Intelligent Data Analysis 19 (2015) 631–658
DOI 10.3233/IDA-150736
IOS Press

A fast and flexible instance selection algorithm adapted to non-trivial database sizes

Frédéric Ros[a,*], Rachid Harba[a], Marco Pintore[b] and Serge Guillaume[c]
[a]*Orleans University, Laboratoire Prisme Orléans, Orléans, France*
[b]*PILA, Saint Jean de la Ruelle, France*
[c]*IRSTEA, UMR ITAP, Montpellier, France*

Abstract. In this paper, a new instance selection algorithm is proposed in the context of classification to manage non-trivial database sizes. The algorithm is hybrid and runs with only a few parameters that directly control the balance between the three objectives of classification, i.e. errors, storage requirements and runtime. It comprises different mechanisms involving neighborhood and stratification algorithms that specifically speed up the runtime without significantly degrading efficiency. Instead of applying an IS (Instance Selection) algorithm to the whole database, IS is applied to strata deriving from the regions, each region representing a set of patterns selected from the original training set. The application of IS is conditioned by the purity of each region (i.e. the extent to which different categories of patterns are mixed in the region) and the stratification strategy is adapted to the region components. For each region, the number of delivered instances is firstly limited via the use of an iterative process that takes into account the boundary complexity, and secondly optimized by removing the superfluous ones. The sets of instances determined from all the regions are put together to provide an intermediate instance set that undergoes a dedicated filtering process to deliver the final set. Experiments performed with various synthetic and real data sets demonstrate the advantages of the proposed approach.

Keywords: Supervised classification, instance selection, clustering algorithm, k-nearest neighbors

1. Introduction

Supervised pattern recognition, or classification, is concerned with the design of decision rules whereby entities, described by feature vectors called patterns, are assigned to predefined categories. Among the numerous techniques available, neighborhood techniques using k nearest neighbor (kNN) algorithms remain very popular. Compared to other well-known classifiers, these techniques remain very attractive thanks to their ease of use and are widely applied in industrial applications [11,42]. Given a database S with training sample patterns, whose class label is known, the principle of the algorithm consists in finding the k nearest neighbors of a pattern in order to classify and assign the unknown pattern a label that takes into account the labels of its neighbors. In practice, not all the information in S is useful for the classifier and working with the whole data set presents some drawbacks (storage, learning rate, accuracy, etc.) that have been extensively discussed in the literature [1,14,24,34,45,51]. It is therefore

[*]Corresponding author: Frédéric Ros, Orleans University, Laboratoire Prisme Orléans, Orléans, France. E-mail: frederic.ros@univ-orleans.fr.

convenient to discard irrelevant and superfluous instances (or prototypes) while keeping the most critical ones. This process known as instance selection extends the concept of nearest neighbor algorithms. Managed by so-called condensation and/or edition approaches, it is considered as one of the most important data reduction approaches (like feature selection) involved in the data mining process [47]. The objective consists in choosing a small region S_z of the available data such that the classification accuracy of C_{1nn} (a nearest classifier) over S is maximal. In this way, it is possible to obtain a tractable amount of data that is usable by practitioners engaged in predictive modeling and in the discovery of interesting knowledge from large databases. In predictive modeling, the selection process (instance and feature selection) may improve the computational time needed to induce the classification model and also the quality of the models.

Lastly, the ideal classifier has to be fast, relevant and interpretable but also tractable even with large databases. The latter criterion is of prime concern when dealing with large databases, since finding a pattern neighborhood requires many distance computations. Many methods generally based on neighborhood concepts have been proposed by different scientific communities and recent surveys of the available selection methods can be found in the literature [22,23,30,32].

In this paper, we are interested in an instance selection approach to manage non-trivial databases. By non-trivial databases we mean databases containing more than one thousand patterns to several hundreds of thousands of patterns. Huge databases, i.e. dealing with hundreds of millions of instances (or more), can also be managed but are not addressed in this paper. They require more parallel techniques and specific storage means to be computationally acceptable. We then propose a scaling approach focusing on the runtime or tractability while respecting accuracy and interpretability. The aim is twofold: to use this process in mining databases where the practitioner needs to understand the data and/or include them as a brick in a more general selection framework under the constraint of being fully automatic. For the first problem, interpretability (instance number) and execution time are the primary objectives and classification accuracy is "secondary". For the second, execution time and classification accuracy are the priorities and instance number is less important. The common objective is to limit the number of input parameters for the user and to embed a given flexibility in the algorithm. This flexibility makes it possible to produce relevant outputs whatever the complexity of the classification problem (overlapping, noise, complex borders…), as input parameters are often application dependent and play an important role in determining the quality of the solution.

Our three-step process is based on an existing IS algorithm that has proved to be efficient when applied on small databases [53]. The first step consists in segmenting the database in regions via the feature space. For this, pattern references of each class are separately determined on the basis of a very quick procedure and a fuzzy 1_{nn} procedure [29] is applied to obtain subsets of patterns for each region. In the second step, we apply a specific stratification algorithm to the non-pure created regions instead of the entire database, thus limiting the accuracy degradation. The strata are specifically designed by considering the class distribution in the regions, cases of minority classes are handled and only the necessary strata are kept. We propose an iterative process including a filtering step that stops when additional instances delivered by a new stratum do not afford any additional information to the existing ones. To overcome the possible redundancy due to the stratification process, however, we optionally optimize this set via a genetic algorithm where its exploration space is highly reduced.

In the final step, instances are classically accumulated to define an intermediate set that is finally reduced via a filtering process that discards superfluous pure regions.

The paper is organized as follows. Section 2 presents related work and our contribution. Section 3 describes our three-step approach for selecting instances and Section 4 presents experimental results. Finally, Section 5 gives some concluding remarks.

2. Related work and contribution

Our work concerns the research challenges of the data mining community that needs to develop fast and efficient processes for handling large databases [25].

2.1. Data mining context

The generic goal is to develop both reliable and understandable models for experts to gain insight into complex real-world systems. Data computing can simulate a real-world system reliably and precisely, whereas experts contribute their ability to make high-level semantic generalizations. Preliminary approaches have led to several generations of techniques aiming at hybridizing model accuracy and interpretability. Reviews of these approaches can be found in [20,21] and a good overview of interpretability-oriented fuzzy inference systems is given in [26].

Today, it is clearly recognized that interaction between experts and data computing contributes to generating more efficient models. When dealing with large volumes of data, the remaining challenging problem is how to generate a model that shows not only good global performance but also good interpretability, as these two objectives conflict in this context. There is no general scheme to handle this problem but some certainties can be underlined. Interaction is more fruitful if there is a common "space" where data and expertise can really interact. As it is difficult to manage the interaction in a single step, it is necessary to progress sequentially: firstly, the database is reduced through data computing in order to keep only interesting information; in a second step, the expert guides a new database reduction to facilitate the generation of accurate and understandable predictive models. This process can be repeated until simple models result, with the final aim of understanding the mechanisms governing the information contained in the databases.

In the context of supervised classification, producing interesting knowledge involves two complementary objectives. The first is understanding which feature or set of features plays a role in the class identity. This consists in finding reduced subsets among the original features, in order to include useful class discriminatory information and remove redundant data and/or noise. Unlike feature transform techniques, the fewer dimensions obtained by feature selection facilitate data mining and generally lead to higher classification accuracy.

Secondly, there is the need to determine which patterns are interesting for both pattern recognition and interpretability. This is related to instance selection techniques able to address the necessity to find reduced sets of key patterns that are relevant for class discrimination. It aims at generating a minimal consistent set, i.e. a minimal set whose classification accuracy is as close as possible to that obtained using all training instances. The problem of computing cost is similar for feature and instance selections, as both have to be processed faster without losing too much accuracy. It has to be underlined that this paper focuses however only on instance selection algorithms that address dedicated techniques discussed in the following section.

2.2. Instance selection algorithms

Contributions concerning instance selection algorithms are often related to the machine learning and pattern recognition scientific communities.

The DROP [50] family methods seem to be the most popular as many papers reports that they outperform several well-known previously proposed methods. Starting from the original set, these methods consist in removing patterns step by step in an ordered way to obtain the final set. An item is removed if without it its neighbors can be well classified. Other approaches such as evolutionary algorithms have been successfully investigated leading to good compromises between classification accuracy and instance number [44].

The ideal instance selection method has to satisfy the objective of finding the smallest subset S that maintains or even increases the generalization accuracy. While the methodologies and algorithms reported here above are sufficiently mature to handle the majority of small-sized problems, they are unable to find the optimal solution to "non-trivial" size problems in a computationally tractable time, due to the resulting exponential search space. If the learning phase takes too long these algorithms no longer present any interest for non-trivial databases and real applications.

Several family solutions have been proposed to solve the issue of the computational costs encountered for large databases.

The ideal instance selection method has to satisfy the objective of finding the smallest subset S that maintains or even increases the generalization accuracy. While the methodologies and algorithms reported here above are sufficiently mature to handle the majority of small-sized problems, they are unable to find the optimal solution of "non-trivial" size problems in a computationally tractable time, due to the resulting exponential search space. If the learning phase takes too long there is no longer interest of these algorithms for non-trivial databases and real applications.

Several family solutions have been proposed to solve the issue of computational costs encountered for large databases.

- Modifying known algorithms [47]: The idea is to redesign an algorithm so that, while almost maintaining its performance, it can be run efficiently with much larger input data sets. Although efficient, one difficulty resides in finding the balance between the so-called level of algorithm simplification and the resulting performance.
- Random sample [16]: The idea is to take a random sample and do data mining on it. In this case, answers are approximate but may be acceptable in several data mining applications. Random sampling is however known to be problematic to apply, as it is difficult to determine an appropriate sample size since theoretical bounds (Chernoff bounds) are often not applicable. These bounds are independent of the data structure and by nature do not embed flexibility. The result is that many patterns are produced by the existing algorithms, and in any case more than necessary for accomplishing the specific classification task. Readers interested in these discussions can refer to the following studies [17–33].
- Scaling [9,10,28,45]: Several studies have proposed the use of scalability in approaches based on data partitioning. The latter involves breaking the data set into regions, learning from one or more of these regions, and possibly combining the results by cumulating the selected instances. The idea is to find instances in small regions instead of all over the training set, which can greatly improve the runtime. The existing methods differ in the way they divide the data up and recombine the elementary results. They can be roughly distinguished by stratification and clustering family approaches.

In stratification, partitioning is done by splitting the original data set into randomly generated strata. A fine granularity (large strata) in the stratification approach limits data degradation and hence improves the classification accuracy, but affects the runtime. A rough granularity speeds up the process, but can lead to many instances and affects the classification accuracy for non-trivial classification tasks.

In clustering, the division is driven by the search for groups of similar patterns in the feature space. Some authors [6,19,32,37,41,49] have proposed some initial ideas for using clustering for instance selection; after splitting T in n clusters, the selected instances will be the centers of the clusters. More recently, the same idea was investigated in [36] by managing non homogeneous clusters via a dedicated process. The clustering approach divides the instance selection task without degrading the data. When managing large databases, it is time consuming and the efforts required to provide good clusters are not always justifiable; clustering is an unsupervised (generally density oriented) approach but used to address a supervised classification problem. The regions obtained by clustering have to be sufficiently representative of the decision borders to extract the right instances while remaining numerous enough to reduce the runtime. If the clusters are too small, the number of clusters will be large. Then the algorithm will take much longer to converge and will produce many instances which is undesirable. If too big, then the approach is no longer interesting as much of the search space is encompassed. Both of these situations have to be avoided.

