Memetic Comp. (2010) 2:183–199
DOI 10.1007/s12293-010-0048-1

REGULAR RESEARCH PAPER

Stratified prototype selection based on a steady-state memetic algorithm: a study of scalability

Joaquín Derrac · Salvador García · Francisco Herrera

Received: 22 September 2009 / Accepted: 9 July 2010 / Published online: 24 July 2010

Abstract Prototype selection (PS) is a suitable data reduction process for refining the training set of a data mining algorithm. Performing PS processes over existing datasets can sometimes be an inefficient task, especially as the size of the problem increases. However, in recent years some techniques have been developed to avoid the drawbacks that appeared due to the lack of scalability of the classical PS approaches. One of these techniques is known as stratification. In this study, we test the combination of stratification with a previously published steady-state memetic algorithm for PS in various problems, ranging from 50,000 to more than 1 million instances. We perform a comparison with some well-known PS methods, and make a deep study of the effects of stratification in the behavior of the selected method, focused on its time complexity, accuracy and convergence capabilities. Furthermore, the trade-off between accuracy and efficiency of the proposed combination is analyzed, concluding that it is a very suitable option to perform PS tasks when the size of the problem exceeds the capabilities of the classical PS methods.

J. Derrac (✉) · F. Herrera
Department of Computer Science and Artificial Intelligence, CITIC-UGR (Research Center on Information and Communications Technology), University of Granada, 18071 Granada, Spain
e-mail: jderrac@decsai.ugr.es

F. Herrera
e-mail: herrera@decsai.ugr.es

S. García
Department of Computer Science, University of Jaén, 23071 Jaén, Spain
e-mail: sglopez@ujaen.es

Keywords Data reduction · Memetic algorithm · Stratification · Scaling up · Prototype selection · Nearest neighbor rule

1 Introduction

In recent years, the size of the data which some data mining applications must manage has increased. Researchers in many fields such as biology, medicine and industry have developed more efficient and accurate data acquisition methods, which have allowed them to face greater and more difficult problems than before. As a result, the amount of data extracted to analyze those new challenges has grown to a point at which many classical data mining methods cannot work properly, or, at least, suffer several drawbacks in their application.

Data reduction [53] is a data preprocessing task which can be applied to ease the problem of dealing with large amounts of data. Its main objective is to reduce the original data by selecting the most representative information. In this way, it is possible to avoid excessive storage and time complexity, improving the results obtained by any data mining application, e.g., supervised classification. The best known data reduction processes are feature selection [42], feature generation [25], attribute discretization [41], instance generation [8,46] and instance selection [40,43].

Prototype selection (PS) is an effective data reduction process developed to improve the performance of the nearest neighbor (NN) rule. As a subtype of instance selection procedures, the objective of PS is to select the most promising examples (prototypes) from the training set, in order to avoid the excessive storage and time complexity of a k-NN classifier.

One of the most well known methods in supervised classification is the k-nearest neighbors classifier (k-NN) [13,51,55]. It is a non-parametric classifier which does not build a model in its training phase. Instead, k-NN predicts the class of a new prototype by computing a similarity measure [14] between it and all prototypes from the training set. Thus, the effectiveness of the classification process relies on the quality of the training data. It is also important to note that its main drawback is its relative inefficiency as the size of the problem grows, regarding both the number of examples in the data set and the number of attributes which will be used in the computation of its similarity functions (distances).

Evolutionary algorithms (EAs) [16] are general-purpose search algorithms that use principles inspired by natural genetic populations to evolve solutions to problems. They have been successfully employed in different data mining [1,20,23] problems. Given that the PS problem can be defined as a combinatorial problem, EAs have been used to solve it with promising results [10], in a process named evolutionary prototype selection (EPS).

The combination of EAs with local search (LS) is called memetic algorithm (MA) [48,50]. Formally, an MA is defined as an EA that includes one or more LS phase within its evolutionary cycle [36]. MAs have been shown to be more efficient (i.e., needing fewer evaluations to find optima) and more effective (identifying higher quality solutions) than traditional EAs for some problem domains. In the literature, we can find a lot of applications of MAs to different problems; see [29] for an understanding of MA issues, and [4,18,30] for examples of MAs applied to different domain problems.

The increase in the size of the database is a staple problem in PS (which is known as the *scaling up* problem). This problem produces excessive storage requirements, increases time complexity and affects generalization accuracy. These drawbacks are also present in EPS because they result in an increment in chromosome size and time execution and also involve a decrease in the convergence capabilities of the EA [19]. Traditional EPS approaches generally suffer from excessively slow convergence between solutions because of their failure to exploit local information. This often limits the practicality of EAs on many large-scale problems where computational time is a crucial consideration.

Several strategies have been proposed to tackle the *scaling up* problem in various domains [52]. Some of them deal with the data mining algorithms themselves. They try to scale up the algorithms by proposing faster and lower resource consumption approaches [6,32,44]. Another interesting group of techniques works directly with the data set. They split the data set into various parts to make the application of a data mining algorithm easier, employing after its execution a mechanism to join the solutions of each part in a global solution. Some of these approaches have appeared recently in the literature [11,26].

The interest of this paper is to analyze the scaling up of a previously proposed Steady-State Memetic Algorithm for EPS (SSMA-PS) [22] when it is applied in conjunction with the stratification procedure [11], a successful technique already developed to employ PS in large sized problems. The existing synergy between these two approaches will help SSMA-PS to tackle successfully greater problems, ranging from 50,000 to 1 million instances.

Several aspects of this combination will be covered: The resulting runtime of the algorithm as the size of strata selected varies, the effects of the employment of stratification over the quality of the final selected subsets, and how the use of stratification affects the convergence capabilities of SSMA-PS.

An experimental study will be carried out to test the proposed combination. Their results will be compared with some classical proposals for PS, by analyzing the results obtained for two disjointed objectives: Accuracy and efficiency. A complete analysis will be performed in order to highlight which method has a better performance when applied in combination with stratification.

The rest of the paper is organized as follows: Sect. 2 shows an introduction to the main topics of our study. Section 3 discusses the usefulness of the application of the stratification strategy to SSMA-PS. Section 4 deals with the experimental framework used. Section 5 presents the analysis of results and presents a discussion of several issues related to the use of stratification. Section 6 presents the conclusions arrived at. Finally, a brief description of the classical PS methods employed is given in Appendix A.

2 Preliminaries: prototype selection, steady-state memetic algorithm, scaling up and stratification

This section discusses the main topics in the field in which our contribution is based:

- In Sect. 2.1, we define PS and discuss the influence of EAs on its development in recent years.
- In Sect. 2.2, we show the main features of SSMA-PS and its local search procedure.
- In Sect. 2.3, we describe the main issues of the *scaling up* problem.
- In Sect. 2.4, we detail the characteristics of the stratification procedure.

2.1 Prototype selection

PS methods are instance selection methods [43] developed to improve the performance of the NN rule. They try to isolate the smallest set of instances which enable k-NN to predict the class of a query instance with the same quality as the initial data set. By minimizing the data set size, it is possible to reduce the space complexity and decrease the

computational cost of k-NN, improving their generalization capabilities through the elimination of noise.

The PS problem can be defined as follows: Let X be a prototype where $X = (X_1, X_2, \ldots, X_m, X_c)$, belonging to a class c given by X_c, and an m-dimensional space in which X_i is the value of the ith feature of the sample. Then, let us assume that there is a training set TR which consists of N instances and a test set TS composed of T instances. Let $S \subseteq TR$ be the subset of selected samples that resulted from the execution of a PS algorithm, then we classify a new pattern X from TS by the k-NN rule acting over S.

With respect to its objective, PS methods can be categorized in three classes: preservation methods, which aim to obtain a consistent subset from the training data, ignoring the presence of noise; noise removal methods, which aim to remove noise both in the boundary points (instances near to the decision boundaries) and in the inner points (instances far from the decision boundaries), and hybrid methods, which perform both objectives simultaneously.

In the data mining field many approaches to PS have been developed, ranging from classical approaches such as CNN [27] or ENN [59] to recent approaches such as SSMA-PS [22], HMNEI [47] or PSC [49]. A wide number of reviews of PS methods can be found in the literature [8,34,35,60].

A high number of the newest proposals of PS methods are based on EAs. The first appearance of the application of an EA to the PS problem can be found in [37], describing the application of a genetic algorithm (GA) to select a reference set for the k-NN rule. It was improved later, in Ishibuchi and Nakashima [33] and Kuncheva and Bezdek [38]. Although some of the following proposals of EPS methods were based on different strategies (e.g. [57], which employed an Estimation of Distribution Algorithm), the majority of them continued to employ a GA as their search procedure. Ho et al. [31], where a GA design for obtaining an optimal NN classifier based on orthogonal arrays is proposed, is the most representative example from this period.

Later, in [10], the behavior of four evolutionary models (generational GAs (GGAs), steady-state GAs (SSGAs), the CHC model [17] and Population Based Incremental Learning (PBIL) [7]) was analyzed. They recommended employing, as the fitness function, the average of the classification rate obtained by using a 1-NN classifier and the reduction ratio obtained by the current selected set of prototypes.

