A review of instance selection methods

Article *in* Artificial Intelligence Review · August 2010

DOI: 10.1007/s10462-010-9165-y · Source: DBLP

José Arturo Olvera-López
Benemérita Universidad Autónoma de Puebla
31 PUBLICATIONS **302** CITATIONS
SEE PROFILE

José Francisco Martínez-Trinidad
Instituto Nacional de Astrofísica, Óptica y Electrónica (INAOE)
236 PUBLICATIONS **1,360** CITATIONS
SEE PROFILE

Josef Kittler
University College London
542 PUBLICATIONS **15,315** CITATIONS
SEE PROFILE

Artif Intell Rev (2010) 34:133–143
DOI 10.1007/s10462-010-9165-y

A review of instance selection methods

**J. Arturo Olvera-López · J. Ariel Carrasco-Ochoa ·
J. Francisco Martínez-Trinidad · Josef Kittler**

Published online: 27 May 2010

Abstract In supervised learning, a training set providing previously known information is used to classify new instances. Commonly, several instances are stored in the training set but some of them are not useful for classifying therefore it is possible to get acceptable classification rates ignoring non useful cases; this process is known as instance selection. Through instance selection the training set is reduced which allows reducing runtimes in the classification and/or training stages of classifiers. This work is focused on presenting a survey of the main instance selection methods reported in the literature.

Keywords Instance selection · Supervised learning · Data reduction · Pre-processing

1 Introduction

Several problems in machine learning require of automatically classifying some instances (in this work, each element in a training set is named instance); this can be done through the supervised classification task which is a process for assigning a class to unseen cases. In

J. A. Olvera-López (✉)
Benemérita Universidad Autónoma de puebla, Facultad de Ciencias de la Computación,
Av. San Claudio y 14 Sur,
Ciudad Universitaria, 72570 Puebla, Mexico
e-mail: aolvera@cs.buap.mx

J. A. Carrasco-Ochoa · J. F. Martínez-Trinidad
National Institute of Astrophysics, Optics and Electronics, Computer Science Department,
Luis Enrrique Erro No. 1, Sta. María Tonantzintla, 72000 Puebla, Mexico
e-mail: ariel@ccc.inaoep.mx

J. F. Martínez-Trinidad
e-mail: fmartine@ccc.inaoep.mx

J. Kittler
University of Surrey, Center for Vision, Speech and Signal Processing, Guilford GU2 7XH, UK
e-mail: j.kittler@surrey.ac.uk

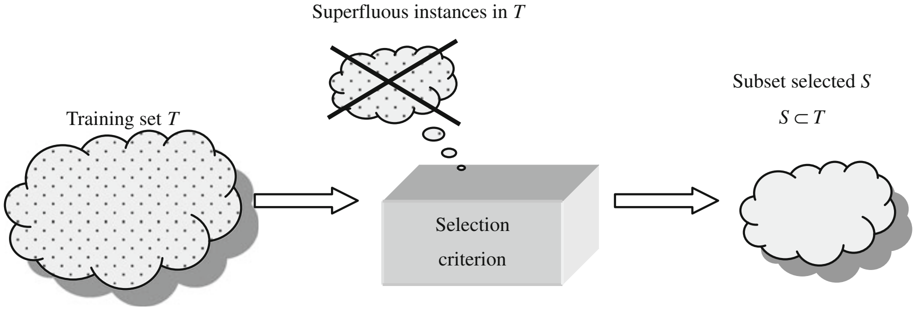

Fig. 1 Instance selection process

order to assign the class to a new case, a previously assessed set is provided to a classifier, this set is known as training set, denoted in this document as T.

In practice, T contains useless information for the classification task (that is, superfluous instances which can be noisy or redundant) therefore a process to discard them from T is needed.

Given a training set T, the goal of an instance selection method (Fig. 1) is to obtain a subset $S \subset T$such that S does not contain superfluous instances and $Acc(S) \cong Acc(T)$where $Acc(X)$ is the classification accuracy obtained using X as training set (henceforth, S is used to denote the selected subset). Instance selection methods can either start with $S = \emptyset$ (incremental method) or $S = T$ (decremental method). The difference is that the incremental methods include instances in S during the selection process and decremental methods remove instances from S along the selection.

Through instance selection, the training set is reduced, which could be useful for reducing the runtime in the training process, particularly in the classification process for instance-based classifiers since to classify just one instance, these classifiers use the whole training set.