2.3. *Our contribution*

As pointed out by the most recent review [22], there is no ideal method that meets all three objectives. RMHC (Real Multi Hill Climbing) and SSMA [45], as representatives of the hybrid family, obtain an excellent tradeoff between reduction and classifier success, RNGE (Relative Neighborhood Graph Editing) [47] achieves the highest accuracy rate within the edition family. HMNEI (Hit Miss Network Edition Iterative) [34] belonging to hybrid methods, is also a good alternative to increase kNN efficacy. Among condensation methods, RNN (Reduced Nearest Neighbor) [15] and FCNN (Fast Condensed Nearest Neighbor) [4] are the best performing techniques. FCNN is one of the fastest pattern selection approaches.

The first contribution of this paper is related to the soft computing aspect of the hybridization scheme. The two concepts of clustering and stratification are embedded in a system with the aim of using them in the best context and reducing their weaknesses. It is quite customary to combine such approaches in a “system” to overcome the problem of data set sizes.

However, the solution described in the article is not only an implemented solution based on simple algorithms that are put together. The novelty is related to the articulation of the hybridization including the mechanisms to optimize the global performances. The result is an algorithm that can manage non-trivial databases within an interesting runtime without too much degrading the other objectives even for complex classification problems.

The second contribution is related to the usability of the algorithm. There is no method that can be perfect for the three objectives listed above and whatever the problem complexity. However, for the non-expert user, the object is to achieve correct global performances but, more importantly, to be able to drive the balance between the three objectives on the basis of very few functional parameters. Most of existing methods are known to perform well for one or two objectives but they do not include the balance flexibility. Their performances are often highly dependent on input parameters. Our hybridization scheme affords an interesting response to this problem as it includes a self-adaptive scheme.

3. The instance selection algorithm

In this section we present our three-step algorithm, illustrated in Figs 1–4, and demonstrate its relative reliability.

3.1. *Region creation*

The idea is to divide the training set into regions in order to find instances in small regions instead

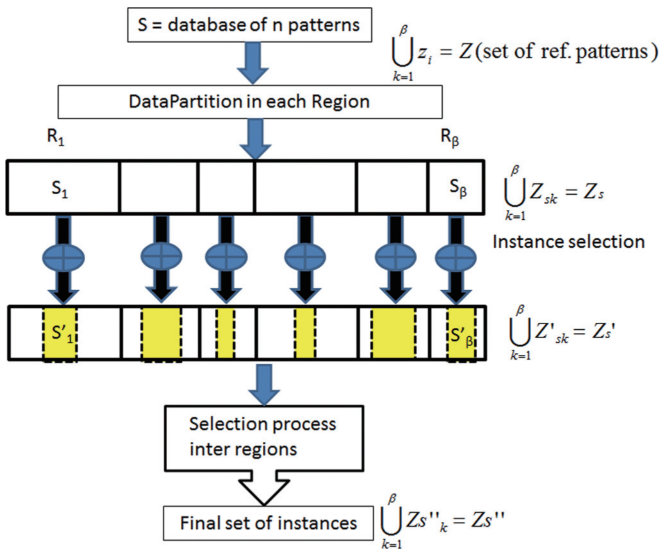

S = database of n patterns
$\bigcup_{k=1}^{\beta} z_i = Z$(set of ref. patterns)
DataPartition in each Region
R_1
R_β
S_1
S_β
$\bigcup_{k=1}^{\beta} Z_{sk} = Z_s$
Instance selection
S'_1
S'_β
$\bigcup_{k=1}^{\beta} Z'_{sk} = Z_s'$
Selection process inter regions
Final set of instances
$\bigcup_{k=1}^{\beta} Zs''_k = Zs''$

Fig. 1. General scheme. (Colours are visible in the online version of the article; http://dx.doi.org/10.3233/IDA-150736)

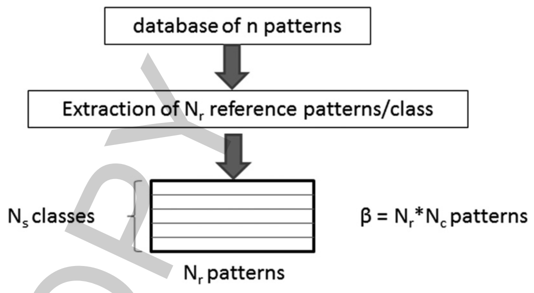

database of n patterns
Extraction of N_r reference patterns/class
N_s classes
$\beta = N_r * N_c$ patterns
N_r patterns

Fig. 2. Recruitment of "influential" patterns.

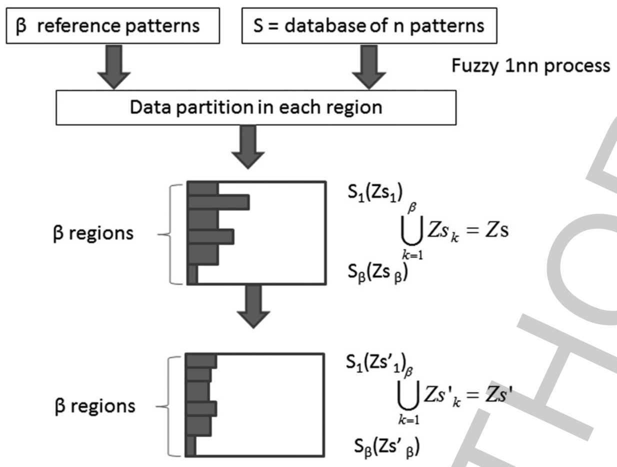


Fig. 3. Instance extraction and reduction in each region.

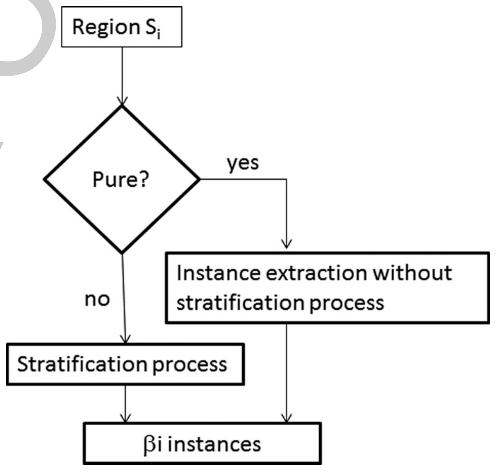


Fig. 4. Decision process for each region.

of finding them over the whole training set, which is very time expensive. We propose two variants to select N_r pattern references in each of the N_c classes independently, and then group them in a so-called target pattern set from which influential patterns are recruited via a fuzzy 1_{nn} procedure. This process is less time-consuming than a generally costly cluster approach and includes the supervised aspect. The question of the number of regions is crucial. There is not a unique and ideal response for any type of data. N_r is defined as an input parameter in order to obtain regions with an average given size S_r. To simplify the reading, classes are assumed to be balanced. If the number of patterns in the classes is n_{cl} then $(N_c * N_r = N_c * (n_{cl}/S_r))$ regions are a priori determined and their number adjusted as a function to the real sizes: too small regions are regrouped and too large ones are divided. Some elementary statistics make it possible to define confidence intervals where N_r can be selected (see Section 3.1.1).

The interesting part of the algorithm is its potential to match with different configurations by the means of the mechanisms introduced. A large number of regions is interesting as likely to simplify the discriminatory aspect in each region and leads to good classification performances. The counterpart is the production of many instances but in our framework they can be reduced by the selection mechanism implemented (see Section 3.3). With small regions, the stratification mechanisms are more important. In this case, the number of instances is more related to the problem complexity. In the case of low complexity (simple boundaries, no overlap between classes), the approach is ideal as quick and relevant. With complex problems, the classification degradation is reduced compared to traditional stratification approaches as it only concerns a region and not the whole data set. The goal is to make the application of an IS algorithm tractable and to be able to use the stratification procedure appropriately.

3.1.1. Determination of reference patterns

This section gives some elementary statistics to define N_r and presents the two alternative approaches to determine reference patterns. The greater the number of reference patterns N_r in each class, the more representative the set of random patterns will be of the probability density f_i of each class.

Some elementary statistics are sufficient to drive this process. Our objective here is simply to capture a reference pattern for each main component of f_i and for a given granularity. Starting from a portion of the probability density p_r, it is easy to estimate the number of extractions necessary to reach this area and reliably assess the possibilities. Let X be the discrete random variable with a sample space Ω that determines the number of random extractions k necessary to reach one pattern from a region R of probability p_r.

One can demonstrate that $\mathit{Prob}(X = k) = p_r * (1 - p_r)^{k-1}$ and therefore deduce that $E(X) = 1/p_r$ and $\mathit{Prob}(X \leqslant k) = 1 - (1 - p_r)^k$.

Proof Let Y be a random variable following a Bernoulli distribution; $Y = 1$ if the random extraction reachs R and $Y = 0$ otherwise. If k random extractions are necessary to reach R with p_r, then R has not been reached during the $k - 1$ previous extractions with $1 - p_r$. In other words, $Y = 0$ during $k - 1$ times and $Y = 1$ at the time k. So,

$$\mathit{Prob}(X = k) = \underbrace{(1 - p_r) * (1 - p_r)}_{k-1} * \ldots p_r = p_r * (1 - p_r)^{k-1}$$

$$E(X) = \sum_{k=1}^{\infty} k * p_r * (1 - p_r)^{k-1} = p_r * \sum_{k=1}^{\infty} k * (1 - p_r)^{k-1}$$

$$p_r < 1 \text{ then } \sum_{k=1}^{\infty} (1 - p_r)^k = 1/(1 - (1 - p_r) = 1/p_r$$

then by deriving term by term

$$\sum_{k=1}^{\infty} k * (1 - p_r)^{k-1} = -\sum_{k=1}^{\infty} k * (-1) * (1 - p_r)^{k-1} = 1/(p_r)^2$$

then

$$E(X) = p_r * \sum_{k=1}^{\infty} k * (1 - p_r)^{k-1} = 1/p_r$$

$$Prob(X \leqslant k) + Prob(X > k) = 1$$

then

$$Prob(X > k) = \sum_{i=k+1}^{\infty} p_r * (1 - p_r)^{i-1} = p_r * \sum_{i=k+1}^{\infty} (1 - p_r)^{i-1}$$

$$Prob(X > k) = p_r * \left[(1-p_r)^k + (1-p_r)^{k+1} + \ldots\right] = p_r * (1-p_r)^k * (1+(1-p_r)+.(1-p_r)^2 + \ldots)$$

$$Prob(X > k) = p_r * (1 - p_r)^k / p_r) = (1 - p_r)^k$$

then

$$Prob(X \leqslant k) = 1 - (1 - p_r)^k$$

If $p_r = 0.2$, five extractions are necessary on average (preferably ten) to reduce the risk of not reaching R.

$$Prob(X \leqslant 10) = 1 - (1 - 0.2)^{10} = 0,9$$

It is then possible to select the number of patterns per class by estimating the number of representative patterns per region of probability p_r.