One of the newest approaches to EPS is SSMA-PS [22]. It incorporates an ad hoc local search specifically designed to optimize the search with the aim of tackling the scaling up problem. Another recent proposal [24], is focused on enhancing the fitness function and the mutation and crossover operators when applied to PS problems.

The properties of EAs in PS are discussed in [10] where the authors differentiate between the selection based on heuristics (which appears in classic non-evolutionary PS algorithms, for example CNN, IB3 or DROP described in [60]) and the selection developed by EPS algorithms. EPS presents a strategy that combines inner and boundary points. It does not tend to select instances depending on their a priori position in the search space (inner class or limit ones). EPS selects the instances that increase the accuracy rates independently of their a priori position.

An interesting question to answer when employing any EPS method (and indeed any PS method) is to determine the k value employed in the k-NN classifier. In [60], the authors suggest that the determination of k may depend on the proposal of the PS algorithm. Setting k as greater than 1 decreases the sensitivity of the algorithm to noise and tends to smooth the decision boundaries. In some PS algorithms, a value $k > 1$ may be convenient, when the interest lies in protecting the classification task of noisy instances. Therefore, the authors state that it may be appropriate to find a value of k to use during the reduction process, and then redetermine the best value of k in the classification task.

For EPS methods, however, the value $k = 1$ is employed most of the time, in order to achieve the greatest possible sensitivity to noise during the reduction process. In this manner, an EPS algorithm could better detect the noisy instances and the redundant ones in order to find a good subset of instances perfectly adapted to the NN rule. By considering only an instance during the evolutionary process, the reduction-accuracy trade-off is more balanced and the efficiency is improved.

2.2 Steady-state memetic algorithm model for prototype selection problem

The SSMA-PS algorithm employs a Steady-State GA as its global search procedure, which allows it to integrate global and local searches more tightly than generational MAs. This interweaving of the global and local search phases allows the two to influence each other, e.g., the SSGA chooses good starting points, and the LS provides an accurate representation of that region of the domain.

The main characteristics of SSMA-PS are:

- *Representation*: Let us assume a data set denoted TR with N instances. The search space associated with the instance selection of TR is constituted by all the subsets of TR. Therefore, the chromosomes should represent subsets of TR. This is achieved by using a binary representation. A chromosome consists of N genes (one for each instance in TR) with two possible states: 0 and 1. If the gene is 1, then its associated instance is included in the subset of TR represented by the chromosome. If it is 0, then this does not occur.
- *Population Initialization*: The chromosomes of the initial population are initialized randomly.

– *Fitness Function*: Let S be a subset of instances of TR to evaluate and be coded by a chromosome. The fitness function is defined considering $clas\_rat$, the number of instances correctly classified using the 1-NN classifier and $perc\_red$, the percentage of reduction achieved with regard to the original size of the training data. The evaluation of S is carried out by considering all the training set TR. For each object y in S, the NN is searched for among those in the set $S\backslash\{y\}$.

$$Fitness(S) = w_\alpha \cdot clas\_rat + (1 - w_\alpha) \cdot perc\_red \quad (1)$$

The objective of the MA is to maximize the weighted sum of the two measures defined (w_α is valued in the interval [0,1]). Note that the fitness function is the same as that used by EPS models previously proposed [10].
– *Parent Selection Mechanism*: In order to select two parents for applying the evolutionary operators, binary tournament selection is employed.
– *Genetic operators*: SSMA-PS employs a one point crossover operator and the classical mutation operator (change 0 to 1 and vice versa)
– *Replacement Strategy*: SSMA-PS uses the standard replacement strategy employed by the classical Steady-State genetic algorithms.
– *Local Search*: SSMA-PS includes a local search specifically designed for PS tasks. From a given chromosome, it considers neighborhood solutions by removing an instance from the current selected subset (i.e., changing 1 to 0 in a gene). These changes are accepted if they improve the classification accuracy of the current solution. However, if premature convergence is detected, improvements in the fitness value (i.e. changes which improve $perc\_red$ but decrease $clas\_rat$) are also accepted, in order to allow the algorithm to escape from local optimum.
– *Mechanism of Local Search Application*: It is necessary to control the operation of the LS over the total visited solutions. This is because the additional function evaluations required for a total search can be very expensive and the MA in question could become a multi-restart LS and not take advantage of the qualities of the EAs. In order to avoid this drawback, SSMA-PS includes the *Adaptive* P_{LS}*Mechanism*, which is an adaptive fitness-based method that is very simple but offers good results in [45]. Indeed, this scheme assigns an LS probability value to each chromosome generated by crossover and mutation,c_{new}:

$$P_{LS} = \begin{cases} 1 & \text{if } f(c_{new}) \text{ is better than } f(C_{worst}) \\ 0.0625 & \text{otherwise} \end{cases} \quad (2)$$

1. Initialize population.
2. While (*not termination-condition*) do
3. Use Binary Tournament to select two parents
4. Apply crossover operator to create offspring (Off_1, Off_2)
5. Apply mutation to Off_1 and Off_2
6. Evaluate Off_1 and Off_2
7. For each Off_i
8. Invoke Adaptive-P_{LS}-mechanism to obtain P_{LS_i} for Off_i
9. If $u(0,1) < P_{LS_i}$ then
10. Perform meme optimization for Off_i
11. End if
12. End for
13. Employ standard replacement for Off_1 and Off_2
14. End while
15. Return the best chromosome

Fig. 1 Pseudocode of SSMA-PS

where f is the fitness function and C_{worst} is the current worst element in the population. As was observed by Hart [28], applying LS to as little as 5% of each population results in a faster convergence towards good solutions.
– *Evaluation Mechanisms*: When evaluating the fitness of a chromosome, it is possible to distinguish between *Total evaluation* and *Partial evaluation*.

  – *Total Evaluation*: consists of a standard evaluation of the performance of a chromosome in EPS. Total evaluations always take place outside the optimization procedure, that is, within the evolutionary cycle.
  – *Partial Evaluation*: can be carried out on a neighbor solution of a current instance that has already been evaluated and differs only in one bit position changed from value 1 to 0. The cost of a partial evaluation is given by Eq. 3

$$PE = \frac{N_{nu}}{n} \quad (3)$$

  where N_{nu} is the number of neighbors updated when a determined instance is removed by meme procedure and $n = |TR|$ is the size of the original set of instances (also the size of the chromosome). Partial evaluations always take place inside the local optimization procedure.

  The SSMA-PS computes total evaluations; when a partial evaluation is considered, SSMA-PS adds to the counter evaluation variable the respective partial value PE (expression 3). Therefore, a certain number of partial evaluations (depending on the PE values) will be considered as a total evaluation.
– *Termination Condition*: The MA continues carrying out iterations until a specified number of evaluations is reached.

Figure 1 shows the SSMA-PS pseudocode. A complete description of it can be found in [22].

2.3 The scaling up problem

The scaling up problem appears when the number of training samples increases beyond the capacity of the traditional data mining algorithms, harming their effectiveness and efficiency. Due to large size data sets, it produces excessive storage requirements, increases time complexity and affects generalization accuracy.

This problem is also present when applying PS methods to large data sets. The NN rule presents several shortcomings discussed in [60], with the most well known being: the necessity of storing all the training instances in order to carry out the classification task; and its high time consumption in classification due to it having to search through all available instances to classify a new input vector.

Furthermore, these drawbacks are even more problematic when applying PS methods. The most important are:

Time complexity The efficiency of the majority of PS algorithms is at least, $O(N^2)$, with N being the number of instances in the data set; moreover, it is not difficult to find classical approaches with an efficiency order of $O(N^3)$, or more (e.g. SNN [54]). The implication of this is that, although the methods could be very suitable when applied to small data sets, their application would become impossible as the size of the problem increases (for instance, if the size of a data set increased ten times, the cost of applying SNN to it would increase by a thousand, which may become prohibitive).

Memory consumption The majority of the PS methods need to have the complete data set stored in their memory in order to be carried out. If the size of the data set is too big, the computer will be unable to execute the method or, at least, would need to use the hard disk to swap memory, which would have an adverse effect on efficiency due to the increased amount of access to the hard disk.

Generalization PS methods are affected in their generalization capabilities due to the noise and over fitting effect introduced by larger size data sets. Most of the classical PS methods are developed to handle small and medium sized data sets, and may not be suitable to find and delete every noisy or irrelevant instance in a greater data set.

EPS methods must also face these drawbacks. In addition, a new problem emerges when applying them to large data sets:

Representation EPS methods are also affected by representation, due to the increment in the size of their chromosomes. When the size of these chromosomes is too big, the algorithms experience convergence difficulties, as well as costly computational time. This drawback affects both time consumption and the quality of the solutions obtained, thus the search process would be ineffective in such large search spaces.

These drawbacks produce a considerable degradation in the behavior of PS algorithms. However, an important consideration must be taken into account: Assuming that a PS method is able to handle a problem, the time elapsed during its execution is not as important as the quality of the selected subset and the reduction rate achieved. In a similar way to which a decision tree or a neural net are built, the PS process need only be performed once; by contrast, the classification process with the selected subset must be performed every time a new example is required to be classified. Thus, a good PS should offer a high quality subset of instances, simultaneously achieving the best possible reduction rate. Thereby, a highly accurate and quick classification process could be carried out by the k-NN classifier.