Like in feature selection, according to the strategy used for selecting instances, we can divide the instance selection methods in two groups:

- *Wrapper.* The selection criterion is based on the accuracy obtained by a classifier (commonly, those instances that do not contribute with the classification accuracy are discarded from the training set).
- *Filter.* The selection criterion uses a selection function which is not based on a classifier.

The aim of this work is to present a review of instance selection methods, describing their main characteristics. The structure of this paper is as follows: in Sect. 2, wrapper and filter methods proposed in the literature related to the instance selection are described; Sect. 3 presents a discussion about the works described in Sect. 2. Finally, in Sect. 4, the conclusions are given.

2 Instance selection methods

Several methods have been proposed in the literature for solving the instance selection problem, in the next sections we present a review of both wrapper and filter strategies.

2.1 Wrapper methods

Most of the wrapper methods have been proposed based on the *k-NN* classifier (Cover and Hart 1967). One of the earliest methods is the *Condensed Nearest Neighbor* (CNN) (Hart 1968). It is an incremental method and its initial step consists in randomly including in S one instance belonging to each class. After the initial step, each instance in T is classified using S as training set, if an instance p is misclassified then p is included in S to ensure that new instances similar to p will be classified correctly; due to this criterion, noisy instances can be retained since they are commonly misclassified by their neighbors. An extension of CNN is the SNN *(Selective Nearest Neighbour rule)* Ritter et al. 1975 which finds a subset S such that every instance in T is closer to a member of S (in the same class) than to any instance in T, which means that T is correctly classified by *1-NN* using S. Another variant of CNN is the *Generalized Condensed Nearest Neighbor rule* (GCNN) (Chien-Hsing et al. 2006) which is identical to CNN but GCNN includes in S instances that satisfy an *absorption* criterion according to a threshold. For each instance, the *absorption* is computed in terms of the nearest neighbors and nearest enemies (nearest instances belonging to a different class). In this method, the instances in T that are absorbed (represented) by S are not included in S. In particular, an instance $p \in T$ is absorbed when $|p - x| - |p - w| > \delta$ where $x, w \in S$ and they are the nearest neighbor and nearest enemy of p respectively.

Another early instance selection method is the *Edited Nearest Neighbor* (Wilson 1972) which is focused on discarding noisy instances in a training set. This method discards an instance in T when its class is different from the majority class of its k nearest neighbors (ENN uses $k = 3$). An extension of ENN is the RENN (*Repeated* ENN) method which repeatedly applies ENN until all instances in S have the same class that the majority of their k Nearest Neighbors. Another variant of ENN is the *all k-NN* method (Tomek 1976) which works as follows: for $i = 1$ to k, flag such instances misclassified by its k nearest neighbors. Once the loop has finished, all flagged instances are discarded. Other method based on ENN is *Multiedit* (Devijver and Kittler 1980) which divides T in r blocks $B_1, B_2, \ldots, B_r$ and applies ENN over each block B_j but finding the neighbors in the next block, that is, in $B_{j+1 \bmod r}$, this process is repeated until there are not instance eliminations in t successive iterations. In Vázquez et al. (2005) a method for instance selection is proposed, which consists in applying ENN but using the probability of belonging to a class instead of the *k-NN* rule.

In Aha et al. (1991) the IB2 and IB3 (Instance Based) methods were proposed; they are incremental methods, IB2 selects the instances misclassified by 1-*NN* (as CNN); IB3 is an extension of IB2 where a classification record is used in order to determine the instances to be retained (instances such that their deletion does not impact the classification accuracy).

Other methods related to *k-NN* are those proposed by Wilson and Martínez (2000). They proposed five methods: DROP1, DROP2,...,DROP5 (*Decremental Reduction Optimization Procedure*); these methods are based on the concept of *associate*. The *associates* of an instance p are those instances such that p is one of their k nearest neighbors. DROP1 discards an instance p from T if the associates of p in S are correctly classified without p; through this rule, DROP1 discards noisy instances since the associates of a noisy instance can be correctly classified without it but in DROP1, when the neighbors of a noisy instance are first removed, then the noisy instance will not be discarded. In order to solve this problem, DROP2 is similar to DROP1 but the associates of an instance are searched in the whole training set, that is, p is deleted only if its associates in T are classified correctly without p. DROP3 and DROP4 first discard noisy instances using a filter similar to *ENN* and then they apply DROP2. DROP5 is based on DROP2 but it starts discarding the nearest enemies (nearest instances with different class) in order to smooth the decision boundaries.