3.1.1.1 First variant

Basic random algorithm: this method simply consists of a procedure that is applied to each class separately. For each class i, the algorithm is applied so as to choose N_r samples.

3.1.1.2 Second variant

Random selection can produce patterns that are not well distributed, especially if the number of random patterns is limited. As already mentioned, clustering is generally itself costly and not necessary for the study. This variant can be seen as an intermediate between a clustering algorithm and a random selection. For each class, it consists in iteratively selecting a new prototype as the pattern that is the most distant from existing prototypes. Note that in our framework, a selected prototype simply becomes a reference pattern. As for a classic clustering algorithm, each prototype defines a set of attached patterns that are closer to it than the other prototypes. The algorithm is very fast $O(kn)$ as it consists only in calculating at each step i ($n - n_p(i)$) distances, where $n_p(i)$ is the number of prototypes at the step i (k iterations). Each time a new prototype is determined, the distribution of patterns among the prototypes has to be reviewed. For each pattern P, the question is whether it will be attached to the new prototype A or remain attached to the same one, say B. We applied the triangular inequality to avoid the calculation of $d(P, B)$ when the configuration is appropriate, where d stands for the selected distance in the feature space. This means that a maximum of $n - n_p(i)$ distances has to be calculated at each iteration.

Let $T = \{\mathbf{x}_1, \ldots, \mathbf{x}_n\} \in R^p$ be a set of n feature vectors in a p-dimensional space representing the database patterns. The objective is to find $S = \{\mathbf{y}_1, \ldots, \mathbf{y}_m\} \in R^p$ the set of selected prototypes.

The algorithm can be summarized as follows:

Algorithm Select 1 [Input: $T, n, d, m, w_{\min}$; Ouput: instance set S]

```
1:  Select an initial pattern $x_i$ randomly chosen from $T$.
2:  $S = \{\mathbf{y}_1\}$ where $\mathbf{y}_1 = \mathbf{x}_i$
3:  $N_{ref} = 1$
4:  for each $\boldsymbol{x_k} \in \{T - S\}$
5:      Find its nearest neighbor $y_{\boldsymbol{k}}$ $(1nn)$ ins $S$ : $y_{\boldsymbol{k}}(1nn) = \min d(x_k, y_l)$ for $l \in \{1, N_{\text{ref}}\}$
6:      Assign $\boldsymbol{x_k}$ to $y_{\boldsymbol{k}}$ $(1nn)$ and store the related distance
7:      Update $T_k$, the subset of patterns from $T$ attached to $y_{\boldsymbol{k}}$ $(1nn)$
8:  end for
9:  for each $y_k \in S$,
10:     Find its farest object $\mathrm{O}_{kf}/d(y_k, \mathrm{O}_{kf}) = \max d(y_k, x_i)$ for $x_i \in T_k$
11:     Store the distance between $y_k$ and $\mathrm{O}_{kf}$
12:     $d_M(y_k) = d(y_k, \mathrm{O}_{kf})$
13: end for
14: Determine the winner prototype w with the farthest distance: $d_M(y_w) = \max(d_M(y_l))$
    $l \in \{1, N_{\text{ref}}\}$
15: Deduce the winner patterns (new prototype) $x_w = \mathrm{O}_{wf}$
16: $S = S + x_w$
17: $N_{\text{ref}} = N_{\text{ref}} + 1$
18: if $N_{\text{ref}} = m$ then go to 19 otherwise go to 4.
19: Sort the prototype by weight importance and skip the ones representing less than $w_{\min}$
20: end
```

From our experimental results the second variant leads to more stable results than those which rely on random selection, especially when the number of prototypes is small. Furthermore, the algorithm is optimized to reduce its computational cost. As for the first variant, the algorithm is used in order to choose N_r samples for each class; this set constitutes the reference patterns. For example, consider a set of 10000 patterns in a 6 dimensional space and are related to 5 classes of 2000 patterns. If $N_r = 30$, then 30 prototypes are extracted in each of the 5 subsets of 2000 patterns to give 150 prototypes.

3.1.2. Recruitment of influential patterns

This section concerns the recruitment of "influential" patterns from the selected $N_r * N_c$ reference patterns. Our algorithm consists of two steps: in the first one, a preliminary set of regions is determined by applying a fuzzy 1_{nn} procedure on the $N_r * N_c$ reference patterns. In the second step, regions having less than $k_{\min}$ patterns are simply discarded and the associated patterns re-distributed in the β remaining ones ($\beta \leqslant N_r * N_c$).

This set is fuzzy partitioned into areas that identify regions of the feature space by the application of a fuzzy 1_{nn} procedure. A "crisp" (non-fuzzy) 1_{nn} approach may not always be sufficient. Some patterns close to the decision boundaries can be assigned to a region just near the decision boundaries. They will then be ignored by the IS, which can slightly affect the classification accuracy. This is particularly true when the number of clusters is large. To handle this situation, a pattern x_j can be assigned to different regions when the distances are similar.

Let $Z = \{z_1, \ldots z_\beta\}$ be the set of β prototypes representing the c classes, then the fuzzy 1_{nn} procedure is defined as follows:

The distance from z_i to x_j is processed to assign membership in all classes.

$$u_i(x_j) = 1/\left\|x_j - z_i\right\|^{2/l-1} \bigg/ \sum_{k=1}^{c} (1/\left\|x_j - z_k\right\|^{2/l-1}) \tag{1}$$

where l stands for the level of fuzziness in the membership. There is no special need for our problem then l is assigned to 2. The difference with a crisp approach is that membership in each class is based only on the distance from the prototype of the class. This is because the prototypes should naturally be assigned complete membership of the class they represent.

The matrix $U = (\mu_{jk})$ $(j = 1, \ldots, n, k = 1, \ldots, \beta)$ is then introduced to identify the degree of belonging of patterns x_j to region k:

$$\sum_{k=1}^{\beta} \mu_{jk} = 1, \forall k; \quad \mu_{jk} > 0 \text{ and } \mu_{jk} = u_k(x_j) \tag{2}$$

A threshold T_h is introduced to decide whether x_j is assigned to a region k or not. It is assigned if and only if $u_k(x_j)/\max_l u_l(x_j) \geqslant T_h$.

Then, given a data set $S = \{x_1, x_2, \ldots x_n\}$ where x_j is a pattern in a p dimensional feature space, and n is the number of patterns in S, S is the partitioning of S into β regions $\{S_1, S_2, \ldots, S_\beta\}$ and finally satisfies the following conditions:

– Each pattern is assigned to at least one region, i.e.

$$\bigcup_{k=1}^{\beta} S_k = S \tag{3}$$

– Each region has at least $k_{\min}$ patterns assigned to it, i.e.

$$|S_k| \geqslant k_{\min} k = 1, \ldots, \beta \tag{4}$$

3.2. Instance extraction using stratification in each region

This section describes how instances are extracted from each region via the use of stratification. A classical stratification scheme is adopted but our stratification process is applied to the regions and not to the entire database, thus greatly reducing the risk of degradability. There is a critical need to consider the problem of minority classes [40], i.e. the existence of classes that have few examples with regard to other classes. Note that the presence of minority classes in our process can be due to an initial imbalance in class distribution that has been widely studied but also to the configuration of the regions themselves. Our stratification process is adaptively applied by distinguishing the regions according to their discrimination power. Furthermore, the number of strata is optimized. Finally our algorithm is based on the following considerations:

– Region analysis: The nature of the regions for deciding whether or not to apply the IS algorithm and therefore improve the runtime.
– Stratification process: The quality or representativeness of each stratum that deals with the stratum size and the consideration of minority classes.
– Optimization of strata number: The number of strata necessary to limit the number of instances without significantly affecting the classification accuracy.
– Instance extraction

3.2.1. *Region analysis*

We propose to determine the nature of the regions by considering two criteria: the first one is the purity $L_p(i)$ that is simply the extent to which patterns belonging to different categories are mixed in the region. This criterion is naturally informative only when the extent of mixture is very weak.

$$\begin{cases} Lp(i) = 1 \quad \textit{if} \max\limits_{j=1 \textit{ to } \beta}\left(|S_i| \Big/ \sum\limits_{i=1}^{\beta} |S_{ij}| \geqslant T_p\right) \\ Lp(i) = 0 \quad \textit{else} \end{cases} \tag{5}$$

T_p is a threshold fixed between 0.95 and 1. The degree of purity is by essence a scalar but the region is considered to be pure or not pure according to T_p that fixes the amount of tolerated noise or outlier patterns.

When a region is pure, it is not necessary to apply IS; the barycenter of the patterns representing the most representative class can be considered as an instance. It may not be representative of all the members especially when the shape is complex. A set of representative patterns is more effective in certain cases.

To complement the notion of purity, we have introduced the notion of criticism. Non pure regions are critical by essence. We consider that the degree of criticism ($L_c(i)$) is meaningful only if $L_p(i) = 1$. Let a data set $R = \{r_1, r_2, \ldots r_\beta\}$ where r_i is the representative pattern of region i in a p dimensional feature space, and β is the number of regions. In our context, the representative pattern is simply the barycenter of patterns belonging to the most representative class.

A pure region i will be critical if its representative pattern r_i, considered among the set of other representative patterns of Z, is relevant for discriminating boundaries.

$$\begin{cases} Lc(i) = 1 \quad \text{if } z_i \text{ is not relevant} \\ Lc(i) = 0 \quad \text{else} \end{cases} \tag{6}$$

When $L_p = L_c = 1$, the region should not have not a strong influence on the discrimination. The patterns of region i are representative and therefore their barycenter can be considered as an instance. When $L_p = 1$ and $L_c = 0$, it means that a set of representative patterns is preferable to the barycenter. Let us now explain what is meant by relevant for discriminating boundaries and what we do (and when) with these elements.

β regions have been identified, some of them are pure and others not. A given region is relevant for discriminating boundaries when one of its neighbor regions is not pure or pure but representing a different class than itself. The notion of neighboring between regions needs to be defined. The best way to define that a region A is neighbor to B is that there exists a pattern of A (B) that is the nearest neighbor of a pattern of B (A). Considering all the elements is incompatible with the goal of optimizing the runtime. We propose a more approximate but less costly procedure that consists in applying nearest procedures by firstly considering the barycenter of each region and secondly a set of representative patterns. Let there be a region A_0 and A_1 its nearest neighbor according to $\|A_0, A_1\| = d(G(A_0), G(A_1)), G(T)$ being the barycenter of the set T. The set of neighbor regions A_i is defined such that $\|A_0, A_i\| \leqslant 2 * \|A_0, A_1\|$. A set of p representative patterns r_{ij} in each A_i is randomly extracted, p being card (A_i)/m_size where m_size is a fixed parameter, card(T) the number of patterns in T. Each pattern is assigned to the class of its representative region if pure and to an additional class called c^* if non-pure. It is then possible for each z_{0j} to determine whether it represents a critical subset or not then deduce the state of A_0. If A_0 is critical, instead of considering only the barycenter of A as instance, the set of representative patterns r_{0j} serves as instances. By reviewing the granularity, one contributes to keeping the ones close to the boundaries and discarding the others that are not useful. This instance reduction part is detailed in Section 3.3.