2.4 Stratification for prototype selection

As we have stated, the *scaling up* problem has several drawbacks which can produce a considerable degradation in the behavior of PS algorithms. To avoid this, in this study we will employ the stratification strategy proposed in [11].

The stratification strategy splits the training data into disjoint strata with equal class distribution. The initial data set D is divided into two sets, TR and TS, as usual (e.g. a tenth of the data for TS, and the rest for TR in 10-fold cross validation). Then, TR is divided into t disjoint sets D_j , strata of equal size, D_1, $D_2 \ldots D_t$, maintaining class distribution within each subset. In this manner, the subsets TR and TS can be represented as follows:

$$TR = \bigcup_{j=1}^{t} D_j \tag{4}$$

$$TS = D \backslash TR \tag{5}$$

Then, a PS method should be applied to each D_j, obtaining a selected subset DS_j for each partition. The prototype selected set is obtained joining every DS_j obtained, and it is called stratified prototype subset selected (SPSS).

$$\text{SPSS} = \bigcup_{j=1}^{t} DS_j \tag{6}$$

When the SPSS has been obtained, the k-NN classifier can be applied to TS, employing SPSS as training data. Figure 2 shows the basic steps of the process.

The employment of the stratification procedure does not have a great cost in time. Usually, the process of splitting the training data into strata, and joining them when the PS method has been applied, is not time-consuming, as it does not require any kind of additional processing. Thus, the time needed for the stratified execution is almost the same as that taken in the execution of the PS method in each strata, which

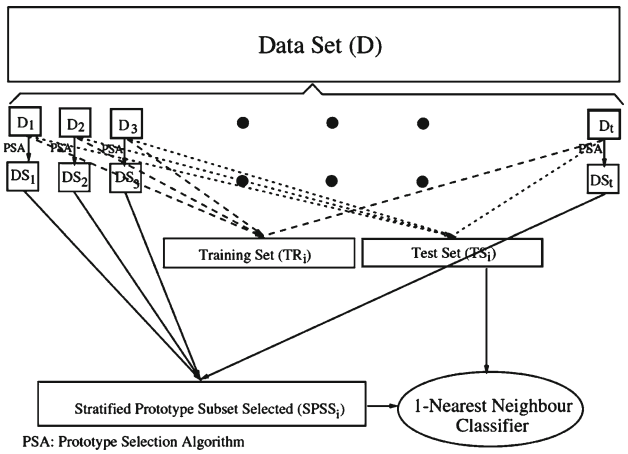

Fig. 2 Scheme of the stratification strategy

is significantly lower than the time spent if no stratification is applied, due to the time complexity of the PS method, which most of the times is $O(N^2)$ or higher.

The prototypes present in TR are independent of each other, so the distribution of the data into strata will not degrade their representation capabilities if the class distribution is maintained. The number of strata, which should be fixed empirically, will determine the size of them. By using a proper number it is possible to greatly reduce the training set size. This situation allows us to avoid the drawbacks that appeared due to the *scaling up* problem.

3 Analysis of the usefulness of the stratified strategy with SSMA-PS

In this section, we analyze the scalability of SSMA-PS when combined with the stratification procedure. Although SSMA-PS has been shown to be a suitable option for medium sized data sets, it may also be overwhelmed by larger problems of hundreds of thousands instances. It is at this point where the stratification strategy can demonstrate its usefulness.

Although there are several parameters which affect the efficiency of SSMA-PS, we will only consider the number of instances of the training set, N. Undoubtedly, the number of features considered in each instance and the number of evaluations employed in the search process will have a strong effect on the final efficiency order of SSMA-PS, but they are not affected by the employment of stratification (i.e. the stratification strategy affects the number of instances of the training set, but the number of features and the number of evaluations remains unchanged).

Looking at the pseudocode of SSMA-PS (Fig. 1), we can highlight the full evaluation of an individual, which is $O(N^2)$, the initialization procedure, which is $O(N^2)$ (since it includes the creation and full evaluation of a fixed number of chromosomes), and the meme optimization procedure, which is also $O(N^2)$, as the most costly parts of the algorithm. The rest of the operations of the algorithm have a lesser cost in terms of instances. Thus, we can estimate a final cost for SSMA-PS of $O(N^2)$.

Then, let us assume that we have to apply SSMA-PS to a large data set, employing M strata, each of one containing N/M training samples. Thus, SSMA-PS would have time complexity $O((N/M)^2)$. Since it is necessary to apply SSMA-PS over all strata (M times), the full application of the stratified strategy will result in a total time complexity $O(M * (N/M)^2) = O(N^2/M)$. Compared to the non-stratified time complexity of $O(N^2)$, the stratification thus leads to a reduction in time complexity of order $O(1/M)$ (i.e. a speedup of M). This means that the application of the stratification strategy can reduce the runtime, at least, M times. Moreover, the employment of stratification also allows to process strata in parallel, thus the reduction in time complexity could be increased up to $O(1/M^2)$ (i.e. a speedup of M^2), if all the strata are processed in parallel.

The same benefits would appear when employing the stratification strategy with other PS methods, or when applying them to other data sets. The concrete gain obtained would vary, depending on both situations, but it would be greater the more inefficient the PS method, and as the size of the problem increases. This is the reason why the stratification strategy is particularly recommended when facing large scale problems.

4 Experimental framework

This section describes the experimental framework employed to test our proposal. It is organized as follows:

- Section 4.1 enumerates the comparison methods employed.
- Section 4.2 fully describes the data sets selected for the experiments.
- Section 4.3 shows the parameters selected for each comparison method and for the stratification strategy.
- Section 4.4 discusses the performance measures selected to analyze the results and the statistical procedures employed to contrast them.

4.1 Comparison methods

To test the performance of SSMA-PS, we have employed two well-known PS methods for comparison: Decremental Reduction Optimization Procedure 3 (DROP3) [60] and Fast Condensed Nearest Neighbor (FCNN) [3]. DROP3 was selected because it is a classical method for PS which should offer suitable behavior over most domains. FCNN was

Table 1 UCI Data sets used in our experiments

Data set	Instances	Attributes	Classes	# Strata	Instances/Strata	Class distribution (%)					
						1	2	3	4	5	Rest
Adult	48,842	14	2	5	9,768	76.07	23.92				
Census	299,285	41	2	30	9,980	93.79	6.20				
Connect-4	67,557	42	3	7	9,651	65.83	24.62	9.55			
Fars	100,968	29	8	10	10,097	41.71	19.81	14.92	13.75	8.59	<1%
KddCup	494,020	41	23	50	9,880	56.84	21.69	19.69	<1%	<1%	<1%
PokerHand	1,025,010	10	10	100	10,250	50.11	42.25	4.76	2.11	<1%	<1%
Shuttle	58,000	9	7	6	9,667	78.59	15.35	5.63	<1%	<1%	<1%

selected due to it being one of the fastest PS methods proposed in the literature. A short description of them can be found in Appendix A.

SSMA-PS, DROP3 and FCNN have been employed along with the stratification procedure in all the problems. Additionally, DROP3 and FCNN have been employed without stratification in those problems where the computational cost was not too high, to have a reference of how the use of stratification modifies their behavior (they are denoted as DROP3-No Stratification and FCNN-No Stratification). Also, the 1-NN classifier has been employed as a baseline measure of accuracy and the time elapsed in the classification phase.

All the methods have been run on an Intel Core 2 Quad (2.50Ghz) with 32 KB of L1 cache, 3MB of L2 cache and 8GB of RAM. We have used standard Java implementations of the methods, coding them as they are described in their respective papers (i.e. we have not performed any additional optimization to their original definition, apart from the use of stratification). They have been adapted from the existing KEEL project implementations,[1] performing the necessary changes to combine them with the stratification procedure.

4.2 Data sets

Seven large data sets have been selected for this study. They have been obtained at the UCI Machine Learning repository [5] (except the Fars data set[2]).

Each dataset has been partitioned to employ a 10-folds cross validation procedure. Moreover, numerical attributes have been normalized to the interval [0,1], to obtain better behavior when employing the NN rule.

It is also important at this point to select the concrete stratification scheme to employ. As we related before, the use of the stratification procedure requires making the decision of how many strata to employ to split the data. In this experimental study, we have fixed a number of strata for each data set, trying to set the size of each strata as near as possible to 10,000 instances, which corresponds to a medium sized problem which SSMA-PS, DROP3 and FCNN should be able to handle adequately.

Table 1 summarizes the characteristics of the data sets:

- The name of the data set.
- The number of instances (examples), attributes (features) and classes.
- The number of strata fixed.
- The number of instances per strata.
- The distribution of instances per class, in descending order.

4.3 Parameters

The parameters employed are the following:

SSMA-PS: Population size = 30, Evaluations = 10,000, Cross probability per bit = 0.5, Mutation probability per bit = 0.001, $w_\alpha = 0.5$.
All methods: $k = 1$, Distance function: Euclidean.

The w_α parameter of the fitness function of SSMA-PS has been set to 0.5, following the recommendations given in [10] for a general fitness function for evolutionary PS methods. The rest of the parameters have been kept at their default values (k is the number of neighbors considered for the k-NN rule).