Another method related to the *associate* set was proposed by Brighton and Mellish (2002), this method is the *Iterative Case Filtering algorithm* (ICF), based on the $Reacheable(p)$ and $Coverage(p)$ sets which are the neighbor and associate sets respectively. ICF discards p if $|Reachable(p)| > |Coverage(p)|$ which means that some instances in T can classify instances similar to p without considering it in the training set; as initial step, ICF applies ENN. In Ke-Ping et al. (2003), *C-Pruner*, another method based on the $Reacheable(p)$ and $Coverage(p)$ is presented. In this method, the $Coverage(p)$ concept only considers the associates with the same class as p in order to discard instances in the same class. Before discarding an element, this technique determines whether an instance is noisy, superfluous or critical. In this context, an instance is critical when its deletion affects the classification of other instances; in particular this method discards either noisy or superfluous (but non critical) instances. When $|Coverage(p)| < |Reachable(p)|$ then p is considered as noisy; p is superfluous when it is correctly classified by $Reachable(p)$.

In terms of instance selection, the SVM (Support Vector Machines) (Vapnik 1995) not only is a classifier but also an instance selector since among the elements in T, only the support vectors V_s are used to discriminate between classes; in this case, $S = V_s$. A wrapper method based on SVM is proposed in Yuangui et al. (2005) which works doing a double selection; the first one applies SVM to obtain V_s and the second one applies DROP2 over V_s. Another method related to instance selection based on SVM is SV-kNNC (*Support Vector k-Nearest Neighbor Clustering*) (Srisawat et al. 2006) which after applying SVM over T uses the *k-means* algorithm for clustering V_s and retains only such instances belonging to homogeneous clusters (clusters where all instances belong to the same class); when the cluster is non homogeneous, only such instances from the majority class in the cluster are preserved.

Evolutionary algorithms (EA) have been used for instance selection (Kuncheva 1995, 1997; Kuncheva and Bezdek 1998; Bezdek and Kuncheva 2001; Cano et al. 2003). EA are based on the natural evolution. Their main idea is as follows: given an initial chromosome population (set of solutions, in our context, a set of instances) and according to a fitness function (in the instance selection context, the fitness function is based on a classifier) the individuals from the population are evaluated, the best chromosomes are selected (which maximize the fitness function) in order to be muted and combined (crossover) for generating new chromosomes. The algorithm is repeated a specific number of iterations (generations) specified by the user and the best chromosome from the last generation is selected. Commonly, a chromosome C is represented as a binary coded array C = [1,0,0,1,…,], where $|\mathrm{C}| = |T|$, which is employed for representing a specific subset of T. each C_j represents the jth element in T. $\mathrm{C}_j = 1$ means that the jth element of T is considered in the subset, otherwise $\mathrm{C}_j = 0$. For the crossover operation, the bits of two individuals are swapped; the mutation operation inverts the values of some bits in the chromosome.

García et al. (2008) used a memetic algorithm for instance selection. Memetic algorithms combine evolutionary algorithms and local search within the evolutionary cycle. The local search is carried out among the chromosomes (local changes of bits) obtained during the evolutionary process; the local search changes 1–0 in the chromosome in order to improve both accuracy and reducing the size of the selected subset. i.e. the best chromosome in both accuracy and size is selected for the next evolutionary steps (crossover and mutation).

A biologically inspired approach for instance selection was proposed in Garain (2008), this approach was named *Clonal Selection Algorithm* (CSA). This method is based on the artificial immune system idea. When a pathogen attacks a body, a set of cells (antigens) work on defending it. In order to provide a response to a new pathogen, cells memorize and mute for adapting to the attack. Among all the cells, only some of them are able to attack the pathogens. Following this idea, CSA selects a set of antigens (instances) in the body (training set). CSA starts randomly selecting one antigen per class, after, a pathogen (set of instances) attacks the immune system and repeatedly a set of clones (the antigens most similar to the pathogen) are mutated in order to select the best ones against the pathogen attack.

In Cerverón and Ferri (2001), Zhang and Sun (2002) the tabu search (TS) (Glover 1986) was applied for selecting instances. TS is applied over an initial set (solution) $S_i \subset T$ and during the search process some solutions are declared tabu (i.e. a solution that should not be changed). Through S_i, other solutions (non tabu solutions) are evaluated (via a classifier) in order to select the best one. The way for choosing the solutions from S_i is evaluating the neighboring subsets, i.e. any subset that differs from S_i by just one element. Repeatedly (the number of iterations is controlled by a parameter named tabu tenure) S_i is replaced by a subset that yields better classification and once the process finishes, S is the best solution found so far.