3.2.2. Stratification process

This section addresses the main parts of the stratification process: the questions of stratum size, the problem of minority classes, and the number of strata to be handled.

3.2.2.1 Minimum size of each elementary stratum for each class

A stratum construction can be regarded as a simple poll. If f_i is the probability density function of class i, the goal is to be able to capture a trace of the main elements of all f_i. Each f_i can always be seen as the union of elementary distributions or regions, each of them presenting a given probability. The basis of our reasoning is the following: in each stratum, a minimum number of points for each class, say $N_{\min}$, and for a region R of probability p_r, is selected. Let X_j be a random variable that is 1 if the j^{th} point in the sample belongs to R and 0 otherwise. $X_1, \ldots, X_{ni}$ are independent Bernoulli variables so that $p(X_j = 1) = p_r$. n_i is the number of patterns of class i in one stratum. The number of data points belonging to R is $X = X_1 + \ldots + X_{ni}$ that gives $E(X) = n_i * p_r$. For $p_r = 5\%$ and $n_i = 400$, we have $E(X) = 20$. For p_r and $N_{\min}$ fixed, we can deduce that n_i has to be on average greater than $N_{\min}/p_r$. $N_{\min} = 5$ and $p_r = 2\%$ give $n_i = 100$ patterns per class, which appears to be the minimum. It should be noted however that there is no guarantee that the resulting granularity will be sufficient to handle boundary complexity. By considering strata per region and not on the whole data set, the risk of degrading the data is however greatly minimized.

3.2.2.2 Minority class consideration

Minority classes [12] are those that have few examples with respect to other classes inside a database. Based on the literature, the general strategy used to manage this issue consists in designing strata by selecting the minor classes more frequently in order to ensure that the class values are uniformly distributed [8]. This completion process is convenient when the imbalance between classes is due to limitations related to the construction of the database itself. In our case, we assume that any initial imbalance has been corrected and therefore that the presence of minority classes is due to the natural distribution of patterns in each region.

The class imbalance on the subset can be handled through the design of the stratification algorithm. We propose a heterogeneous stratification algorithm, that is to say that each stratum will not necessarily have the same distribution patterns of different classes.

Applying a so-called "homogeneous" stratification algorithm in the case of imbalance between the classes is likely to increase the computational time but especially may significantly degrade the classification accuracy. Seeking to balance the distribution patterns by homogeneous strata, we may give undue importance to patterns related to small minority classes that may statistically be equivalent to noise. Conversely, the strata procedure has to be efficient for non-noisy patterns even if they are few of them.

Each class c_i is characterized by its level of representativeness in the region with that of the densest class. This measure is defined by the ratio between the numbers of patterns (Nps_i and $Nps_{\max}$). For a very large ratio, there is a high probability of having the presence of noise for the current class, and for very low ratios, the opposite. It is therefore possible to achieve a compromise between these two extremes and to construct the strata more consistently. Two coefficients a_1 and a_2 are used to determine d_1 ($0 < d_1 < 1$) (Fig. 5), which stands for the degree of possibility of the class not being noise. a_1 determines the threshold from which the presence of noise is considered as null while a_2 the threshold from which it is considered certain.

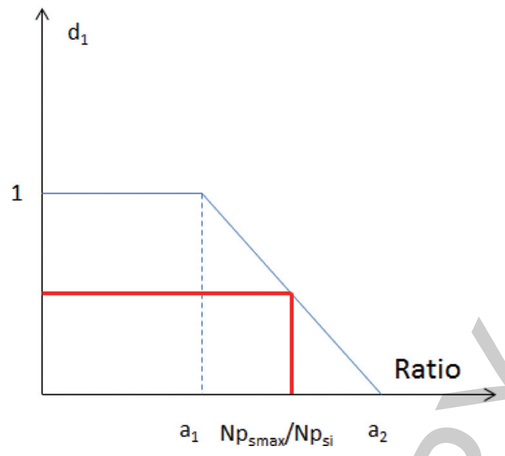


Fig. 5. Membership function to determine the stratification process. (Colours are visible in the online version of the article; http://dx.doi.org/10.3233/IDA-150736)

The number of strata of (N_s) patterns for class i can then be calculated: $S_i = \max(1, S_{\max} * d_1)$, where $S_{\max}$ is the number of strata of the category having the largest number of subgroups. If $(Nps_i < N_s)$, Nps_i patterns are naturally considered for class i. Finally, our algorithm consists in selecting the category having the largest number of subgroups, from which the stratum number $S_{\max}$ is deduced. The available subgroups of the other categories are then sequentially joined. The completion is done for each stratum by randomly selecting a subgroup related to the categories not yet included. The fundamental difference with conventional algorithms resides in the level of completion. Instead of systematically adding each category in the $S_{\max}$ strata and obtaining balanced strata, there is a limitation for each class. For each region i, S_i strata having the same number of patterns (N_s) are then generated. They are completed with $(S_{\max} - S_i)$ small strata having $(N_s) * (Nps_i/Nps_{\max})$ patterns. The strata are ordered by starting with the most complete ones. For example, let a problem with two classes and a region containing 800 and 200 patterns respectively. With $a_1 = 1$, $a_2 = 10$ and $N_s = 100$, one obtains 8 strata, 5 of them contain 200 patterns (100 for each class) and they are completed by two strata having 125 patterns (100 for class 1 and 25 for class 2).

3.2.3. *Optimization of the number of strata*

All non-pure regions are cut into strata but they are not always necessarily useful. The optimal number of strata depends on the data and stratum size. In this phase, we therefore propose to limit the number of strata in an on-line sequential fashion in order to reduce the runtime. Our algorithm obtains instances sequentially stratum by stratum and determines from these instances whether it has already received enough instances to issue the currently best rule as that with high confidence. Strata are ordered so that the more complete are presented first. The number of strata is therefore not fixed *a priori* but is adaptively determined.

Different stopping criteria are used:

– Stop related to the classification level. The new interesting instances are evaluated by a C_{1-nn} procedure applied to the existing ones taken as references. If they are well classified (more than a given score T_σ), this means that the new references do not afford any additional relevant information. If there is no real progression in the classification level between three consecutive steps, it means that the existing set is sufficiently relevant for further processes. In both cases, certain precautions need to be taken to obtain a minimum number of strata.

Let I_{ik} be the set of instance patterns for the stratum number k and the region i. The procedure is simply stopped when I_{ik+1} is well classified by the union of the I_{ik}.

$$Score(I_{ik+1}) = C_{1-nn}\left(\bigcup_{j=1}^{k} I_{ij}\right) \tag{7}$$

– Stop when the amount of noise is too relevant or the data include too complex boundaries; the idea is to identify extreme cases and therefore stop the iterative process.
  For an IS algorithm, two important indicators can be considered, related to the proportion of noise in the data and to the complexity of boundaries, respectively. If these indicators are both strong, the active strata should not deliver any interesting information. To go further, suppose that our IS algorithm is based on a process that delivers a number of instances able to classify all the training data. Each instance is the nearest neighbor of a set of patterns having a size w and belonging to the same category. It is then possible to study the size distribution of the different sets and apply a filtering process. Let s be the size of the data in the active stratum, m the number of instances selected and m_f the number of instances assimilated to noise ($w \leqslant w_f = 1$ for example).

$$\begin{cases} \lambda = \dfrac{m_f}{m_f + m} \\ \mu = \dfrac{m_f + m}{s} \end{cases} \tag{8}$$

  If λ is more than 50% (R_{noise}) for example, this means that the amount of noise is very high as half of the instances are isolated. Similarly, if μ is more than 30% ($R_{complex}$), for example, it means that one third of the data is necessary to classify the database. This denotes a high level of boundary complexity. When the two conditions are present, one can discard this stratum and, if two strata are in this situation, stop the iterative process.
– Stop when a maximum number of iterations $\mathrm{It}_{\max}$ has been exceeded; the limit reveals the size that is maximally tolerated in terms of processing.

It should be underlined that other stopping criteria can added, depending on the final goal. For example, when the search is rather exploratory, there is not the necessity to evaluate all the regions and partial results are sufficient. This concept can be introduced via simple rules. Similarly, when dealing with non-pure regions, the number of studied strata can also be limited.

3.3. *Selection processes and final set extraction*

This section deals with the last step of the framework, when instances have been extracted in each region and the final set has to be generated. Two selection processes are proposed: the first one is intra region and concerns only non-pure regions, whereas the second one is inter regions but is configured to manage superfluous instances coming from pure regions.

3.3.1. *Instance extraction*

S can be seen as the union of βS_i, each S_i divided into $N_{\beta i}$ strata when IS is applied. Instances are extracted from each stratum and joined to form the preliminary instance set Z_s. $N_{\beta i}$ and β_i define the number of strata of the region i and the whole number of instances recruited, respectively.

$$S = \bigcup_{k=1}^{\beta} S_i = \bigcup_{i=1}^{\beta}\bigcup_{j=1}^{N\beta i} S_{ij} \tag{9}$$

$$Z_s = \bigcup_{i=1}^{\beta} Z_{si} = \bigcup_{i=1}^{\beta}\bigcup_{j=1}^{\beta i} z_{ij} \tag{10}$$

β_i depends on the classification complexity in each region and/or the size, for example the number of patterns. If the region is pure then β_i is equal to 1 for pure and non critical regions (see Section 3.2.1). When the region is pure but critical, β_i is directly linked to the size, otherwise it depends on the outputs of IS applied in each stratum.

3.3.2. Instance selection in each region (intra region)

This section considers non-pure regions and aims at reducing the number of instances Z_s. versus Z'_s. The idea is to have parameters calibrated for a large number of problems, the primary objective being to avoid deterioration of the classification performances. Then, the precautions taken in the selection of parameters (especially λ and μ) may result in some redundancy. It is then proposed to reduce the number of instances optionally according to the final goal.

$$Z's \subset Zs = \bigcup_{i=1}^{\beta} Z's_i = \bigcup_{i=1}^{\beta}\bigcup_{j=1}^{\beta' i} z'_{ij} \tag{11}$$

To limit the computational cost, we suggest a decremental process, i.e. only a reduction of instances is carried out. The level of reduction in percentage is limited. This approach avoids the possible damaging effect of removing too many instances while increasing the chances of removing no useful instances. The use of a dedicated genetic algorithm (GA) is suggested but other techniques are also suitable. GAs are not the topic of this paper: what is important here is to limit the exploratory search of the GA via the decremental process and the maximum level of reduction. This option remains costly compared to the other parts of the framework. It should be activated only when storage requirements are critical for the study.

3.3.3. Instance selection inter regions devoted to pure regions

Depending on the complexity of the problem and the level of division into regions, some components of I_s are not useful as instances.

The goal is to obtain a further instance set $Z''s(Z''s \subseteq Z's)$ so that the accuracy degradation is limited. Applying massively a so-called merging algorithm to the entire set of instances is not acceptable as the accuracy would be substantially affected, even for a discovery context. At this step, the origin of the instances is unimportant, only their nature (pure or not).