4.4 Performance measures and statistical analysis

To analyze the results obtained in the study, we have employed four performance measures:

Classification rate The classification rate is the number of successful hits relative to the total number of classifications. It has been by far the most commonly used metric

[1] http://sci2s.ugr.es/keel/.

[2] It can be obtained at ftp://ftp.nhtsa.dot.gov/FARS/.

for assessing the performance of classifiers for years [2,61,39].

Kappa rate is an alternative to classification rate, that compensates for random hits [12]. Equation 7 shows how it can be computed:

$$kappa = \frac{n \sum_{i=1}^{C} x_{ii} - \sum_{i=1}^{C} x_{i.} x_{.i}}{n^2 - \sum_{i=1}^{C} x_{i.} x_{.i}} \quad (7)$$

where x_{ii} is the cell count in the main diagonal of the confusion matrix, n is the number of examples, C is the number of class values, and $x_{.i}$, $x_{i.}$ are the columns' and rows' total counts, respectively.

The main difference between classification rate and Cohen's kappa rate is the scoring of the correct classifications. Classification rate scores all the successes over all classes, whereas Cohen's kappa scores the successes independently for each class and aggregates them. The second way of scoring is less sensitive to randomness caused by different numbers of examples in each class, which causes a bias in the learner towards obtaining data-dependent models.

Reduction rate The reduction rate is defined as the ratio of selected instances by the PS algorithm. It has a strong influence on the efficiency of the solutions obtained, due to the cost of the final classification process performed by the 1-NN rule ($O(N^2)$).

Time The simplest way to measure the practical efficiency of a method. Times elapsed in the execution of the PS methods, and in their subsequent training and test classification phases will be analyzed.

In addition, to complete the experimental study, we have performed a statistical comparison of accuracy measures between SSMA-PS and the comparison methods by using a nonparametric test. Concretely, we have decided to employ the Wilcoxon Signed-Ranks test [56,58], as recommended in [15,21,22].

The reason for employing it, instead of a parametric one, is that parametric tests are based on assumptions (for example, independence, normality or homoscedasticity) which are most likely to be violated when analyzing the performance of computational intelligence and data mining algorithms [62]. This fact is widely discussed, along with some case studies and several statistical procedures, in the SCI2S thematic public Website on *Statistical Inference in Computational Intelligence and Data Mining*.[3]

[3] http://sci2s.ugr.es/sicidm/.

5 Results obtained and discussion

This section shows the full results obtained in the experimental study, and discusses several issues about the employment of the stratification procedure in the enhancement of SSMA-PS:

- Section 5.1 shows the results obtained in the study.
- Section 5.2 gives a complete analysis of the results, and discuss the benefits and drawbacks of our proposal.
- Section 5.3 discusses the complexity/runtime trade-off of the employment of stratification.
- Section 5.4 shows a study of the effects of the use of stratification in the accuracy of SSMA-PS.
- Section 5.5 analyzes the effects of the stratification strategy in the convergence capabilities of SSMA-PS.
- Section 5.6 discusses the accuracy and efficiency trade-off achieved by every method of the study.

5.1 Experimental results

The results of the experimental study carried out are shown as follows (results have been averaged by employing a 10-fold cross validation procedure):

Accuracy Table 2 shows the average classification rate results (mean and standard error) obtained in training and test phases. The best results in the test phase are highlighted in bold.

Table 3 shows the average kappa rate results (mean and standard error) obtained in training and test phases. The best results in the test phase are highlighted in bold.

Efficiency Table 4 shows the average reduction rates obtained (excluding 1-NN). The best results are highlighted in bold.

The reduction rate achieved has a strong influence on the efficiency of the training and test classifications phases, since it is directly related to the final size of the training subset selected. For example, if SSMA-PS has achieved a reduction rate of 95.60 in *PokerHand* domain, it means that the original training set has been reduced to 4.4% of its original size.

Table 5 shows the time elapsed (in seconds) in the training phase (the execution of the PS method in all strata) and in the test phase (the classification of the test set employing the reduced training set obtained by the execution of the stratified PS method), both in stratified and non-stratified mode. Stratified times are computed as the sum of the time elapsed in every strata. Note that the column 1-NN means the time employed in the classification phase without applying a PS phase.

Table 2 Classification rate

Algorithms	SSMA-PS		DROP3		FCNN		DROP3-No stratification		FCNN-No stratification		1-NN	
	Training	Test	Training	Test	Training	Test	Training	Test	Training	Test	Training	Test
Adult	83.34 ± 0.07	**82.77** ± 0.41	80.32 ± 0.24	78.59 ± 0.49	73.38 ± 0.07	74.95 ± 0.33	81.76 ± 0.22	79.78 ± 0.45	81.01 ± 0.09	80.43 ± 0.31	79.55 ± 0.07	79.60 ± 0.35
Census	94.27 ± 0.03	**94.28** ± 0.11	91.22 ± 0.13	91.05 ± 0.20	87.05 ± 0.09	87.12 ± 0.14	–	–	93.68 ± 0.11	93.18 ± 0.18	92.50 ± 0.01	92.53 ± 0.1
Connect-4	68.23 ± 0.21	**67.52** ± 0.44	66.94 ± 0.14	64.93 ± 0.54	60.98 ± 0.18	62.31 ± 0.37	69.11 ± 0.15	66.50 ± 0.59	66.27 ± 0.17	65.57 ± 0.36	67.11 ± 0.07	67.20 ± 0.36
Fars	76.06 ± 0.14	**75.73** ± 0.32	75.94 ± 0.12	74.44 ± 0.49	70.88 ± 0.08	71.61 ± 0.32	77.68 ± 0.14	72.38 ± 0.47	75.25 ± 0.09	74.14 ± 0.27	74.66 ± 0.03	74.66 ± 0.34
KddCup	99.78 ± 0.02	99.77 ± 0.03	99.86 ± 0.02	99.84 ± 0.03	99.87 ± 0.00	99.87 ± 0.02	–	–	99.82 ± 0.02	99.80 ± 0.03	99.93 ± 0.00	**99.93** ± 0.01
PokerHand	52.45 ± 0.04	**52.28** ± 0.13	51.67 ± 0.04	51.17 ± 0.17	50.44 ± 0.03	48.11 ± 0.11	–	–	51.92 ± 0.04	51.71 ± 0.13	50.44 ± 0.03	50.34 ± 0.10
Shuttle	99.75 ± 0.04	99.75 ± 0.06	96.69 ± 1.93	96.65 ± 1.90	99.89 ± 0.01	99.91 ± 0.04	97.90 ± 1.96	97.87 ± 1.99	99.71 ± 0.02	99.62 ± 0.05	99.93 ± 0.01	**99.93** ± 0.04
Average	81.98 ± 0.08	**81.73** ± 0.21	80.38 ± 0.37	79.52 ± 0.55	77.50 ± 0.07	77.70 ± 0.19	–	–	81.09 ± 0.08	80.64 ± 0.19	80.59 ± 0.03	80.60 ± 0.19

Table 3 Kappa rate

Algorithms	SSMA-PS		DROP3		FCNN		DROP3-No stratification		FCNN-No stratification		1-NN	
	Training	Test	Training	Test	Training	Test	Training	Test	Training	Test	Training	Test
Adult	49.93 ± 0.04	**49.24** ± 0.20	44.37 ± 0.13	41.50 ± 0.25	33.28 ± 0.03	34.53 ± 0.12	46.21 ± 0.12	42.87 ± 0.23	47.46 ± 0.04	45.43 ± 0.15	43.23 ± 0.04	43.53 ± 0.15
Census	37.73 ± 0.01	**37.75** ± 0.04	31.04 ± 0.04	30.00 ± 0.06	23.61 ± 0.03	23.82 ± 0.05	–	–	37.22 ± 0.03	36.01 ± 0.05	35.01 ± 0.01	34.77 ± 0.03
Connect-4	30.51 ± 0.12	**29.82** ± 0.23	29.44 ± 0.06	29.06 ± 0.27	25.48 ± 0.09	26.78 ± 0.19	31.07 ± 0.07	29.76 ± 0.22	30.03 ± 0.11	29.72 ± 0.16	23.66 ± 0.04	23.61 ± 0.15
Fars	67.03 ± 0.10	**66.95** ± 0.23	66.83 ± 0.09	65.41 ± 0.35	60.99 ± 0.06	61.73 ± 0.23	67.98 ± 0.12	63.24 ± 0.31	66.54 ± 0.05	65.71 ± 0.26	65.65 ± 0.02	65.66 ± 0.24
KddCup	99.63 ± 0.02	99.62 ± 0.03	99.77 ± 0.02	99.73 ± 0.02	99.78 ± 0.01	99.79 ± 0.02	–	–	99.72 ± 0.01	99.61 ± 0.02	99.91 ± 0.00	**99.89** ± 0.01
PokerHand	12.77 ± 0.01	**12.81** ± 0.03	10.68 ± 0.01	10.13 ± 0.04	10.42 ± 0.01	10.26 ± 0.03	–	–	12.75 ± 0.02	12.63 ± 0.05	12.64 ± 0.01	12.52 ± 0.03
Shuttle	99.61 ± 0.04	99.60 ± 0.07	95.64 ± 1.76	91.28 ± 1.82	99.75 ± 0.01	99.73 ± 0.03	97.34 ± 0.96	97.01 ± 1.11	99.51 ± 0.01	99.43 ± 0.03	99.86 ± 0.01	**99.82** ± 0.04
Average	56.74 ± 0.05	**56.54** ± 0.12	53.97 ± 0.30	52.44 ± 0.40	50.47 ± 0.03	50.95 ± 0.10	–	–	56.18 ± 0.04	55.51 ± 0.10	54.28 ± 0.02	54.26 ± 0.09