Another way for finding sub optimal solutions in search problems is the sequential search which has been used in selection problems such as feature selection (Pudil et al. 1994; Blum and Langley 1997) and for instance selection (Olvera-López et al. 2005) where the BSE (*Backward Sequential Edition*) method uses the backward sequential search (Kittler 1986) in order to select S. BSE is a decremental method that evaluates every single instance elimination (using a classifier) and discards the instance with the lowest classification contribution for the partial subset; the process is repeated until the classification accuracy starts to decrease. In the sequential search a discarded instance cannot be re-considered (backtracking) once the selection criterion has decided to remove it from the partial subset. This drawback is solved with the sequential floating search (SFS) (Pudil et al. 1994) which like the sequential search, can be done in the forward and backward directions. SFS allows backtracking steps i.e. the inclusion/exclusion of elements previously discarded/included. SFS consists in applying after each forward/backward step a number of backward/forward steps (conditional inclusion/exclusion of the instances previously discarded/included) as long as the quality subsets are better than the previously evaluated ones. In Olvera-López et al. (2007a) the RFOS (*Restricted Floating Object Selection*) method was proposed. It adapts the SFS for selecting instances but in a restricted way since the whole SFS is a very expensive process. RFOS restricts the floating process applying only one forward /backward step followed by one backward/forward conditional step.

2.2 Filter methods

This section describes filter methods proposed in the literature; unlike wrapper methods they are not based on a classifier to determine the instances to be discarded from the training set.

In a training set, an instance can be either a border instance or an interior instance. The instance $p_j \in T$ is a border instance for the class C_i if $p_j \in C_i$ and p_j is the nearest neighbor of an instance from another class $C_k \neq C_i$. On the other hand, for the class C_i, p_j is an interior instance if it is not a border instance. The border instances of a class provide useful information for preserving the class discrimination regions (Wilson and Martínez 2000; Brighton and Mellish 2002); therefore some filter methods are focused on selecting border instances.

Riquelme et al. (2003) proposed the POP (*Pattern by Ordered Projections*) method that discards interior instances and selects some border instances. This approach is based on the *weakness*(p) concept which is defined as the numbers of times that p is not a border in a class (with respect to its attribute values). The selection rule discards irrelevant instances that according to this method, are those instances such that *weakness*$(p) = m$, where m is the total number of features describing p. The *weakness* of an instance is computed by increasing weakness for each attribute where the instance is not near to another instance with different class. According to this approach, in a two dimensional case ($m = 2$), for representing a region of instances in the same class, at most four instances are needed to determine the boundaries (two instances representing the maximum and minimum value per attribute) and in general in d-dimensions at most $2d$ instances are needed.

The POC-NN (*Pair Opposite Class-Nearest Neighbor*) method (Raicharoen and Lursinsap 2005) also selects border instances. The selection process in POC-NN computes the mean of the instances in each class. For finding a border instance p_{b1} in the class C_1, POC-NN computes the mean m_2 of the instances in the opposite class C_2 and then p_{b1} is the instance belonging to class C_1which is the nearest to m_2. The border instances in the class C_2 are found in a similar way; finally, only the border instances are selected.

A filter method based on *kd* trees (Friedman et al. 1997) was proposed in Narayan et al. (2006), where a binary tree is constructed. In the root node all the instances are included; for constructing each child node, a pivot is selected which is the feature with the maximum difference (*Maxdiff*) between consecutive ordered values; the left child contains those instances whose values for the corresponding attribute are lower than *Maxdiff* and the remaining instances are contained in the right node. The splitting process is repeated until the nodes cannot be split (the variance of the data along the features is greater than a threshold given by the user). Finally, instances located in the leaves are selected.

Some authors (Bezdek and Kuncheva 2001; Liu and Motoda 2002; Spillmann et al. 2006; Caises et al. 2009) have stated the idea of using clustering for instance selection; this idea consists of: after splitting T in n clusters, the selected instances will be the centers of the clusters. In Mollineda et al. (2002), the GCM (*Generalized-Modified Chang Algorithm*) method was proposed; this method merges the same-class nearest clusters and selects the centers from the new merged clusters. The NSB *(Nearest Sub-class Classifier)* method (Venmann and Reinders 2005) allows selecting different number of instances (centers) per class via the Maximum Variance Cluster algorithm (Venmann et al. 2002).