$Z's$ can be seen as the union of two subsets, the first representing the pure instances $(Z'p_s)$ and the second the non-pure ones $(Z'q_s) : Z's = Z'p_s + Z'q_s$ and the goal is to reduce $Z'p_s$ while disregarding the non-pure ones. The final set $Z''s = Z''p_s + Z'q_s$ where $Z''p_s \subseteq Z'p_s$. The core of the reduction algorithm is the following:

- The components of Ip_s that have as nearest neighbor a pure instance of a different class or a non-pure one are automatically preserved.
- The others are candidates to be discarded and are submitted to the reduction algorithm.
- The algorithm is iterative and consists in examining a candidate instance to be added at each step. The non-selected candidates are considered as useless.

Let $Z' = \{\mathbf{z}'_1, \ldots, \mathbf{z}'_n\} \in R^p$ be a set of n feature vectors in a p-dimensional space, each z_i is assigned to a class among the c available if it represents a pure region otherwise it is assigned to an additional class called c^*. The objective is to find $Z'' = \{\mathbf{z}''_1, \ldots, \mathbf{z}''_m\} \in R^p$ the set of selected instances.

The reduction algorithm is based on the FCNN idea to which some modifications have been made. It can be summarized as follows:

Algorithm Select 2 [Input: Z', Z'_p, Z'_q; Ouput: instance set Z'']

```
1:  initialize $Z''_p = \{Z'_q\}$
2:  **for** each $\boldsymbol{z'_k} \in \{Z'_p\}$
3:     find its one nearest neighbor $z'_{1nn}$ in $Z'$
4:     if Cat $(z'_{1nn}) \neq$ Cat $(\boldsymbol{z'_k}) Z'' = Z'' + \boldsymbol{z'_k}$
5:  **end for**
6:  $N_c = 0$ (number of members misclassified)
7:  for each $\boldsymbol{z} \in \{Z'_p\}$
8:     find its nearest neighbour $z''_{(1nn)}$ in $(Z'' - z) : d(z''_{(1nn)}, z) = min\ d(z, z''_l) z'' \in Z''$
9:     add $\boldsymbol{z}$ as member of $z''_{\boldsymbol{k}(1nn)}$
10:    **if** Cat$(z_{\boldsymbol{k}}(1nn)) \neq$ Cat $(z)$
11:        assign $z$ as misclassified ($N_c = N_c$+ 1) and
12:       update $\mathrm{distz_{1NN}}(\mathrm{distz_{1NN}} = \min(\mathrm{distz_{1NN}}, d(z''_{(1nn)}, z))$
13: **end if**
14: **end for**
15: if $N_c$ equal 0 then **go to** 21
16: **find** $z''_0 \in Z''$ so that $d(z''_0, \mathrm{O}_f) = \min(dz''_{(1nn)})$ with $z'' \in Z''$ and $\mathrm{O}_f$ the nearest misclassified
    member of $z''$.
17: $Z'' = Z'' + \mathrm{O}_f$
18: end
```

4. Experimental results

The objective of this section is to validate the potential of our soft computing approach. The aim is to show that the algorithm can be used without tuning the parameters to match the specificities of a particular database. The algorithm is naturally controlled by a number of parameters. Evaluation on real-life data of such algorithms could reveal that finding proper parameters might be difficult and costly. However, in our framework, the user only needs follow a set of input parameters based on best practices and specify a few extra parameters, without being faced with all the computation complexity.

We then broaden the first trials to measure the algorithm consistency by varying some input parameters on large scale and performing a pre-calibration. The second trials are complementary and we compare the performances of the algorithm with those of other approaches in the same context. According to the literature, the DROP methods [37–49] are among the most successful instance selection approaches; DROP outperforms several well-known wrapper methods and even some filter ones in both classification and reduction. However, DROP methods are rather costly and also need to be well-parameterized. We selected an adapted version of the FCNN approach for our IS algorithm. It is reputed to be among the quickest approaches (its complexity is $o(kn)$, k being the number of iterations), is well adapted to our way of optimizing the number of strata, and is completely consistent with our main objective, i.e.

optimizing the runtime. FCNN is seldom selected as a reference because it does not handle the issue of noise, but this is managed in our framework. It should be also noted that the computational gain of our approach is naturally better with algorithms of $o(n^2)$ complexity.

4.1. Databases

Eighteen popular databases with known difficulties were evaluated: 7 are synthetic and 11 are real. The database characteristics are summarized in Table 1. For each of them, we specify the number of patterns, features and classes. All of the vectors were normalized to be within the range $[-1, 1]$ using their standard deviations and the data set for class j was randomly split into two subsets of equal size. One of them was used for choosing the initial instances and training the classifiers, and the other one was used in their validation.

The features are the original ones. They are reasonable in size (less than 15 dimensions) so as to be suitable for applying k nearest approaches. This is done simply to avoid well-known limitations due the dimensionality of data. It does not mean that the proposed algorithm cannot run with high dimensional databases. However, neighborhood classification approaches perform better with low dimension spaces and therefore have to be associated with feature selection approaches that are beyond the scope of this paper.

The synthetic databases Gauss1D, Square and MultiBlueEyes were used as benchmarks for our proposal. For the Gauss1D database, the first class is represented by a multivariate normal distribution with means -3 and 1, and a standard deviation of 0.5. In the second class, the means are -1 and 3, with a standard deviation of 0.5. The second database represents a two-class problem in 2 dimensions where data are included in a square divided in four quarters. Each class is represented by two opposite quarters.

The MultiBlueEyes sets are derived from the popular "Blue eye" classification set. The latter is a two-class problem in which patterns of the first class are located in a first inner sphere and the others in a second sphere that has the same center but a greater radius. Several "Eyes" with different internal/external radii and positions were generated. The idea was to focus on different levels of overlapping between categories in order to study the behavior of our approach. Despite overlapping, these synthetic sets (except "BlueEyes 1") present no particular difficulties for classification purposes as the boundaries are only concerned by a small ratio of data (from 5% to 15%) and there is no difficulty to distinguish the nature of data (superfluous, critical, noise.). They are interesting for illustrating the characteristics of our algorithm and can also reveal some differences with other approaches.

Later, in order to better measure the efficiency of our method in large databases, some original and popular databases (7, 8, 12–16) [5] were inflated to make the problem harder in terms of runtime. Inflation is not just simple duplication as a little random noise was added to each duplicated pattern (limit of 10% of the standard deviation of each variable). This is the case of the popular Iris database; two versions were generated, one with two features and the second with four features. The skin data set was collected by randomly sampling B, G, R values from face images of various age groups (young, middle-aged, and old), race groups (white, black, and Asian), and genders obtained from various biometric databases. The bank database is related with direct marketing campaigns of a Portuguese banking institution based on phone calls. More than one contact to the same client was often required, in order to access if the product would be subscribed to or not. The classification goal is to predict whether the client will subscribe to a term deposit. The sensor databases are related to robot navigation on the basis of two and four sensors. The RedWine database has been recently studied [13] and the expert evaluations have been grouped together in three balanced categories. This database is related to red variants of the popular Portuguese

Table 1
Database characteristics

Database	n°	Patterns	Features	Classes
Gauss1D	0	40000	1	2
Squares	1	20000	2	2
BlueEyes-0	2	24000	2	2
BlueEyes-1	3	12000	2	2
BlueEyes-2	4	12000	2	2
BlueEyes-3	5	12200	2	2
BlueEyes_4		12400	2	2
Iris_0	7	15000	4	3
Iris_1	8	15000	3	3
Skin	9	245056	3	2
Bank	10	45222	10	2
Telescope	11	19020	5	2
Sensor_2D	12	27280	16	2
Sensor_4D	13	27280	2	4
Wine	14	27280	4	4
Banana	15	7995	11	2
Shuttle	16	5300	2	2
Phoneme	17	57758	8	3

Table 2
Reference classification scores

Database	Reference classification score
0	96.99%
1	99.99%
2	94.83%
3	49.77%
4	97.97%
5	89.75%
6	88.00%
7	98.79%
8	96.59%
9	98.00%
10	76.00%
11	71.50%
12	97.59%
13	91.59%
14	91.00%
15	86.00%
16	84.64%
17	97.96%

"Vinho Verde" wine. Features are physicochemical inputs and the category corresponds to a sensory evaluation. The "banana" and "shuttle" databases are available in repositories [5] and the phoneme one was used in the European ESPRIT project [2]. The aim of the phoneme database is to distinguish between nasal and oral vowels. It contains vowels coming from isolated syllables characterized by five different attributes representing the amplitudes of the first harmonics.

For all the databases, we generated reference classification scores by dividing each database into two subsets (30%, 70%) and applying a 1-nn procedure to the second set by using the first set as reference. The "so-called" reference results are shown in Table 2.

4.2. *Preliminary tests*

The purpose of this experiment was to study the behavior of our approach when some relevant input parameters vary in a coherent scale while the others are fixed at nominal values (Table 3). These latter parameters have less influence. They have been calibrated for a class of problems to avoid the user to being faced with all the computation complexity. Then the user only needs to follow a set of input parameters, and specify a few secondary parameters. The efficiency of the algorithm has to be measured not only by the best performances obtained by tuning input parameters, but also by its capacity to always give satisfactory results with pre-calibrated parameters.

4.2.1. *Effect of N_r and N_s*

We performed experiments using different values for N_r (from 10 to 50) and N_s (from 100 to 280) to measure the role of these two parameters and especially to show the consistency of our algorithm. The trend is the following: the relative importance of these parameters is linked to the complexity of the databases. Results are very stable with easy databases and a little less so for more complex ones. We have selected two representative databases to discuss the results. Figure 6 shows the average values of testing error, storage requirements and execution time for databases 8 and 11, respectively.

The results of the two databases are very different. The first database does not present any difficulty for neighbor classifiers even if it shows a slight amount of overlapping. Our approach gives optimal results

Table 3
Nominal input parameters

Parameter	Nominal value	Description
n/Nr	1000	General hybrid parameters
Ns	100	
Kmin	10	
m_size	100.00	
Th	2.00%	Fuzzy theshold
Tp	98.00%	Purity parameter
a_1	1	Umbalanced distribution
a_2	20	
Tσ	0.90	Stratification parameters
λ	10.00%	
μ	20.00%	
Wf	1	
Itmax	50	

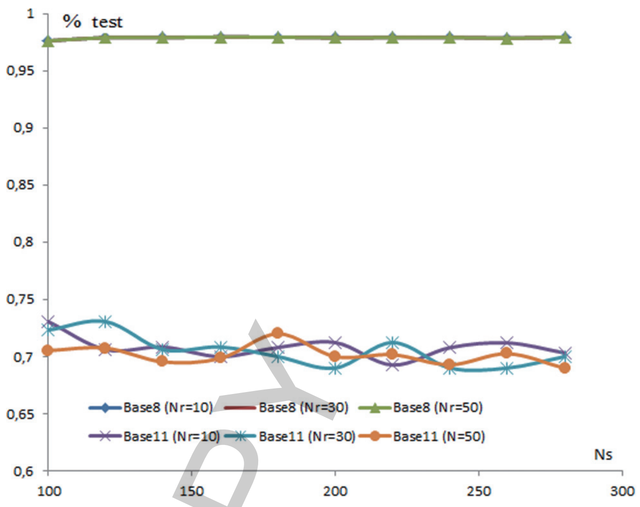


Fig. 6. Effect of N_r and N_s for databases 8 and 11. (Colours are visible in the online version of the article; http://dx.doi.org/10.3233/IDA-150736)

for the percentage of classification, the reduction rate is around 0.25% and the runtime ranges between 110 ms and 130 ms. These times are related to the size of the stratum, the maximum being reached for strata with 160 patterns and 10 regions. On this basis, the algorithm is almost insensitive to the level of stratification. Whatever the configuration, the results are close to 98% of correct classification. The second database presents more classification difficulty; the results remain very close to the reference ones but show some variability. The processing time and reduction rates remain acceptable whatever the configuration. The results in the range of input parameters provide evidence against some aspects of our method that deserve comment. Some variability in the classification results can be observed, less than 3% for the test set, while it seems difficult to determine rules or discern a general tendency in the results. It is necessary to extend the range to obtain more interesting information. By fixing $N_r = 1$ and varying N_s from 50 to 1000, the best classification results are obtained when the level of stratification is weak ($N_s = 1000$). Worse classification results (around 60%) appear when the stratification level is high ($N_s = 50$). In this case, degradation is directly related to the level of stratification. This highlights a weakness of pure stratification approaches that may encounter some difficulties when the configuration of classes is complex (partial overlap, heterogeneous shapes). It should be mentioned that the size of strata is not however a relevant factor in our context. The degradation due to stratification is attenuated as it is applied on the regions and not on the whole data set.