Table 4 Reduction rate

	SSMA-PS	DROP3	FCNN	DROP3-No stratification	FCNN-No stratification
Adult	**98.96**	87.38	66.56	89.31	88.31
Census	**99.75**	95.37	84.46	96.18	96.43
Connect-4	**98.00**	78.39	47.14	85.43	92.31
Fars	**97.23**	86.00	61.38	90.92	94.56
KddCup	**99.73**	99.28	99.08	99.21	99.34
PokerHand	**95.60**	73.51	28.98	84.53	92.17
Shuttle	**99.81**	99.15	99.34	99.21	99.46
Average	**98.44**	88.44	69.56	92.11	94.65

5.2 Analysis and discussion

In this subsection, we will perform an extensive analysis of all the results obtained. We will focus our attention on two topics:

- Accuracy obtained in the test phase (Sect. 5.2.1). We will analyze results employing classification rate and kappa rate measures, to find which method had a better performance in the experimental study.
- Efficiency of the methods (Sect. 5.2.2). We will discuss the reduction rates achieved and the time elapsed for each method, highlighting which method presents the most efficient classification phase.

5.2.1 Accuracy

As we can see in Tables 2 and 3, our proposal shows the best behavior in most of the domains considered. SSMA-PS is able to outperform the rest of the methods in all problems except *shuttle* and *kddcup*. Looking at the tables, we can see three important results:

- SSMA-PS obtains the best average results both in kappa rate and classification rate measures. The combination with the stratification strategy allows it to tackle the problems considered and select a proper subset of prototypes which increases the predictive power of the 1-NN classifier.
- DROP3 and FCNN achieve worse results than 1-NN. Although they are good methods to handle standard problems, the accuracy obtained from their selected subsets decreases when the size of the problems increases too much.
- The employment of stratification, although it reduces the computational cost of tackling these larger domains, has reduced the accuracy results of DROP3 and FCNN. The non-stratified version of these methods achieves better results in most of the problems. However, the combination of SSMA-PS and stratification still achieves better accuracy results than the non-stratified methods.

To contrast these results, we have performed two two-tailed Wilcoxon Signed-Ranks Tests [62] comparing SSMA-PS with the respective comparison algorithms, one for each accuracy measure. The results are shown in Tables 6 and 7. These tables show the R^+ and R^- values achieved and their associated p value.

In terms of classification rate, SSMA-PS improves significantly over DROP3 and FCNN (N-S) (with a level of significance 0.0312), and shows a better behavior than the rest of comparison methods, although the differences detected by the test are not too significant. In terms of kappa rate, SSMA-PS improves significantly over DROP3 (with a level of significance 0.0312), FCNN (with a level of significance 0.0469) and FCNN (N-S) (with a level of significance 0.0156), and shows a better behavior than DROP3 (N-S) and 1-NN (note that in the comparisons between SSMA-PS and DROP3 (N-S) the number of samples is too small to allow obtaining a higher level of significance).

Table 5 Time elapsed (seconds)

Phase	Training					Test					
Mode	Stratified			Non-stratified		Stratified			Non-stratified		
Algorithms	SSMA-PS	DROP3	FCNN	DROP3	FCNN	SSMA-PS	DROP3	FCNN	DROP3	FCNN	1-NN
Adult	11,107	279	15	6,957	49.6	0.075	1.01	2.92	0.84	0.98	29.8
Census	77,089	11,334	62.7	–	1,943	1.41	69.2	220	52.3	47.5	1,663
Connect-4	21,149	511	46.5	17,699	382	1.24	18.1	42	12.4	5.32	93.1
Fars	57,097	1,159	40.6	20,345	440	1.36	19.9	53.1	5.88	1.98	184
KddCup	119,416	35,371	86.8	–	2,343	4.34	11.3	17.3	12.2	8.75	5,670
PokerHand	730,236	68,450	448	–	5,495	256	994	3,146	732	496	13,937
Shuttle	13,097	1190	0.766	7,654	12.5	0.022	0.087	0.068	0.076	0.039	36

Time in stratified mode is computed as the sum of the time elapsed in every strata

Table 6 Wilcoxon test for classification rate

Methods	R^+	R^-	p value
SSMA-PS vs DROP3	27	1	0.0312
SSMA-PS vs FCNN	25	3	0.0781
SSMA-PS vs DROP3 (N-S)	15	0	0.0625
SSMA-PS vs FCNN (N-S)	27	1	0.0312
SSMA-PS vs 1-NN	25	3	0.0781

Table 7 Wilcoxon test for kappa rate

Methods	R^+	R^-	p value
SSMA-PS vs DROP3	27	1	0.0312
SSMA-PS vs FCNN	26	2	0.0469
SSMA-PS vs DROP3 (N-S)	15	0	0.0625
SSMA-PS vs FCNN (N-S)	28	0	0.0156
SSMA-PS vs 1-NN	25	3	0.0781

5.2.2 *Efficiency*

The first factor that affects the efficiency of the training and test classification phases is the reduction rate achieved with the execution of a PS method. The efficiency of these phases will be greatly increased with higher reduction rates.

As Table 4 shows, SSMA-PS achieves the best results in reduction in all domains. SSMA-PS is able to select the best prototypes to simultaneously increase the objectives defined in its fitness function, without being influenced by local restrictions related to the distribution of data.

From the time elapsed table (Table 5) some interesting conclusions can be drawn:

- In the training phase, FCNN has extremely low time consumption, and SSMA-PS consumes a high amount of time. Although this is the most severe drawback of SSMA-PS, it is important to note that the PS phase only needs to be executed once, in contrast to the test phase. Thus, having a high time cost here may not be a great drawback, as long as the PS method can be executed in a reasonable time. In fact, the employment of stratification allows SSMA-PS to successfully achieve this objective.
- In the test phase, the results are inverted: SSMA-PS is able to greatly increase the speed of classification, in contrast to the rest of the methods. FCNN and DROP3 achieve better reduction rates when stratification is not employed, but they are still less than the ones obtained by SSMA-PS. However, its speed increment is still considerable when compared to the 1-NN itself, with no PS performed.

To further demonstrate the benefits of the achievement of higher reduction rates to speed up the test phase, we have performed an additional analysis: Table 8 shows a comparison of the speed of the 1-NN classifiers resulting from the application of the three PS methods along with the stratification procedure and the 1-NN without PS phase. For each dataset the number of instances that compose its test partition is shown, along with the average number of instances classified per second in the test phase by each method.

Table 8 Average number of instances classified per second in test phase

	# Instances	SSMA-PS	DROP3	FCNN	1-NN
Adult	4,884	65,207	4,847	1,673	164
Census	29,928	21,240	433	136	18
Connect-4	6,756	5,441	373	161	73
Fars	10,097	7,432	507	190	55
KddCup	49,402	11,392	4,391	2,848	9
PokerHand	102,501	400	103	33	7
Shuttle	5,800	268,519	66,438	85,672	161
Average	–	54,234	11,013	12,959	70

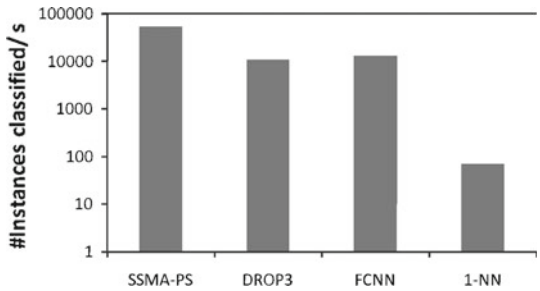

Fig. 3 Average number of instances classified per second. The three PS methods improves greatly the classification speed of the 1-NN rule

The importance of achieving a good reduction rate to improve the classification speed of the 1-NN rule is shown by the stated results: A data set preprocessed with SSMA-PS will offer classification times five times lower (on average; this difference ranges from three or four times in kddcup to 50 or 150 times in census, compared to DROP3 and FCNN, respectively) than when preprocessed with DROP3 and FCNN. Furthermore, this difference becomes even greater when we compare these ratios with the ones achieved by the 1-NN itself: DROP3 and FCNN are 150–180 times quicker than 1-NN, where SSMA-PS is nearly 800 times faster than 1-NN, on average. These differences are depicted in Fig. 3.

This great gain in classification times leads to some important conclusions:

- Firstly, it confirms the utility of PS techniques, because they are able to greatly speed up the 1-NN classifier without losing too much (or even increasing) prediction accuracy.

– And secondly, it states the superiority of SSMA-PS over the rest of the PS method compared due to its greater speed up achievement.