In Lumini and Nanni (2006), the CLU (*Clustering*) instance selection method which selects only the centers of the clusters was proposed; CLU was applied to signature recognition. The OSC (*Object Selection by Clustering*) (Olvera-López et al. 2007b) method, also based on clustering, selects border instances and retains some interior ones. OSC divides T in clusters; then border instances are searched in non homogeneous clusters (border regions) and the interior ones in homogeneous clusters. In order to find the border, the selection criterion considers p as a border if p is the closest to another instance in the cluster belonging to a different class. Although interior instances could be discarded from T, in order to preserve representative instances of homogeneous regions, OSC selects the center instance from each homogeneous cluster.

In the literature, some filter methods consist in assigning a weight to the instances and selecting those with an acceptable weights range (according to a threshold). The WP (*Weighting Prototypes*) (Paredes and Vidal 2000) method uses gradient descent for computing weights for each instance in terms of both nearest neighbors an nearest enemies; then the instances having weights larger than a certain threshold are removed. In Olvera-López et al. (2008) another filter method based on assigning weights to the instances was introduced: the PSR (*Prototype Selection by Relevance*) which computes the relevance of each instance in T. The relevance in this method is computed in terms of the average similarity i.e. the most similar among those in the same class the most relevant; PSR selects a percentage (specified

Table 1 Characteristics of the methods described in Sect. 2

Method	Kind	Based on	Reference
CNN	W	Misclassification	Hart 1968
SNN	W	Misclassification	Ritter et al. 1975
GCNN	W	Misclassification, *absorption*	Chien-Hsing et al. 2006
ENN	W	Misclassification	Wilson 1972
All k-NN	W	Misclassification	Tomek 1976
Multiedit	W	Misclassification	Devijver and Kittler 1980
IB	W	Misclassification	Aha et al. 1991
DROP	W	*Associates*	Wilson and Martínez 2000
ICF	W	*Reachable, Coverage*	Brighton and Mellish 2002
SV-kNNC	W	SVM, k-means	Srisawat et al. 2006
Evolutionary algorithms	W	Natural evolution	Kuncheva 1995, 1997; Kuncheva and Bezdek 1998; Bezdek and Kuncheva 2001; Cano et al. 2003
Memetic algorithms	W	Evolutionary algorithms, local search	García et al. 2008
CSA	W	Artificial immune systems	Garain 2008
TS	W	Tabu search	Cerverón and Ferri 2001; Zhang and Sun 2002
BSE, RFOS	W	Sequential search	Olvera-López et al. 2005, 2007a,b
POP	F	*Weakness,* border instances	Riquelme et al. 2003
POC-NN	F	Opposite class means, border instances	Raicharoen and Lursinsap 2005
kd-trees	F	kd-trees	Narayan et al. 2006
GCM	F	Clustering	Mollineda et al. 2002
NSB	F	Clustering	Venmann and Reinders 2005
CLU	F	Clustering, interior instances	Lumini and Nanni 2006
OSC	F	Clustering, border and interior instances	Olvera-López et al. 2007a,b
WP	F	Instance weight	Paredes and Vidal 2000
PSR	F	Instance relevance	Olvera-López et al. 2008

by the user) of the most relevant instances per class and through them, the most similar instances belonging to different classes (border instances) are also selected.

3 Discussion

In this section, we give a brief discussion about the characteristics and performance of the instance selection methods described in Sect. 2.

In Table 1, as a summary, the characteristics of the methods described in Sect. 2 are shown. This table contains four columns:

- *Method*. The name of the instance selection method.
- *Kind.* The kind of the instance selection method, Wrapper (W) or filter (F).
- *Based on.* The main idea on which the selection criterion is based on.
- *Reference.* The author(s) of the method.

It can be noticed that most of the methods are wrapper, mainly based on the *k-NN* rule. A characteristic of the earliest proposed CNN (its extensions and similar methods such as IB) is that they are based on retaining misclassified instances; this strategy retains noisy instances since they are misclassified by its neighbors. Among the methods based on this selection criterion, IB3 presents the best performance in both classification accuracy (obtained by S) and retention ($|S|$). The drawback of retaining noisy instances can be solved by using a noise filter method, the best known and with a good performance among its variants is ENN. Since in a training set only a few instances are noisy, the ENN reduction is not as big as in other instance selection methods.