4.2.2. *Effect of λ and μ*

We have performed experiments using different values of λ (from 5% to 70%) and μ (from 5% to 60%) by fixing $w_f = 1$. Figure 7 shows the average values of testing error, storage requirements and execution time with the different values for database 16. The latter is particularly interesting as categories partially overlap and some noise is present in the data. The range of parameters for the test is intentionally broad to highlight their influences. There is a change in the classification score around 10% with values ranging roughly from 74% to 87% for the training set, and from 72% to 86% for the test set. Regarding the storage requirements, it goes from 0.60% to 10.00% while the computational time remains relatively low (less than 30 ms).

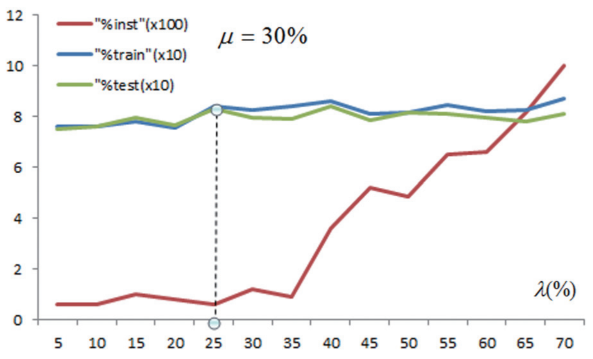


Fig. 7. Effect of λ and μ for database 16. (Colours are visible in the online version of the article; http://dx.doi.org/10.3233/IDA-150736)

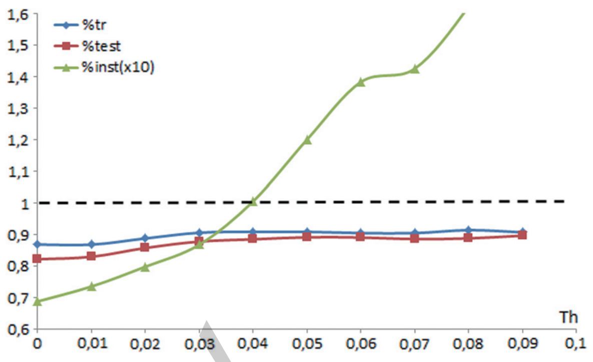


Fig. 8. Effect of the fuzzy 1-NN option for database 15. (Colours are visible in the online version of the article; http://dx.doi.org/10.3233/IDA-150736)

The larger the value of λ, the greater the proportion of prototypes that will be allowed. Concerning μ, it controls the presence of non-representative prototypes favoring more details of the training set. In extreme cases, the danger is to be unable to provide general classification systems as testing errors are deteriorated. It should be underlined that the couple $\lambda = 70\%$, $\mu = 30\%$ achieves the best score (87%) for the classification training set. However, the consequence is a lower classification test score (82%) and the need to store 10.00% of prototypes, which is a dramatic increase when compared to 0.60%.

On the other hand, too low values for both criteria will not allow the complexity of data to be accounted for. The granularity remains too coarse. It appears that with the couple ($\lambda = 5\%$, $\mu = 30\%$) the classification training and test scores are almost the same (76% and 75%), and very few prototypes are concerned (0.60%). The calibrated level of granularity is insufficient for this database. The ranges of acceptable parameters are respectively for λ and μ [0%; 20%] and [0%; 30%]. This calibration makes it possible to control the granularity while limiting the number of prototypes.

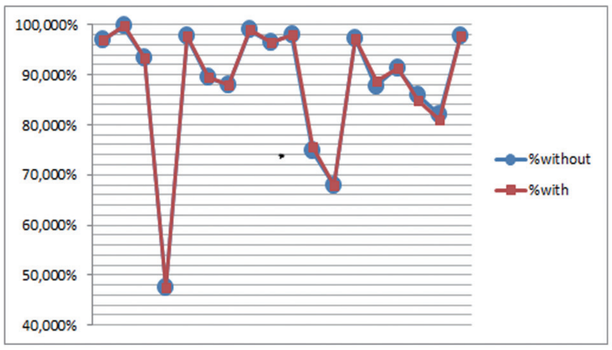
4.2.3. *Effect of the fuzzy 1-NN option*

We performed experiments using different values of T_h (defined in Section 3.1.2) from 0% to 9% to measure the role of the fuzzy mechanisms.

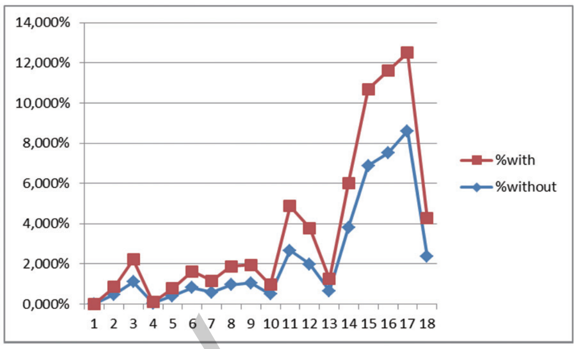
Figure 8 shows the average values of testing error, storage requirements and execution time with the different values for database 15. It is interesting as the proportion of pure regions is weak and there is no possible reduction. Furthermore, there is non-negligible overlapping between the categories.

Regarding storage requirements, the effect is clear. As T_h grows, the storage requirement automatically grows and, to a lesser extent, the execution time also. When T_h rises from 0 to 9%, the storage requirement increases from 6.88% to 18.92%. The performances vary with the boundary complexity and with the degree of overlapping between categories. As T_h increases under a given threshold, the algorithm is generally more efficient in achieving better classification results. However, this efficiency is achieved at the cost of deteriorating the storage requirement and execution time, which are larger.

We cannot establish a general rule affirming that the increment in the classification performance is larger than the increment in the storage requirements or execution time. For databases presenting no classification complexity for neighbor approaches, the effect of the fuzzy 1-NN option is negligible. The algorithm gives similar performances for the three criteria with or without considering the fuzzy option. Classification results are the same while the storage requirement and execution time are very close (less than 1% of difference). For databases presenting high complexity, in contrast, the increment in classification performances does not compensate for the deterioration of the other criteria, particularly



(a)



(b)

Fig. 9. (a) Classification score with and without selection for the 18; (b) Storage requirements with and without selection for the 18 databases. (Colours are visible in the online version of the article; http://dx.doi.org/10.3233/IDA-150736)

the runtime. More processing is needed for no improvement. We consider that most real databases are between these two extremes. Then, if we are mainly interested in classification performances we can use the fuzzy 1-NN option as a source of improvement. If the other criteria are more important, the fuzzy 1-NN option can be used but with moderation. We have calibrated T_h to 2% for all our tests. This slightly improves the classification performances without affecting the other criteria too much.

4.2.4. *Effect of the reduction process*

We propose two types of reduction for instances. The first concerns instances extracted from pure regions as presented in Section 3.4, whereas the second option addresses all instances. During the various experiments performed to show the components of our algorithm, we consistently applied the first reduction approach to show its contribution to our method. The first observation is that this procedure is not time-consuming because it finally addresses a reduction on a small set of data. We can see that the CPU based on a standard computer varies only in the range of 10 ms to 20 ms on all databases.

The second remark is that its effect depends on two factors: the complexity of the data and the number of subsets considered. The more complex the data are, the less efficient this option is as its contribution concerns only pure regions. For example, there is no consistent improvement for database 11. By increasing the number of clusters, the probability of pure regions is increased, making this option very useful. The storage requirements for database 9 decreases from 0.24% to 0.01% when $N_r = 50$ and $N_s = 280$ when the option is activated.

When data are simple, this option provides flexibility in choosing the number of clusters. Storage requirements are optimized whatever the initial configuration. Most databases that can be discriminated, at least partially, have necessarily pure and non-pure areas. In summary, this option compensates for the choice of the number of clusters without degrading the CPU performances.

The second reduction approach is not the main topic of this paper, but needs to be quickly discussed. We tested it on all the databases and its effect was particularly evident on the complex ones. It can significantly reduce the number of prototypes without deteriorating the classification performances. It agrees with literature results that highlight evolutionary methods as the most efficient ones. The counterpart is the CPU time. The difference with our approach is that GA is constraint that makes the process much faster. Figure 9 summarizes the contribution of an elementary GA in reducing the storage requirements. In the simulations, we set $T_h = 0.6$ to produce more instances and better delineate the reduction effect. The GA was configured with 10 chromosomes and launched for 5 minutes of processing.

4.3. Comparison with other approaches

This section highlights the adaptability of our solution (called “Hyb”) versus other approaches by starting with pre-calibrated parameters. In our framework, a generic calibration is sufficient to deliver coherent classification results, very quickly and without any parameter tuning. We compared our approach with popular selection approaches and the most similar instance selection algorithms.

1. CNN: Condensed Nearest Neighbor Rule [46].
2. FCNN: Fast Nearest Neighbor by Cluster [4].
3. “Strat” – Classic Stratification scheme: Stratification as presented in [9].
4. “StratRec” – Recursive stratification approach [27]: the method divides the original training set into small subsets on which the instance selection algorithm is applied. Then the selected instances are combined in a new training set and the same procedure, partitioning and application of an instance selection algorithm, is repeated.
5. “Cl” – Clustering approach: The prototype selection is based on clustering (c-means approach), and selects border prototypes and some interior prototypes according to [36].
6. “Hill” – Multi Hill Climbing: N binary chromosomes evolve in parallel by switching one component at each iteration [45].

4.3.1. Configuration of the experiments

For each database, all the algorithms were launched in order to understand their behavior and study some inherent limitations. Different tests were carried out by varying the relevant input parameters that are available for users. For CNN and FCNN, there is no need to define parameters. Concerning the cluster approach, the number of desired clusters was selected; the stopping criteria were the number of iterations and the average distances between mean centers in two successive iterations. In this case, the number of iterations determines the number of times the instance selection algorithm is applied. For “Strat”, the size of each stratum and the number of strata were selected. For large databases, we limited the number of strata to 30 (pre-calibration) for the simulation as the CPU burden became unacceptable, without any visible improvements in accuracy as the reduction technique becomes similar to a simple random sampling. In any cases, the stratification process was stopped as soon as the number of instances was more than 35% for the same reason.