As we explained before, it is more important to obtain better performance in classification phases than in the PS phase, due to the fact that they will be executed many times in a real-life environment. Thus, the obtaining of a faster classifier would be a great improvement for most real-life applications, especially those where the time consumption in the classification phase is critical.

5.3 Analysis of the complexity/runtime trade-off according to the strata size

One interesting question that arises when using stratification to improve the results of a PS method is to select a suitable size of strata. Indeed, this selection will have a strong influence on the behavior of the algorithm, especially in the runtime of the method, as we discussed in Sect. 4. Therefore, it is interesting to analyze the runtime of SSMA-PS scales as the complexity of the search space (i.e. the size of the strata) increases.

Two data sets have been selected to perform this analysis: a low dimensional data set (Shuttle, 9 attributes) and a high dimensional one (Connect-4, 42 attributes). We have collected the average time taken by SSMA-PS when processing these data sets with different strata sizes, ranging from, approximately, 1,000 instances (58 strata in Shuttle, 68 strata in Connect-4) to 15,000 instances (4 strata in Shuttle, 5 strata in Connect-4). Figure 4 shows the runtimes obtained (seconds) in a complete SSMA-PS training phase (i.e. the sum of time elapsed by every strata). Each point correspond to the average of 10 runs (for each one the graphic also depicts the standard error with bars).

As the graphic shows, there is a great increase in the runtime when the complexity of the search space increases too much, both in the low dimensional and in the high dimensional problems. This increment is almost quadratic, thus confirming the theoretical analysis given in Sect. 3.

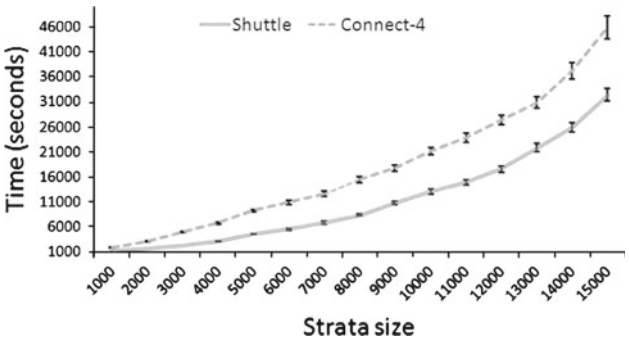

Fig. 4 Time consumption for each strata size for SSMA-PS

In summary, this analysis shows that when employing stratification with SSMA-PS, the runtime obtained will be less if a small size is selected for each strata. Consequently, a general recommendation can be given by stating that a suitable strata size will be the smallest possible, as long as the accuracy related to the resulting training subset does not become reduced too much.

5.4 Effects of the strata size in the accuracy of SSMA-PS

The second factor which is influenced by the concrete size of strata selected is the accuracy obtained by the 1-NN classifier using the final selected subset (both in training and test phases). Depending on the size selected, each isolated execution of the PS method will cover a greater or smaller fraction of the initial training set, thus producing a slightly different behavior with each configuration.

In this experiment, we tried three representative sizes of strata: 1,000, 5,000 and 10,000 instances. For each size, Table 9 shows the mean and standard error of accuracy obtained, both in training and test phases, in each domain of the experimental study. The best results in the test phase for each domain are stressed in bold.

Table 9 Accuracy obtained by SSMA-PS with several strata sizes

Data set	Training			Test		
	Size 10000	Size 5000	Size 1000	Size 10000	Size 5000	Size 1000
Adult	83.34 ± 0.07	82.91 ± 0.07	83.47 ± 0.08	**82.77** ± 0.41	82.68 ± 0.39	82.19 ± 0.41
Census	94.27 ± 0.03	94.19 ± 0.04	93.80 ± 0.04	**94.28** ± 0.11	94.20 ± 0.13	93.89 ± 0.13
Connect-4	68.23 ± 0.21	67.02 ± 0.18	64.14 ± 0.21	**67.52** ± 0.44	66.66 ± 0.46	65.22 ± 0.38
Fars	76.06 ± 0.14	76.24 ± 0.16	75.86 ± 0.15	75.73 ± 0.32	**75.97** ± 0.35	74.76 ± 0.36
KddCup	99.78 ± 0.02	99.68 ± 0.02	99.45 ± 0.02	**99.77** ± 0.03	99.68 ± 0.03	99.37 ± 0.04
PokerHand	52.45 ± 0.04	52.55 ± 0.04	52.66 ± 0.03	52.28 ± 0.13	**52.39** ± 0.13	51.98 ± 0.12
Shuttle	99.75 ± 0.04	99.70 ± 0.05	99.58 ± 0.05	**99.75** ± 0.06	99.67 ± 0.07	99.51 ± 0.06
Average	81.98 ± 0.08	81.75 ± 0.08	81.28 ± 0.08	**81.73** ± 0.21	81.61 ± 0.22	80.99 ± 0.21

Fig. 5 Average accuracy of SSMA-PS with several strata sizes. The best behavior appears when using a strata size of 10000 instances

The results obtained are better as the size of strata increases: The best results in most domains are obtained with a size of 10,000 instances, and the average result is better with this size. Figure 5 depicts these results.

Therefore, a suitable configuration to employ the stratification with SSMA-PS over these domains is to select a strata size of 10,000 instances. This value offers good results in accuracy (i.e. the best possible reduced subsets from the training sets), keeping a reasonable runtime in the training phase of its execution. This is the reason why this size was selected in the main experimental study.

5.5 Effects of stratification in convergence capability

As we stated in Sect. 2.3, one of the main drawbacks of the *scaling up* problem is the lack of convergence of EPS methods when the size of the chromosome is too high. To test this behavior in PS procedures, we have compared the processing of the smallest data set of this study, Adult (which has roughly 50,000 instances), employing SSMA-PS with stratification and SSMA-PS alone.

We selected a random partition of data (90% training-10% test), and applied it to SSMA-PS in the following modes:

- Employing stratification: Data was split into five strata. We gave SSMA-PS 10000 evaluations for each.
- Employing SSMA-PS alone: Data was not split. We gave SSMA-PS the total number of evaluations consumed by the first mode: 50000 evaluations.

Each configuration was carried up 10 times (for the stratification mode we only studied the behavior of the first strata).

Table 10 summarizes the results of the study. For each configuration we show the average point in which we detected the convergence of the algorithm (i.e. the number of fitness function evaluations spent when the last improvement was detected), and the percentage of evaluations spent at that point.

Moreover, Fig. 6 shows a representative example of the evolutionary process graphically. The vertical axis represents the fitness values of the best chromosome for each mode, while the horizontal axis represents the percentage of evaluations performed.

Table 10 Results of the convergence analysis

Mode	Evaluations	Percentage (%)
Stratified	$6{,}487 \pm 185$	64.87
Non-stratified	$48{,}545 \pm 1{,}188$	97.09

For each mode the table shows the average point in which convergence is achieved, and the percentage of evaluations spent so far

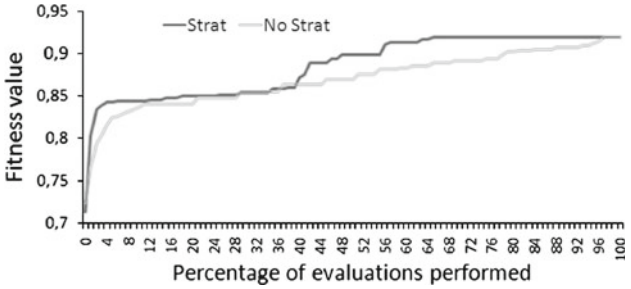

Fig. 6 Comparison of convergence capabilities of the stratified and the non stratified version of SSMA-PS

As the table and the graphic show, although both modes are able to converge, the employment of stratification allows SSMA-PS to converge faster, resulting in more stable behavior when compared to the execution when stratification is not employed. Therefore, we can state that the use of stratification improves effectively the convergence capabilities of SSMA-PS, due to the reduction of the search space obtained by its application.

5.6 Accuracy and efficiency trade-off

The last part of our analysis is devoted to test the compromise between accuracy and efficiency achieved by the PS methods of this study. The best way to perform it is to select the same performance measures employed to obtain this balance in the fitness function of EAs for PS [10], classification and reduction rates.

Figure 7 shows, for each domain, a representation of an opposition between the two objectives in test. Each algorithm located inside the graphics gets its position from the average value of each measure evaluated (exact position corresponding to the beginning of the name of the algorithm). Across the graphic, there is a line that represents the threshold of test accuracy achieved by the 1-NN algorithm without preprocessing.

As we can observe, only SSMA-PS is above the 1-NN horizontal line in most domains (except in *KddCup* and *Shuttle*). FCNN and DROP3 fall under the 1-NN line in every domain (except DROP3 in *PokerHand*). The graphics also clearly emphasizes that SSMA-PS always get the best reduction rates, which locates it always at the right part.