Acording to reported experiments (Wilson and Martínez 2000; Brighton and Mellish 2002; Olvera-López et al. 2009), the DROP methods are among the most successful; the best one is DROP3 and it outperforms (in both classification and reduction) to several well known previous proposed wrapper methods and even to some filter ones. A more recently approach that is competitive in accuracy against the DROP3 (as it can be seen in Chien-Hsing et al. 2006; Olvera-López et al. 2008) is GCNN. When *k-NN* is used for evaluating *S,* DROP3 is a good option for selecting instances but some results reported (Grochowski and Jankowski 2004; Olvera-López et al. 2009) have shown that evaluating *S* using other classifiers, different from *k-NN*, the subsets selected by DROPs are not as good in accuracy as when *k-NN* is used; this is expected because like most of the wrapper methods, the DROPs selection criterion is based on the *k-NN* rule.

A characteristic of some wrapper approaches like TS, evolutionary algorithms, memetic algorithms, and those based on sequential search is that they can use any classifier for selecting *S* during the selection process; this is an advantage over other methods since the selection can be done with the classifier that will be used during the classification stage. TS has shown better performance in accuracy and retention than evolutionary algorithms (Bezdek and Kuncheva 2001; Cerverón and Ferri 2001) but the approaches based on sequential search (Olvera-López et al. 2008) outperform TS in accuracy.

Cano et al. (2003) presented an experimental study of different evolutionary algorithms; according to their results, the evolutionary CHC approach (heterogeneous recombination and cataclysm mutation) achieve the best performance in accuracy and retention. In addition, CHC was the method that required less runtime.

Filter methods have showed good performance with respect to wrappers, but only a few methods of this approach have been proposed (in contrast to the wrapper approach). Several selection criteria in filter methods are focused on selecting border instances (POP, POC-NN, WT) due to these instances delimits the frontier regions and preserving them allows representing all different class regions; other methods select both border and interior instances (OSC, PSR). According to the experimentation carried out by their authors (Paredes and Vidal 2000; Riquelme et al. 2003; Raicharoen and Lursinsap 2005; Olvera-López et al. 2005, 2007b, 2008), filter methods reached good accuracy and retention with respect to some methods such as CNN, IB2, ENN, DROP3 and GCNN.

The main characteristic of filter methods is that they spent less runtime than the wrapper ones (Raicharoen and Lursinsap 2005; Olvera-López et al. 2008). Additionally, since filter selection criterion is not related to a classifier, the obtained subsets achieve acceptable accuracies when different classifiers are used (as shown the experimentation in Riquelme et al. 2003; Olvera-López et al. 2007b, 2008).

Commonly, instance selection methods present the scaling up problem: runtimes grow (in some cases, the methods become inapplicable) when very large datasets are processed. Some ideas for reducing the runtimes have been proposed. Cano et al. (2005) proposed dividing T into r disjoint sets (strata) $D_1, D_2, \ldots, D_r$ of equal size and maintaining the class distribution of T; after the splitting phase, any instance selection method is separately applied over each D_j and S is the set of all the instances selected in each D_j. In De Haro-García and García-Pedrajas (2009) the same idea is followed but in an recursive way: first the data is divided into small disjoints subsets and then an instance selection method is separately applied over each one of them; after, the selected instances are joined in subsets of proximately the same size and the instance selection method is applied over these subsets; the process is repeated until a validation error starts to grow. This idea is not only focused on reducing runtimes but also on improving the classification accuracies (with respect to those obtained by the used instance selection method used over each subset). In order to do this, two approaches have been proposed; the first one consist in randomly including a percentage of the removed instances and the second one consists in discarding p only if two consecutive instance selection steps select p for removing. It is important to highlight that that the strategies proposed in Cano et al. (2005), De Haro-García and García-Pedrajas (2009) can be applied using any instance selection method but they do not constitute an instance selection by themselves.

4 Conclusions

In the field of supervised learning, instance selection is an important task, especially for instance-based classifiers, because through this task, superfluous instances are removed from the training set, which could reduce runtimes in the learning and classification stages.

Several authors have proposed different solutions for the instance selection problem; in this work, a review about instance selection methods was presented. The works described in this paper were grouped in two categories: wrapper and filter. Based on the results reported in the literature about instance selection, among the wrapper methods based on the *k-NN* rule, the best options for selecting instances are DROP3 and GCNN since they have good performance in both accuracy and retention; between them, the best reduction is obtained by DROP3 however, GCNN is a good option for medium/large datasets where DROP3 becomes inapplicable.

Another kind of wrapper methods are not restricted to use the *k-NN* rule during the selection process, in this case, TS, CHC, memetic algorithms and those based on sequential search seem to be a good option for selecting instances. These methods are competitive in accuracy (in some cases better) with those successful wrapper ones but they are much slower than *k-NN* based wrapper methods.