Concerning the recursive stratification method, we performed experiments using subset sizes of 250, 500 and 1000 instances. Each configuration was repeated 10 times.

To produce comparable results, FCNN was also used as the IS algorithm for stratification approaches.

For “Hill”, the number of chromosomes was set. For each of the alternative algorithms, the runtime necessary to determine the instances, the classification performances, and the percentage of reduction were evaluated (Figs 10–12). The reduction percentage corresponds to the ratio between the number of selected instances and the cardinality size of the database. Concerning our algorithm, we used the same pre-calibrated parameters and we varied N_s from 100 to 1000 and put N_r so that $n/N_r = 1000$, within the constraint of having $N_r = 3$ as a minimum. For the implementation of the algorithm, some precautions were taken to respect integrity and allow the software to run whatever the input parameters.

4.3.2. Results and discussion

The CNN and FCNN approaches belong to the same family since they can generate as many instances as necessary to classify the whole training set. Except for trivial databases like 2 or 7, CNN generates many instances and a high computational cost in relation to the complexity of decision boundaries. Concerning FCNN, better performances are obtained than with the pioneer CNN approach. FCNN remains

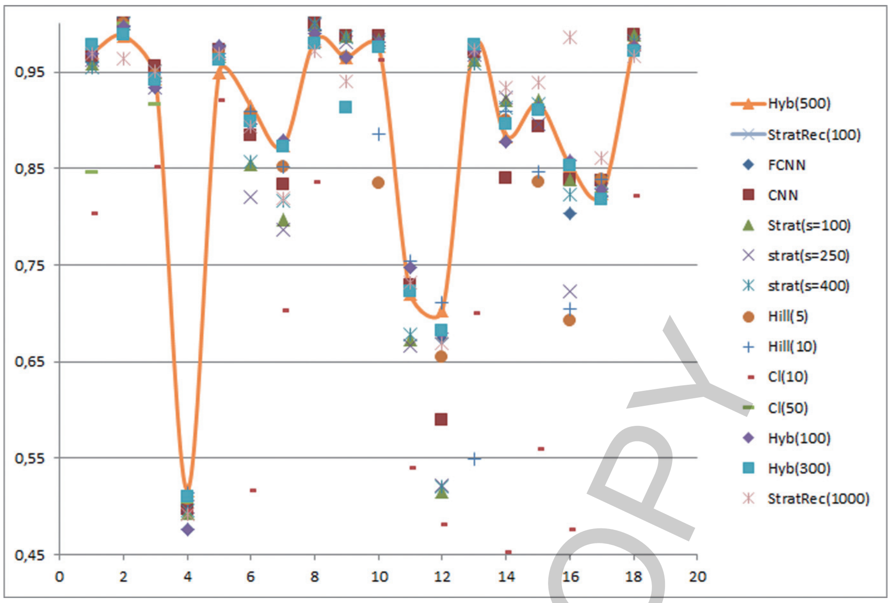


Fig. 10. Comparison with other instance selection algorithms: classification performances. (Colours are visible in the online version of the article; http://dx.doi.org/10.3233/IDA-150736)

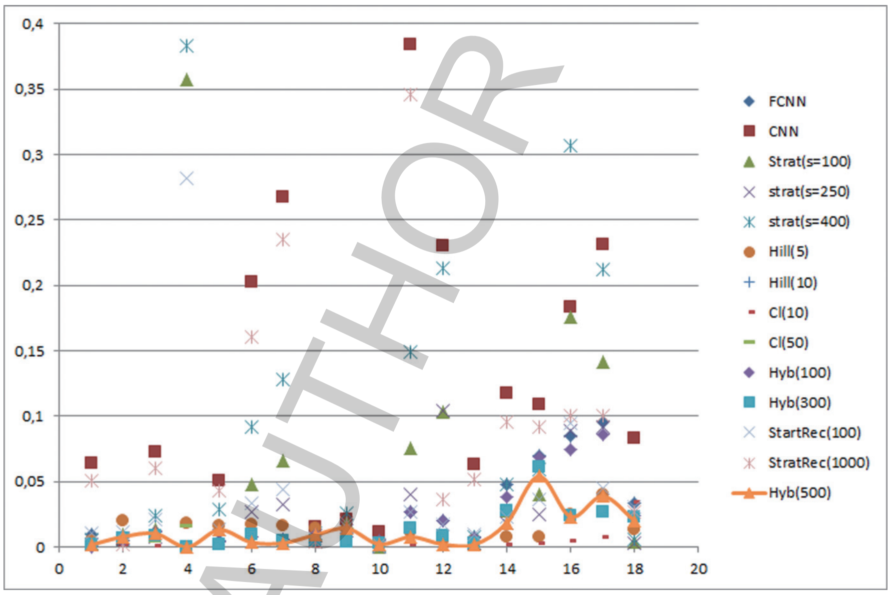


Fig. 11. Comparison with other instance selection algorithms: storage requirements. (Colours are visible in the online version of the article; http://dx.doi.org/10.3233/IDA-150736)

sensitive to noise compared to the other algorithms. The classification results are very close to the reference results for simple databases. It is however very powerful in terms of CPU. Except for databases 10 and 18, it required a CPU runtime of less than 100 ms to generate instances.

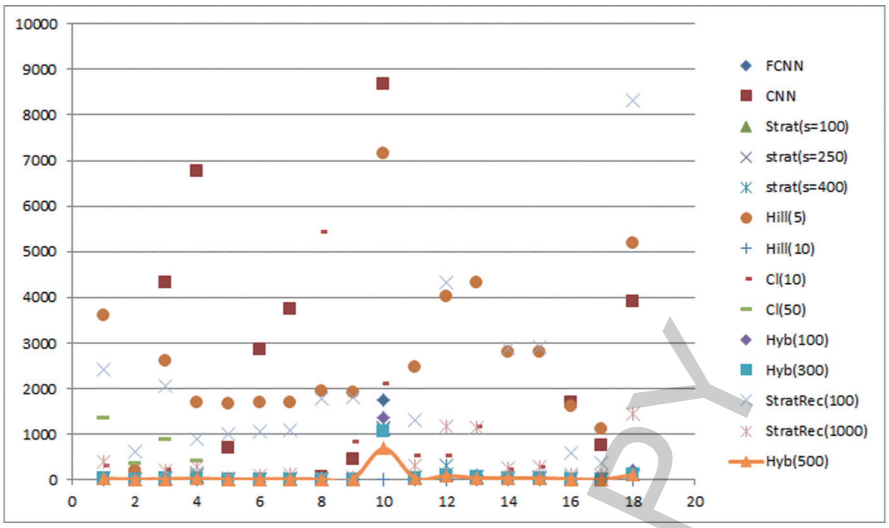


Fig. 12. Comparison with other instance selection algorithms: computational performances. (Colours are visible in the online version of the article; http://dx.doi.org/10.3233/IDA-150736)

Concerning "Strat", the method generally produces more instances than necessary. These results point out several factors: first, the ability of the approach to optimally treat cases having no particular classification complexity. Whatever the configuration parameters, the approach provides excellent results comparable with the best obtained by other approaches.

Concerning more difficult databases, particularly with a high degree of noise, the method reaches its limits in terms of performance. Each stratum generates many instances producing large sets for a long processing time. Due to the stratification process, data are degraded and these instances are not always optimal. In this case, the algorithm delivers poor classification results far from the reference ones and results are very heterogeneous from one configuration to another.

Concerning "StratRect" in its basic version, the same comments about the classification scores can be made. Stable and satisfactory results were obtained for easy databases but much greater difficulties were encountered for the complex ones. Overall, the results are better when increasing s (size of the stratification set). The method is known to be fast but the recursion scheme is more computational than the conventional method of stratification. In terms of storage requirements, there is the same problem as for conventional methods. The complexity of the databases results in the generation of many instances. The application of a selective method is possible but the scheme needs a heavy CPU time.

Concerning "Cl", the runtime can be very long even for synthetic databases. Nevertheless, we can observe more stable results than for the stratification approach. It seems that the approach presented in [30] is more efficient when the number of clusters is large and unsuitable when there are few clusters. This is very clear for database 7, where the results rise from (83%, 84%) to (98%, 98%). Generally, it seems difficult to extract a configuration that works well for any databases. We can only infer that a fine granularity is necessary to match the complexity of decision boundaries and make the rule presented in [30] efficient. It can be seen that the runtime increases very significantly when the number of clusters is larger. With many clusters, the time required by this approach is incomparably longer than that required by most of the other approaches. When a standard IS is applied on each cluster instead of the rule presented in [30], the runtime problem is the same.

Concerning "Hill", our results confirm those of recent studies [22]: the approach provides a good compromise between the various criteria especially for the classification score and storage requirements.

For example, for database 9, only 0.08% of instances were needed in order to classify the optimal set with a moderate processing cost. But "Hill" remains an evolutionary approach without embedded intelligence and there is a risk of premature convergence. In the application context (small number of chromosomes and five minutes of processing), the results concerning database 12 are far from the ones obtained with the other approaches. If we ignore this convergence issue, the major weakness of the approach remains the processing time. Even for databases of low complexity and by considering only few chromosomes, the search for nearest neighbors requires computing a lot of distances automatically penalizing the approach for large databases. Limiting the computational costs is always a possibility but the counterpart is degradation of the classification results.

Unlike the above algorithms that can fail for a specific database on one or two criteria, the results obtained within our framework are very acceptable whatever the database. Most of the parameters are pre-calibrated and the remaining ones can be changed in accordance with the objective. For easy databases, classification performances are optimal and the other criteria are close to the best approaches. Some major differences between the different databases can be pointed out but they are related to the complexity. For example, less than 30 ms are necessary to classify database 2, whereas more than 600 ms are required to mine database 9. The runtime is dependent on the database size but also on the classification problem. This criterion stands for the decision boundary complexity and the degree of overlapping between classes. Less than 50 ms are needed to handle database 1 containing 40000 patterns presenting rather simple boundaries (less than 0.2% of instances are necessary) and a slight overlapping between classes (about 2.5% overlap with one another).

Globally, our approach was developed with the objective of achieving the best hybridization possible between existing techniques or concepts and reducing the weaknesses identified. The idea of segmentation by region is included but without the disadvantages of clustering approaches; the stratification approach is applied but on regions and not on the whole data set, thus greatly minimizing degradation; lastly, the fastest instance selection algorithm (FCNN) is considered by reducing its sensitivity to noise. It is quite logical that the results are generally better and more regular than those obtained with the different approaches taken separately.

The reduction criterion calls for several remarks. First, although the "scale" process including stratification logically encourages a larger number of instances, the results remain satisfactory even without the application of the filtering process. Second, we can observe, especially for the synthetic databases, that our algorithm is able to deliver only the necessary instances while discarding the superfluous ones, as evidenced by the results. It can therefore be claimed that we have introduced different mechanisms to optimize the number of instances. While they tend to affect the classification accuracy slightly, they enable the number of instances to be greatly reduced. For complex databases, classification performances are close to the optimal and the algorithm outperforms the other approaches on the basis of one or two criteria.