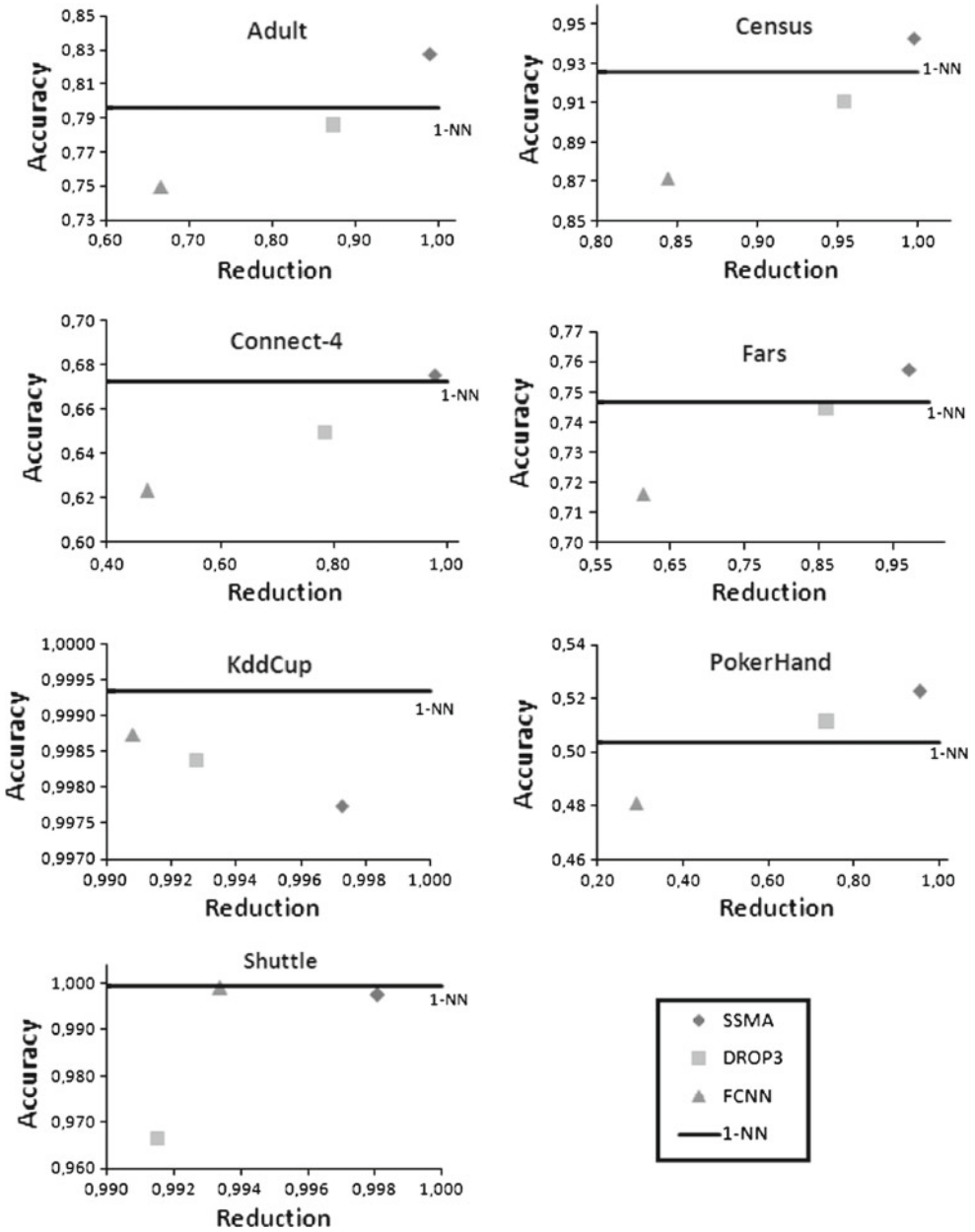

Fig. 7 Graphical comparison of the accuracy-reduction tradeoff in every dataset. The *horizontal* and *vertical axis* represent reduction and classification rates respectively. 1-NN accuracy is represented as a *horizontal line* across the graphics

The balance between accuracy and efficiency achieved by SSMA-PS in these domains fits perfectly with the requirements of a suitable PS method. It gets the best reduction rates possible without harming (or even increasing) the accuracy rate of the baseline 1-NN classifier.

5.7 Future work

Although the application of the stratification procedure has been discussed thoroughly, pointing out many issues about how it modifies the behavior of PS methods, particulary SSMA-PS, there are still some uncovered trend of work which may be taken into account in the future.

A deeper insight into the implications of the use of stratification with PS methods may be gained by studying in which way the splitting of the training set affects the final subsets obtained. For example, PS methods that focus their efforts on selecting the boundary points may be harmed due to the modification of the decision boundaries which may appear when using stratification. Also, competence enhancing PS methods may be affected by this fragmentation of the training set, making it harder for them to search and identify noisy points in the whole data set.

Another trend of future work could be to characterize the usefulness of the final selected subsets with respect to their final employment. It would be interesting to test if the final training sets adequately represent the knowledge contained in the initial sets, and if this knowledge is employed by the classifier to improve its performance. The work presented in [9] should be a suitable starting point for this study.

6 Conclusions

The stratification strategy is a very useful technique to allow PS methods to handle large data sets. In this experimental study, PS methods have been able to tackle domains from 50,000 to more than 1 million instances with a reasonable time cost, thanks to the use of stratification.

SSMA-PS has offered the best behavior over these large problems in the experimental study. It has obtained the best reduction rates in every domain, which has allowed it to offer the best reduced training sets for the 1-NN classifier, both in execution time and accuracy. It allows to reduce the size of the training sets without harming their inherent accuracy.

The main drawback found for SSMA-PS is their large time consumption in the PS phase. However, this is compensated for by its ability to generate smaller training sets than the rest of proposals, leading it to obtain lower execution times in classification phases, where the time elapsed is critical, especially in many real life applications.

Acknowledgments This work was supported by TIN2008-06681-C06-01. J. Derrac holds a research scholarship from the University of Granada.

Appendix A: Description of the PS methods employed in the experimental study

This appendix describes the main characteristics of the PS methods employed as a comparison in the experimental study.

A.1 DROP3

DROP3 was the third of a family of decremental PS methods proposed in [60].

In order to present this reduction technique, we need to define some concepts. In DROP methods, each instance p has k nearest neighbors where k is typically a small odd integer. p also has a nearest enemy, which is the nearest instance with a different output class. Those instances that have p as one of their k nearest neighbors are called associates of p.

DROP3 can be divided into two parts:

– *Noise filtering phase* The first step of DROP3 consists of applying a noise filter to the training set. Wilson's editing (Edited Nearest Neighbor, ENN) is applied to remove as many noisy instances as possible. To perform this process, ENN checks for each instance in the training set if its class agrees with the majority of the class of its k nearest neighbors. Otherwise, the instance is removed.
– *Instance removal phase* The second phase of DROP3 performs an ordered removal of the instances which remain in the training set. Firstly, the instances are ordered by the distance to their nearest enemy. Then, they are checked for removal beginning at the instance furthest from their nearest enemy.
An instance is removed if at least as many of its associates in the original training set (thus including those which were discarded in the noise filtering phase) would be classified correctly without it.
To perform that process, each instance in the original training set maintains a list of its $k + 1$ nearest neighbors in the currently selected subset, even after the instance is removed. This means that instances in the currently selected subset will have associates that will be both selected and not, while instances that have been removed will not have associates.

Due to its suitable behavior in most classical problems, DROP3 has become a widely employed PS method in the literature, obtaining a great success in most of its applications.

A.2 FCNN

The FCNN rule [3], is a condensing PS method devised to be applied to huge collections of data. It is order independent, with a time complexity of $O(N^2)$, with a main objective of selecting points very close to the decision boundaries.

As their authors stated, it is expected to be three orders of magnitude faster than hybrid instance-based learning algorithms over larger problems.

The FCNN algorithm starts by selecting as prototypes the centroids of each class. Then, for each prototype selected, its nearest enemy (the definition of enemy is the same as the one given for DROP3) inside its Voronoi region is found, and is added to the selected subset. This process is performed iteratively until no enemies are found in a single iteration.

Its author proposed some variations to its method. They were:

– Adding the centroid of the enemies found in the Voronoi region at each step, instead of just adding the nearest enemy.
– Adding only one prototype per region for each iteration (the one which belongs to the Voronoi region with most enemies). Also, the selected subset is initialized only with the centroid of the most populated class.

– Adding only one prototype per region in each iteration (the centroid of the Voronoi region with most enemies). Also, the selected subset is initialized only with the centroid of the most populated class.

In their experimental study, the authors found some differences in the behavior of FCNN depending on the concrete variation employed. However, because no variation is shown to be the best, we have employed the basic version of FCNN in this paper.