The selection criterion in filter methods is not based on a classifier, some methods of this type are focused on selecting border instances, among them, POC-NN, OSC and PSR have good performance, reaching with the selected subsets, acceptable accuracy rates when different classifiers are used. Filter methods are competitive with the wrapper in both accuracy and retention and, as an additional characteristic, filter methods are faster than wrapper when medium/large datasets are processed.

Instance selection methods have to deal with the scaling up problem when very large training sets are processed; for these cases, the strategies proposed in Cano et al. (2005), De Haro-García and García-Pedrajas (2009) are good options for dealing with this problem.

References

Aha DW, Kibler D, Albert MK (1991) Instance-based learning algorithms. Mach Learn 6:37–66

Bezdek JC, Kuncheva LI (2001) Nearest prototype classifier designs: an experimental study. Int J Hybrid Intell Syst 16(12):1445–1473

Brighton H, Mellish C (2002) Advances in instance selection for instance-based learning algorithms. Data Min Knowl Discov 6(2):153–172

Blum AL, Langley P (1997) Selection of relevant features and examples in machine learning. Artif Intell 97:245–271

Caises Y, González A, Leyva E, Pérez R (2009) SCIS: combining instance selection methods to increase their effectiveness over a wide range of domains. In: Corchado E, Yin H (eds) IDEAL 2009, LNCS 5788. Burgos, Spain, pp 17–24

Cano JR, Herrera F, Lozano M (2005) Stratification for scaling up evolutionary prototype selection. Pattern Recognit Lett 26:953–963

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

Cerverón V, Ferri FJ (2001) Another move toward the minimum consistent subset: a tabu search approach to the condensed nearest neighbour rule. IEEE Trans Syst Man Cybern B 31(3):408–413

Chien-Hsing C, Bo-Han K, Fu C (2006) The generalized condensed nearest neighbor rule as a data reduction method. In: Proceedings of the 18th international conference on pattern recognition. IEEE Computer Society, Hong-Kong, pp 556–559

Cover T, Hart P (1967) Nearest neighbor pattern classification. IEEE Trans Inf Theory 13:21–27

De Haro-García A, García-Pedrajas N (2009) A divide-and-conquer approach for scaling up instance sele ction algorithm. Data Min Knowl Discov 18:392–418

Devijver PA, Kittler J (1980) On the edited nearest neighbor rule. In: Proceedings of the 5th international conference on pattern recognition. Los Alamitos, CA, pp 72–80

Friedman JH, Bentley JL, Finkel RA (1997) An algorithm for finding best matches in logarithmic expected time. ACM Trans Math Softw 3(3):209–226

Garain U (2008) Prototype reduction using an artificial immune model. Pattern Anal Appl 11:353–363

García S, Cano JR, Herera F (2008) A memetic algorithm for evolutionary prototype selection: a scaling up approach. Pattern Recognit 41:2693–2709

Glover F (1986) The general employee scheduling problem: an integration of management science and artificial intelligence. Comput Oper Res 13(4):563–593

Grochowski M, Jankowski N (2004) Comparison of instance selection algorithms II. Results and comments. In: Rutkowski L et al (eds) ICAISC 2004, LNAI, Zacopane, Poland, pp 580–585

Hart PE (1968) The condensed nearest neighbor rule. IEEE Trans Inf Theory 14:515–516

Ke-Ping Z, Shui-Geng Z, Ji-Hong G, Ao-Ying A (2003) C-Pruner: An improved instance pruning algorithm. In: Proceedings of 2nd IEEE international conference on machine learning and cybernetics, vol 1. pp 94–99

Kittler J (1986) Feature selection and extraction. In: Young TY, Fu KS (eds) Handbook of pattern recognition and image processing. Academic Press, New York, pp 203–217

Kuncheva LI (1995) Editing for the *k*-nearest neighbors rule by a genetic algorithm. Pattern Recognit Lett 16:809–814

Kuncheva LI (1997) Fitness functions in editing k-NN referent set by genetic algorithms. Pattern Recognit 30:1041–1049

Kuncheva LI, Bezdek JC (1998) Nearest prototype classification, clustering, genetic algorithms, or random search? IEEE Trans Syst Man Cybern C 28(1):160–164

Liu H, Motoda H (2002) On issues of instance selection. Data Min Knowl Discov 6:115–130

Lumini A, Nanni L (2006) A clustering method for automatic biometric template selection. Pattern Recognit 39:495–497