In a discovery process, the classification accuracy remains important but the essential issue is not to change the reference result drastically. It is also mandatory to deliver understandable and viewable information that provides the expert with a better exploration support. Within this aim, tractability and flexibility are critical, and our approach contributes well in this direction. It allows specific trials for an expert and can be easily included in a global framework where thousands of trials are often necessary to provide useful information.

We believe that the level of self-tuning achieved in our framework is interesting. There is however room for improvement by categorizing the classification difficulty of the database being processed (boundary complexity, data heterogeneity, level of noise). This would make it possible to better adapt the hybridization strategy and directly include the parameters that are the most appropriate; more "anticipation" and less "correction". To be advantageous in terms of tractability, the computational cost of the required pre-processing step has to be less than that of the process itself. Some optimizations are therefore underway to make the progress more automatic and reliable for managing very large databases and tackle different complexities. They especially concern the automatic design of Nr and Ns that have to be managed jointly.

5. Conclusion

In this paper, we have presented a hybrid method for scaling up instance selection algorithms. The scalability concerns the division of the algorithm into regions that are partitioned in strata. An instance selection algorithm is applied to each subset, and then the selected instances are filtered and combined to form the final set. Additional mechanisms are suggested to limit the number of strata, filter superfluous instances and make the method flexible to input pre-calibrated parameters. We have shown that our method is able to match the performance of the original algorithms with a better consistency. Furthermore, we have also shown that our approach is able to scale up to very large problems with hundreds of thousands of instances. Experiments performed with various databases revealed the effectiveness and applicability of the approach.

References

[1] D. Aha, D. Kibler and M.K. Albert, Instance-based learning algorithms, *Journal of Machine Learning* **6** (1991), 37–66.
[2] P. Alinat, Periodic progress report 4, Tech Rep, ROARS Project ESPRIT II- Number 5516 (1993).
[3] F. Angiulli and G. Folino, Distributed nearest neighbor-based condensation of very large data sets, *IEEE Transactions on Knowledge and Data Engineering* **19**(12) (2007), 1593–1606.
[4] F. Angiulli, Fast nearest neighbor condensation for large data sets classification, *IEEE Transactions on Knowledge and Data Engineering* **19**(11) (2007), 1450–1464.
[5] A.A. Asuncion and D. Newman, UCI machine learning repository, 2007. http://www.ics.edu/∼mlearn/MLRepository.html.
[6] J.C. Bezdek and L.I. Kuncheva, Nearest prototype classifier designs: An experimental study, *International Journal of Intelligent Systems* **16** (2001), 1445–1473.
[7] H. Brighton and C. Mellish, Advances in instance selection for instance-based learning, *Data Mining and Knowledge Discovery* **6**(2) (2002), 153–172.
[8] E. Byon, A.K. Shrivastava and Y. Ding, A classification procedure for highly imbalanced class sizes, *IIE Transactions* **43** (2010), 288–303.
[9] J.R. Cano, F. Herrera and M. Lozano, Stratification for scaling up evolutionary prototype selection, *Pattern Recognit Lett* **26**(7) (2005), 953–963.
[10] J.-R. Cano, F. Herrera and M. Lozano, Evolutionary stratified training set selection for extracting classification rules with trade off precision-interpretability, *Data and Knowledge Engineering* **60**(1) (2007), 90–108.
[11] C.-A. Li, Credit scoring analysis using B-cell algorithm and K-nearest neighbor classifiers, *lecture notes in computer science,Advances in Swarm Intelligence* (2013), 191–199.
[12] J.J. Chen, C.A. Tsai, J.F. Young and R.L. Kodell, Classification ensembles for unbalanced class sizes in predictive toxicology, *SAR and QSAR in Environmental Research* **16**(6) (2005), 517–529.
[13] P. Cortez, A. Cerdeira, F. Almeida, T. Matos and J. Reis, Modeling wine preferences by data mining from physicochemical properties, in: *Decision Support Systems, Elsevier* **47**(4) (2009), 547–553.
[14] T.M. Cover and P.E. Hart, Nearest neighbor pattern classification, *IEEE Transactions on Information Theory* **13**(1) (1967), 21–27.
[15] B.V. Dasarathy, Minimal consistent set (mcs) identification for optimal nearest neighbor decision systems design, *IEEE Transactions on Systems Man and Cybernetics* **24**(3) (1994), 511–517.
[16] C. Domingo, R. Gavalda and O. Watanabe, Adaptive sampling methods for scaling up knowledge discovery algorithms, *Data Mining and Knowledge Discovery* **6**(2) (2002), 131–152.

[17] P. Domingos and G. Hulten, A general framework for mining massive data streams, *Journal of Computational and Graphical Statistics* **12**(4) (2003).

[18] R. Duda, Hart PE pattern classification and scene analysis, *Wiley* New York, (1973).

[19] A. Fred, Finding consistent clusters in data partitions, in: *Multiple Classifer Systems* **LNCS 2096** Springer (2001), 309–318.

[20] M. Fazzolari, R. Alcala, Y. Nojima, H. Ishibuchi and F. Herrera, A review of the application of multiobjective evolutionary fuzzy systems: Current status and further directions, *IEEE Trans Fuzzy Syst* **21**(1) (2013), 45–65.

[21] M.J. Gacto, R. Alcala and F. Herrera, Adaptation and application of multi-objective evolutionary algorithms for rule reduction and parameter tuning of fuzzy rule based systems, *Soft Computing* **13** (2009), 419–436.

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

[23] N. García-Pedrajas and A. Haro-García, Scaling up data mining algorithms, review and taxonomy, *Prog Artif Intelligence* **1** (2012), 71–87.

[24] G.W. Gates, The reduced nearest neighbor rule, *IEEE Transactions on Information Theory* **22** (1972), 431–433.

[25] J. Gehrke, R. Ramakrishnan and V. Ganti, RainForest – A framework for fast decision tree construction of large datasets, *Data Min Knowl Discovery* **4**(2/3) (2000), 127–162.

[26] S. Guillaume, Designing fuzzy inference systems from Data: An interpretability-oriented review, *IEEE Transaction on Fuzzy Systems* **9**(3) (2001).

[27] A. Haro-Garcia and N. Garcia-Pedrajas, A divide-and-conquer recursive approach for scaling up instance selection algorithms, *Data Mining and Knowledge Discovery* **18**(3) (2009), 392–418.

[28] M.K. James, M.R. Gray and A.G. James, A fuzzy K-nearest neighbor algorithm, *IEEE Transactions on Systems Man and Cybernetics* **15**(4) (1985).

[29] N. Jankowski and M. Grochowski, Comparison of instances selection algorithms i, algorithms survey, in: *Artificial Intelligence and Soft Computing* (2004), 598–603.

[30] A.M. Kibriya and E. Franck, An empirical comparison of exact nearest neighbor algorithms, in: *Proceedings of the 11*$^{\text{th}}$ *European Conference on Principles and Practice of Knowledge Discovery in Databases* (2007), 140–151.

[31] S.W. Kim and J. Oomenn, A brief taxonomy and ranking of creative prototype reduction schemes, *Pattern Analysis and Applications* (6) (2003), 232–244.

[32] S.W. Kim and B.J. Oommen, On using prototype reduction schemes to optimize dissimilarity-based classification, *Pattern Recognition* **40**(11) (2007), 2946–2957.

[33] G.D. Kollios, D. Gunupulos and S. Koudas, Berchtold, An efficient approximation scheme for data mining tasks, *Proceedings 17th International Conference on Data Engineering* (2001).

[34] E. Marchiori, Hit miss network edition iterative, *The Journal of Machine Learning Research* **9**(6) 1 (2008), 997–1017.

[35] B. Nitin and S. Vandana, Survey of nearest neighbor techniques, *International Journal of Computer Science and Information Security* **8**(2) (2010).

[36] J.A. Olvera-Lopez, J.A. Corrasco-Ochoa and J.F. Martínez-Trinidad, A new fast prototype selection method based on clustering, *Pattern Analysis and Application* **13**(2) (2010), 131–141.

[37] J.A. Olvera-López, J.A. Carrasco-Ochoa, J.F. Martínez-Trinidad and J. Kittler, A review of instance selection methods, *Artif Intell Rev* **34** (2010), 133–143.

[38] A.N. Papadopoulose and Y. Manolopoulos, Nearest neighbor search: A database perspective, Springer, 2004.

[39] J. Perez-Rodriguez, A.D. Har-Garcia and N. Garca-Pedrajas, Instance selection for class imbalanced problems by means of selecting instances more than once, *CAEPIA'11 Proceedings of the 14th International Conference on Advances in Artificial Intelligence: Spanish Association for Artificial Intelligence* (2011), 104–113.

[40] F. Provost and V. Kolluri, A survey of methods for scaling up onductive algorithms, *Data Mining and Knowledge Discovery* **2** (1990), 131–169.

[41] Q.P. He, Fault detection using the k-nearest neighbor rule for semiconductor manufacturing processes, *IEEE Transactions on Semiconductor Manufacturing* **20**(4) (2007).

[42] G.L. Ritter, H.B. Woodruff, S.R. Lowry and T.L. Isenhour, An algorithm for a selective nearest neighbor decision rule, *IEEE Transactions on Information Theory* **21**(6) (1975), 665–669.

[43] F. Ros and S. Guillaume, An efficient nearest classifier, book chapter of hybrid evolutionary systems, *Studies in Computational Intelligence*, Springer Verlag **75** (2007).

[44] F. Ros and R.Harba, Use of neighborhoord and stratification approaches to speed up instance selection algorithms, *International Journal of Computer Information Systems and Industrial Management Applications* **4** (2012), 537–545.

[45] D.B. Skalak, Prototype and feature selection by sampling and random mutation hill-climbing algorithms, *Proceedings of the Eleventh International Conference on Machine Learning New Brunswick*, N J: Morgan Kaufmann (1994), 293–301.

[46] C.W. Swonger, Sample set condensation for a condensed nearest neighbor decision rule for pattern recognition, *Watanabe S Ed Academic Orlando* (1972), 511–519.

[47] G.T. Toussaint, Proximity graphs for nearest neighbor decision rules: Recent progress, in: *Interface Symposium on Computing and Statistics* (2002), 83–106.
[48] D. Wang, X.J. Zeng and J.A. Kean, A clustering algorithm for radial basis function neural network initialization, *Neurocomputing* **77** (2012), 144–155.
[49] D.R. Wilson and T.R. Martinez, Reduction techniques for instance-based learning algorithms, *Machine Learning* **38**(3) (2000), 257–286.
[50] D.R. Wilson and T.R. Martinez, An integrated instance-based learning algorithm, *Computational Intelligence* **16** (2000), 1–28.
[51] D.R. Wilson and A.R. Martinez, Instance pruning techniques, in: *Machine Learning: Proceedings of the Fourteenth International Conference*, D. Fisher, ed., San Francisco, CA. Morgan Kaufmann, 1997.
[52] T. Yu, L. Davis, C. Baydar and R. Roy, Evolutionary computation in practice, *Studies in Computational Intelligence*, Springer Berlin **88** (2008).
[53] X. Zhu and X. Wu, Scalable representative instance selection and ranking, in: *Proceedings of the 18*$^{\text{th}}$ *International Conference on Pattern Recognition, IEEE Computer Society* **3** (2006), 3522–3533.