References

1. Abraham A, Grosan C, Ramos V (eds) (2006) Swarm intelligence in data mining. Studies in computational intelligence, vol 34. Springer, New York
2. Alpaydin E (2004) Introduction to machine learning. The MIT Press, New York
3. Angiulli F (2007) Fast nearest neighbor condensation for large data sets classification. IEEE Trans Knowl Data Eng 19(11):1450–1464
4. Aranha C, Iba H (2009) The memetic tree-based genetic algorithm and its application to portfolio optimization. Memetic Comp 1(2):139–151
5. Asuncion A, Newman DJ (2007) UCI repository of machine learning databases. http://www.ics.uci.edu/~mlearn/MLRepository.html
6. Bacardit J, Burke E, Krasnogor N (2009) Improving the scalability of rule-based evolutionary learning. Memetic Comp 1(1):55–67
7. Baluja S (1994) Population-based incremental learning: a method for integrating genetic search based function optimization and competitive learning. Tech Rep CMU-CS-94-163. Computer Science Department, Pittsburgh, PA
8. Bezdek J, Kuncheva L (2001) Nearest prototype classifier designs: an experimental study. Int J Intell Syst 16(12):1445–1473
9. Cano J, Lozano FHM (2007) Evolutionary stratified training set selection for extracting classification rules with trade-off precision-interpretability. Data Knowl Eng 60:90–100
10. Cano JR, Herrera F, Lozano M (2003) Using evolutionary algorithms as instance selection for data reduction in KDD: an experimental study. IEEE Trans Evol Comput 7(6):561–575
11. Cano JR, Herrera F, Lozano M (2005) Stratification for scaling up evolutionary prototype selection. Pattern Recognit Lett 26(7):953–963
12. Cohen J (1960) A coefficient of agreement for nominal scales. Educ Psychol Meas 20(1):37–46
13. Cover TM, Hart PE (1967) Nearest neighbor pattern classification. IEEE Trans Inf Theory 13(1):21–27
14. Cunningham P (2008) A taxonomy of similarity mechanisms for case-based reasoning. IEEE Trans Knowl Data Eng 21(11):1532–1543
15. Demšar J (2006) Statistical comparisons of classifiers over multiple data sets. J Mach Learn Res 7:1–30
16. Eiben AE, Smith JE (2003) Introduction to evolutionary computing. Natural computing. Springer-Verlag, New York
17. Eshelman LJ (1991) The CHC adaptative search algorithm: how to have safe search when engaging in nontraditional genetic recombination. In: Rawlins GJE (ed) Foundations of genetic algorithms. Morgan Kaufmann, San Mateo, pp 265–283
18. Fischer T, Bauer K, Merz P, Bauer K (2009) Solving the routing and wavelength assignment problem with a multilevel distributed memetic algorithm. Memetic Comp 1(2):101–123
19. Forrest S, Mitchell M (1993) What makes a problem hard for a genetic algorithm? some anomalous results and their explanation. Mach Learn 13(2-3):285–319
20. Freitas AA (2002) Data mining and knowledge discovery with evolutionary algorithms. Springer-Verlag, New York
21. García S, Herrera F (2009) A study of statistical techniques and performance measures for genetics-based machine learning: accuracy and interpretability. Soft Comput 13(10):959–977
22. García S, Cano JR, Herrera F (2008) A memetic algorithm for evolutionary prototype selection: a scaling up approach. Pattern Recognit 41(8):2693–2709
23. Ghosh A, Jain LC (eds) (2005) Evolutionary computation in data mining. Springer-Verlag, New York
24. Gil-Pita R, Yao X (2008) Evolving edited k-nearest neighbor classifiers. Int J Neural Syst 18(6):459–467
25. Guyon I, Gunn S, Nikravesh M, Zadeh LA (eds) (2006) Feature extraction: foundations and applications. Springer, New York
26. de Haro-García A, García-Pedrajas N (2009) A divide-and-conquer recursive approach for scaling up instance selection algorithms. Data Min Knowl Discov 18(3):392–418
27. Hart PE (1968) The condensed nearest neighbor rule. IEEE Trans Inf Theory 14(3):515–516
28. Hart WE (1994) Adaptive global optimization with local search. PhD thesis, University of California, San Diego
29. Hart WE, Krasnogor N, Smith JE (eds) (2005) Recent advances in memetic algorithms. Springer-Verlag, New York
30. Hasan SMK, Sarker R, Essam D, Cornforth D (2009) Memetic algorithms for solving job-shop scheduling problems. Memetic Comp 1(1):69–83
31. Ho SY, Liu CC, Liu S (2002) Design of an optimal nearest neighbor classifier using an intelligent genetic algorithm. Pattern Recognit Lett 23(13):1495–1503
32. Hore P, Hall LO, Goldgof DB (2009) A scalable framework for cluster ensembles. Pattern Recognit 42(5):676–688
33. Ishibuchi H, Nakashima T (1998) Evolution of reference sets in nearest neighbor classification. In: Second Asia-Pacific conference on simulated evolution and learning on simulated evolution and learning (SEAL'98). Lecture notes in computer science, vol 1585, pp 82–89
34. Jankowski N, Grochowski M (2004) Comparison of instances selection algorithms I. Algorithms survey. In: Rutkowski L (ed) International conference on artificial intelligence and soft computing (IAISC'04). LNAI, vol 3070, pp 598–603
35. Kim SW, Oommen BJ (2003) On using prototype reduction schemes to optimize dissimilarity-based classification. Pattern Anal Appl 6(3):232–244
36. Krasnogor N, Smith J (2005) A tutorial for competent memetic algorithms: model, taxonomy, and design issues. IEEE Trans Evol Comput 9(5):474–488
37. Kuncheva LI (1995) Editing for the k-nearest neighbors rule by a genetic algorithm. Pattern Recognit Lett 16:809–814
38. Kuncheva LI, Bezdek JC (1998) Nearest prototype classification: clustering, genetic algorithms, or random search? IEEE Trans Syst Man Cybernet C 28(1):160–164
39. Lim TS, Loh WY, Shih YS (2000) A comparison of prediction accuracy, complexity, and training time of thirty-three old and new classification algorithms. Mach Learn 40(3):203–228
40. Liu H, Motoda H (eds) (2001) Instance selection and construction for data mining. The Springer international series in engineering and computer science. Springer, New York
41. Liu H, Motoda H (2002) On issues of instance selection. Data Min Knowl Discov 6(2):115–130
42. Liu H, Motoda H (eds) (2007) Computational methods of feature selection. CRC Data mining and knowledge discovery series. Chapman & Hall, London

43. Liu H, Hussain F, Tan CL, Dash M (2002) Discretization: an enabling technique. Data Min Knowl Discov 6(4):393–423
44. Liu Z, Elhanany I (2008) A fast and scalable recurrent neural network based on stochastic meta descent. IEEE Trans Neural Netw 19(9):1652–1658
45. Lozano M, Herrera F, Krasnogor N, Molina D (2004) Real-coded memetic algorithms with crossover hill-climbing. Evol Comput 12(3):273–302
46. Lozano M, Sotoca JM, Sánchez JS, Pla F, Pekalska E, Duin RPW (2006) Experimental study on prototype optimisation algorithms for prototype-based classification in vector spaces. Pattern Recognit 39(10):1827–1838
47. Marchiori E (2008) Hit miss networks with applications to instance selection. J Mach Learn Res 9:997–1017
48. Moscato P (1989) On evolution, search, optimization, genetic algorithms and martial arts: towards memetic algorithms. Tech Rep C3P 826, California Institute of Technology, Pasadena
49. Olvera-López JA, Carrasco-Ochoa JA, Martínez-Trinidad JF (2009) A new fast prototype selection method based on clustering. Pattern Anal Appl (in press). doi:10.1007/s10044-008-0142-x
50. Ong YS, Krasnogor N, Ishibuchi H (2007) Special issue on memetic algorithms. IEEE Trans Syst Man Cybernet B 37(1):2–5
51. Papadopoulos AN, Manolopoulos Y (2004) Nearest neighbor search: a database perspective. Springer-Verlag Telos, Santa Clara
52. Provost FJ, Kolluri V (1999) A survey of methods for scaling up inductive algorithms. Data Min Knowl Discov 3(2):131–169
53. Pyle D (1999) Data preparation for data mining. The Morgan Kaufmann series in data management systems. Morgan Kaufmann, San Fransisco
54. Ritter G, Woodruff H, Lowry S, Isenhour T (1975) An algorithm for a selective nearest neighbor decision rule. IEEE Trans Inf Theory 21(6):665–669
55. Shakhnarovich G, Darrell T, Indyk P (eds) (2006) Nearest-neighbor methods in learning and vision: theory and practice. The MIT Press, New York
56. Sheskin DJ (2007) Handbook of parametric and nonparametric statistical procedures, 4th edn. Chapman & Hall/CRC, London
57. Sierra B, Lazkano E, Inza I, Merino M, Larrañaga P, Quiroga J (2001) Prototype selection and feature subset selection by estimation of distribution algorithms. A case study in the survival of cirrhotic patients treated with tips. In: Proceedings of the 8th conference on AI in medicine in Europe (AIME'01). Springer-Verlag, London, pp 20–29
58. Wilcoxon F (1945) Individual comparisons by ranking methods. Biom Bull 1(6):80–83
59. Wilson DL (1972) Asymptotic properties of nearest neighbor rules using edited data. IEEE Trans Syst Man Cybernet 2(3):408–421
60. Wilson DR, Martinez TR (2000) Reduction techniques for instance-based learning algorithms. Mach Learn 38(3):257–286
61. Witten IH, Frank E (2005) Data mining: practical machine learning tools and techniques. Morgan Kaufmann series in data management systems, 2nd edn. Morgan Kaufmann, San Fransisco
62. Zar JH (2009) Biostatistical analysis, 5th edn. Prentice Hall, Upper Saddle River