Mollineda RA, Ferri FJ, Vidal E (2002) An efficient prototype merging strategy for the condensed 1-NN rule through class-conditional hierarchical clustering. Pattern Recognit 35:2771–2782

Narayan BL, Murthy CA, Pal SK (2006) Maxdiff kd-trees for data condensation. Pattern Recognit Lett 27: 187–200

Olvera-López JA, Carrasco-Ochoa JA, Martínez-Trinidad JF (2005) Sequential search for decremental edition. In: Gallagher M, Hogan J, Maire F (eds) LNCS 3578: IDEAL 2005. Queensland, Australia, pp 280–285

Olvera-López JA, Martínez-Trinidad JF, Carrasco-Ochoa JA (2007a) Restricted sequential floating search applied to object selection. In: Perner P (ed) MLDM 2007:LNAI 4571. Leipzig, Germany, pp 694–702

Olvera-López JA, Carrasco-Ochoa JA, and Martínez-Trinidad JF (2007b) Object selection based on clustering and border objects. In: Kurzynski M et al (eds) Computer recognition systems 2, ASC 45. Wroclaw, Poland, pp. 27–34

Olvera-López JA, Carrasco-Ochoa JA, Martínez-Trinidad JF (2008) Prototype selection via prototype relevance. In: Ruiz-Shulcloper J, Kropatsch WG (eds) CIARP 2008, LNCS 5197. Habana, Cuba, pp. 153–160

Olvera-López JA, Martínez-Trinidad JF, Carrasco-Ochoa JA, Kittler J (2009) Prototype selection based on sequeintial search. Intell Data Anal 13(4):599–631

Paredes R, Vidal E (2000) Weighting prototypes. A new editing approach. In: Proceedings of the international conference on pattern recognition ICPR, vol. 2. pp 25–28

Pudil P, Ferri FJ, Novovicová J, Kittler J (1994) Floating search methods for feature selection with nonmonotonic criterion functions. In: Proceedings of the 12th international conference on pattern recognition. IEEE Computer Society Press, pp 279–283

Raicharoen T, Lursinsap C (2005) A divide-and-conquer approach to the pairwise opposite class-nearest neighbor (POC-NN) algorithm. Pattern Recognit Lett 26(10):1554–1567

Ritter GL, Woodruff HB, Lowry SR, Isenhour TL (1975) An algorithm for a selective nearest neighbor decision rule. IEEE Trans Inf Theory 21(6):665–669

Riquelme JC, Aguilar-Ruíz JS, Toro M (2003) Finding representative patterns with ordered projections. Pattern Recognit 36:1009–1018

Srisawat A, Phienthrakul T, Kijsirikul B (2006) SV-kNNC: an algorithm for improving the efficency of k-Nearest neighbr. In: Yang Q, Webb G (eds) PRICAI 2006:LNAI 4099. Guilin, China, pp 975–979

Spillmann B, Neuhaus M, Bunke H, Pękalska E, Duin RPW (2006) Transforming strings to vector spaces using prototype selection. In: Yeung D-Y et al (eds) SSPR&SPR 2006, LNCS 4109. Hong-Kong, pp. 287–296

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

Vapnik V (1995) The nature of statistical learning theory. Springer, New York

Vázquez F, Sánchez S, Pla F (2005) A stochastic approach to Wilson's editing algorithm. In: Marques JS et al (eds) IbPRIA 2005, LNCS 3523. Estoril, Portugal, pp 35–42

Venmann CJ, Reinders MJT (2005) The nearest sub-class classifier: a compromise between the nearest mean and nearest neighbor classifier. IEEE Trans Pattern Anal Mach Intell 27(9):1417–1429

Venmann CJ, Reinders MJT, Backer E (2002) A maximum variance clustering algorithm. IEEE Trans Pattern Anal Mach Intell 24(9):1273–1280

Wilson DL (1972) Asymptotic properties of nearest neighbor rules using edited data. IEEE Trans Syst Man Cybern 2:408–421

Wilson DR, Martínez TR (2000) Reduction techniques for instance-based learning algorithms. Mach Learn 38:257–286

Yuangui L, Zhonhui H, Yunze C, Weidong Z (2005) Support vector based prototype selection method for nearest neighbor rules. In: Wang L et al (eds) ICNC 2005, LNCS 3610. Changsha, China, pp 528–535

Zhang H, Sun G (2002) Optimal reference subset selection for nearest neighbor classification by tabu search. Pattern Recognit 35:1481–1490