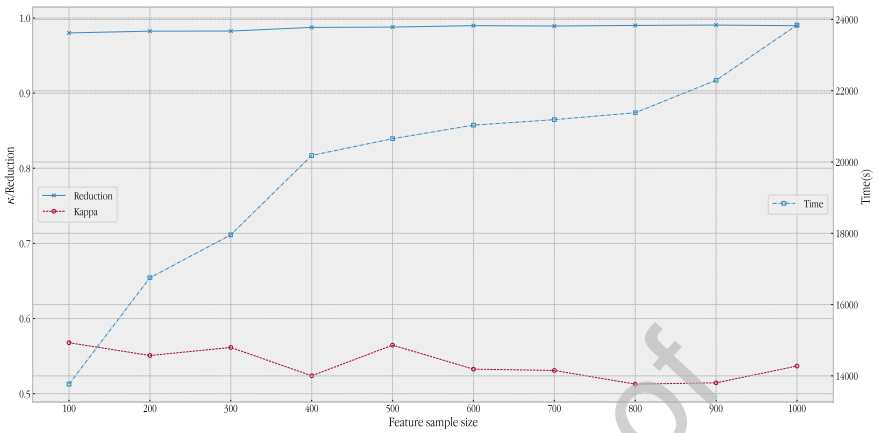


Figure 10: Behavior of SI(FS)^2 in terms of κ, data reduction and run time as a function of the feature sample size.

performance. Finally, the behavior of the run time of the algorithm was similar to that observed for the instance sample size.

6. Conclusions and future work

In this paper, we proposed a new heuristic instance and feature selection method designed to deal with datasets that are large in terms of the numbers of features and/or instances. We successfully applied the developed algorithm to datasets with millions of instances and features. The proposed method showed improved performance over the previous best methods in terms of three considered factors: data reduction, classification performance and run time.

Several new lines of future research are motivated by our proposal. One of these is the improvement of the search process using more powerful search heuristics than the simple forward selection heuristic used in this paper. Furthermore, as our method has proven its good performance on datasets with many features, it would be interesting to evaluate its behavior in certain application areas, such as microarray datasets, in which problems involving many features and few instances are common. Another interesting new line of research

is the adaptation of the method for class-imbalanced datasets.

References

[1] Y. Saeys, T. Abeel, S. Degroeve, Y. V. de Peer, Translation initiation site prediction on a genomic scale: beauty in simplicity, Bioinformatics 23 (2007) 418–423.

[2] P. Narendra, K. Fukunaga, Branch, and bound algorithm for feature subset selection, IEEE Transactions Computer C-26 (9) (1977) 917–922.

[3] L. Kuncheva, L. C. Jain, Nearest neighbor classifier: simultaneous editing and descriptor selection, Pattern Recognition Letters 20 (11) (1999) 1149–1156.

[4] H. M. Zawbaa, E. Emary, C. Grosan, V. Snasel, Large-dimensionality small-instance set feature selection: A hybrid bio-inspired heuristic approach, Swarm and Evolutionary Computation 42 (2018) 29–42.

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

[6] J. Pérez-Rodríguez, A. Arroyo-Peña, N. García-Pedrajas, Simultaneous instance and feature selection and weighting using evolutionary computation: Proposal and study, Applied Soft Computing 37 (2015) 416–443.

[7] S. S. S. Ahmad, W. Pedrycz, Feature and instance selection via cooperative pso, in: 2011 IEEE International Conference on Systems, Man, and Cybernetics, 2011, p. 2127–2132.

[8] J. R. Cano, F. Herrera, M. Lozano, Stratification for scaling up evolutionary prototype selection, Pattern Recognition Letters 26 (7) (2005) 953–963.

[9] J. Pérez-Rodríguez, A. de Haro-García, J. A. R. del Castillo, N. García-Pedrajas, A general framework for boosting feature subset selection algorithms, Information Fusion 44 (2018) 147–175.

[10] J. H. Chen, H. M. Chen, S. Y. Ho, Design of nearest neighbor classifiers: multi-objective approach, International Journal of Approximate Reasoning 40 (1-2) (2005) 3–22.

[11] D. B. Skalak, Prototype and feature selection by sampling and random mutation hill climbing algorithms, in: W. W. Cohen, H. Hirsh (Eds.), International Conference on Machine Learning, Morgan Kaufmann, San Francisco (CA), 1994, p. 293–301.

[12] B. V. Dasarathy, J. Sánchez, Concurrent feature and prototype selection in the nearest neighbour decision process, in: Proc. of 4th World Multiconference on Systemics, Cybernetics and Informatics, Vol. 7, Orlando (USA), 2000, p. 628–633.

[13] Y. Villuendas-Rey, C. Rey-Benguría, M. Lytras, C. Yanez-Marquez, O. Camacho, Simultaneous instance and feature selection for improving prediction in special education data, Program 51 (2017) 278–297.

[14] T. R. Babu, M. N. Murty, V. K. Agrawal, On simultaneous selection of prototypes and features in large data, in: S. K. Pal, S. Bandyopadhyay, S. Biswas (Eds.), Pattern Recognition and Machine Intelligence. PReMI 2005, Vol. 3776 of Lecture Notes in Computer Science, 2005, p. 595–600.

[15] L. Zhang, C. Chen, J. Bu, X. He, A unified feature and instance selection framework using optimum experimental design, IEEE Transactions on Image Processing 21 (5) (2012) 2379–2388.

[16] C.-F. Tsai, W. Eberle, C.-Y. Chu, Genetic algorithms in feature and instance selection, Knowledge-Based Systems 39 (2013) 240–247.

[17] J. Derrac, S. García, F. Herrera, Ifs-coco: Instance and feature selection based on cooperative coevolution with nearest neighbor rule, Pattern Recognition 43 (2010) 2082–2105.

[18] M. Suganthi, V. Karunakaran, Instance selection and feature extraction using cuttlefish optimization algorithm and principal component analysis using decision tree, Cluster Computing 22 (2019) S89–S101.

[19] X. Zhang, C. Mei, D. Chen, Y. Yang, A fuzzy rough set-based feature selection method using representative instances, Knowledge-Based Systems 151 (2018) 216–229.

[20] G. Mcdonald, N. García-Pedrajas, C. Macdonald, I. Ounis, A study of svm kernel functions for sensitivity classification ensembles with pos sequences, in: Proceedings of the the 40th International ACM SIGIR Conference, 2017, p. 1097–1100.

[21] Y. Villuendas-Rey, Y. Caballero-Mota, M. M. García-Lorenzo, Intelligent feature and instance selection to improve nearest neighbor classifiers, in: I. Batyrshin, M. G. Mendoza (Eds.), Proceesings of MICAI'12, Vol. 7629 of Lecture Notes on Artificial Intelligence, 2013, p. 27–38.

[22] N. García-Pedrajas, A. de Haro-García, Scaling up data mining algorithms: review and taxonomy, Progress in Artificial Intelligence 1 (1) (2012) 71–87.

[23] J. R. Vergara, P. A. Estévez, A review of feature selection methods based on mutual information, Neural Computing & Applications 24 (2014) 175–186.

[24] N. García-Pedrajas, J. A. R. del Castillo, D. Ortiz-Boyer, A cooperative coevolutionary algorithm for instance selection for instance-based learning, Machine Learning 78 (2010) 381–420.

[25] J. Demšar, Statistical comparisons of classifiers over multiple data sets, Journal of Machine Learning Research 7 (2006) 1–30.

[26] A. Ben-David, A lot of randomness is hiding accuracy, Engineering Applications of Artificial Intelligence 20 (7) (2007) 875–885.

[27] A. de Haro-García, G. Cerruela-García, N. García-Pedrajas, Instance selection based on boosting for instance-based learners, Pattern Recognition 96 (2019) 1–13.

[28] J. A. Olvera-López, J. A. Carrasco-Ochoa, J. F. Martínez-Trinidad, J. Kittler, A review of instance selection methods, Artificial Intelligence Reviews 34 (2010) 133–143.

[29] D. R. Wilson, T. R. Martinez, Reduction techniques for instance-based learning algorithms, Machine Learning 38 (2000) 257–286.

[30] C.-H. Chou, B.-H. Kuo, F. Chang, The generalized condensed nearest neighbor rule as a data reduction method, in: 18th International Conference on Pattern Recognition (ICPR'06), Vol. 2, 2006, p. 556–559.

[31] E. Marchiori, Hit miss networks with applications to instance selection, Journal of Machine Learning Research 9 (2008) 997–1017.

[32] H. Brighton, C. Mellish, Advances in instance selection for instance-based learning algorithms, Data Mining and Knowledge Discovery 6 (2002) 153–172.

[33] R. Barandela, F. J. Ferri, J. S. Sánchez, Decision boundary preserving prototype selection for nearest neighbor classification, International Journal of Pattern Recognition and Artificial Intelligence 19 (6) (2005) 787–806.

[34] H. Shin, S. Cho, Pattern selection for support vector classifiers, in: IDEAL'02, Vol. 2412 of Lecture Notes in Computer Science, 2002, p. 469–474.

[35] M. Sebban, R. Nock, J. H. Chauchat, R. Rakotomalala, Impact of learning set quality and size on decision tree performances, International Journal of Computers, Systems and Signals 1 (1) (2000) 85–105.

[36] B. D. Dasarathy, Minimal consistent set (mcs) identification for optimal nearest neighbor decision systems design, IEEE Transactions on Systems, Man and Cybernetics 24 (1994) 511–517.

Declaration of Interest

The authors declare that there's no financial/personal interest or belief that could affect their objectivity.

Nicolás García-Pedrajas is a Full Professor in the Department of Computing and Numerical Analysis at the University of Córdoba (Spain). He is the leading researcher in the Computation Intelligence and Bioinformatics Research Groups. He has published more the sixty papers in peer-reviewed journals, including the JCR. His major research interests include Data Mining, Classification, Bioinformatics and other aspects of Machine Learning.

Juan A. Romero del Castillo is an Associate Professor in Computer Science and Artificial Intelligence in the Department of Computing and Numerical Analysis at the University of Córdoba. His major research interests include Data Mining, Classification, Bioinformatics and other aspects of Machine Learning.

Gonzalo Cerruela-García is an Associate Professor in Computer Science and Artificial Intelligence in the Department of Computing and Numerical Analysis at the University of Córdoba. He received his Bachelor's of Electronic Engineering in 1989 and Ph.D. in Computer Science in 1999. His teaching responsibilities include the disciplines of Databases, Data Structures and Information, and Web and Mobile Engineering. His research activities are oriented toward new computational solutions in Chemistry and Computer Information Systems.