Intelligent Data Analysis 13 (2009) 599–631
DOI 10.3233/IDA-2009-0383
IOS Press

Prototype selection based on sequential search

J.A. Olvera-López[a,*], J. Fco. Martínez-Trinidad[a], J.A. Carrasco-Ochoa[a] and J. Kittler[b]
[a]*Computer Science Department, National Institute of Astrophysics, Optics and Electronics, Luis Enrique Erro No. 1, Sta. María Tonantzintla, Puebla, CP: 72840, Mexico*
[b]*Center for Vision, Speech and Signal Processing, University of Surrey, Guildford GU2 7XH, UK*

Abstract. In this paper, we propose and explore the use of the sequential search for solving the prototype selection problem since this kind of search has shown good performance for solving selection problems. We propose three prototype selection methods based on sequential search. The main goal of our methods is to reduce the training data without losing too much classification accuracy. Experiments and results are reported showing the effectiveness of the proposed methods and comparing their performance against other prototype selection methods.

Keywords: Prototype selection, quality training set, supervised classification, sequential search

1. Introduction

The supervised classifiers recognize unseen (i.e., unclassified) prototypes (objects) through an already labeled training set (T). Usually, all the available prototypes are used as training for the classifier; a sample pre-processing is not performed and as result, harmful and superfluous prototypes could be stored needlessly. Thus, what prototypes to store for using as training in order to avoid excessive storage and too long runtimes, and possibly to improve the classification accuracy by avoiding noise and over-fitting should be decided. Therefore, a good prototype set (S) that would ideally be of minimal cardinality and produce the highest possible classification accuracy is needed. Following [19,24], there are two strategies to find this prototype set:

Prototype Selection. A subset of prototypes from the original data set is retained, eliminating those that do not contribute significantly to the classification accuracy.

Prototype Replacement. A number of labeled prototypes that do not necessarily coincide with any prototype in the original data set are used instead of it.

In this paper, the prototype selection strategy is addressed. Many researchers have faced this problem, most of them have proposed methods which are designed to solve the prototype selection problem based on the *k-NN* (*k-Nearest Neighbor*) rule [29]. The prototype subsets produced by these methods have a good performance when they are used as training for *k-NN*. However, when other classifiers are used, the subsets obtained by these methods do not have as good performance as they have for *k-NN* (as we will show in this paper), therefore, prototype selection methods that allow using, during the selection stage, the classifier that will be used at the classification stage are needed.

[*]Corresponding author. E-mail: aolvera@ccc.inaoep.mx.

As it is well known, the sequential search is a useful strategy for solving selection problems; in this paper, we explore the use of this kind of search for prototype selection, guided by the accuracy of a classifier. We propose three prototype selection methods based on sequential search for reducing the stored prototype set trying to maintain acceptable recognition accuracy.

The paper is organized as follows. Section 2 describes the main related works. The proposed methods and experimental results are presented in Section 3. In Section 4, a discussion about the performance of our methods is presented. Finally, in Section 5, the conclusions and future research directions are given.

2. Related works

There has been a lot of research in developing efficient algorithms for prototype selection; some of these methods are briefly described in this section.

Hart [26] proposed the Condensed Nearest Neighbor method (CNN); it begins by randomly selecting one prototype from T and putting it in S. Then each prototype in T is classified using only the prototypes in S. If a prototype is misclassified, it is added to S, to ensure that it will be correctly classified. This process is repeated until there are not prototypes in T that are misclassified. This method ensures that S correctly classifies all prototypes in T, this is, S is consistent. CNN does not guarantee to find a minimal consistent subset.

The Reduced Nearest Neighbor rule (RNN) [15] consists in deleting from T those prototypes that do not affect the classification of the remaining prototypes, the goal is to find a minimal consistent subset.

The Selective Nearest Neighbor rule (SNN) [14] finds a minimal subset such that all prototypes in T would be closer to a prototype of the same class in S than to any prototype of a different class in T.

Wilson [10] proposed a method for prototype selection through the Edited Nearest Neighbor (ENN), here a prototype is deleted from T if its class does not coincide with the class of the majority of its k nearest neighbors. A variant of this method is the Repeated ENN (RENN) where ENN is repeatedly applied until all remaining prototypes have the majority of their nearest neighbors with the same class.

Tomek [18] extended the ENN with his *All k-NN* prototype selection method. This method works as follows: for $i = 1$ to k, flag as bad any prototype misclassified by its i nearest neighbors. After completing the loop all k times, remove any prototypes flagged as bad.

Devijver and Kittler [25] proposed the *Multiedit* method for prototype selection, which randomly divides T in m blocks ($P_1 \ldots P_m$). After that, ENN (using 1-NN) is applied over each block P_i finding the neighbors of P_i in $P_{(i+1) \mathrm{mod}\ m}$. This process is repeated until there are not changes (eliminations) in f successive iterations.

Lowe [9] presented a Variable Similarity Metric (VSM) learning system that produces a confidence level of its classifications. In order to reduce storage and remove noisy prototypes, a prototype p is removed if all of its k nearest neighbors are of the same class, even if they are of a different class than p (in which case p is likely to be noisy).

Kuncheva [22–24] used genetic algorithms (GA) to select the prototype subset S, codifying a selection as a binary valued chromosome and using a fitness function based on the classification accuracy and the size of S.

Wilson and Martinez [12] presented five methods DROP1,…, DROP5 (Decremental Reduction Optimization Procedure) for prototype selection. DROP1 is identical to RNN but the classification accuracy is verified in S instead of T. This algorithm uses the next selection rule: delete the prototype p if the associates of p are correctly classified without p. An associate is a prototype that has p as one of its nearest neighbors. *DROP2* takes into account the effect in T of eliminating a prototype in the partial subset, i.e., a prototype p is deleted only if its associates in T are correctly classified without p. DROP3 uses a filter similar to ENN and after applies DROP2. DROP4 is similar to DROP3 but it uses a different filter. DROP5 is based in DROP2 but it starts deleting the prototypes that are nearest to their nearest enemies, i.e., the nearest prototypes but with different class.

Another strategy used for solving search problems is the tabu search (TS) [13]. Cerverón and Ferri [31] proposed a method based on TS for prototype selection.

Brighton and Mellish [16] proposed the ICF (Iterative Case Filtering) method based on the *Coverage* (associates) and *Reachable* (neighborhood) sets, a prototype p is flagged for removal if $|Reachable(p)| > |Coverage(p)|$, which means that more cases can solve p than p can solve itself. After, all prototypes flagged for removal are deleted.

Riquelme et al. [20] proposed the POP (Pattern by Ordered Projections) method based on the *weakness* concept, which is defined as the number of times that a prototype is not a border, that is, the prototype is not close to prototypes belonging to different class. POP discards those prototypes whose *weakness* is equal to the number of attributes. Aguilar et al. [28] modified this method using a threshold weakness factor.

The POC-NN (Pair Opposite Class-Nearest Neighbor) method proposed by Raicharoen and Lursinsap [30] selects border prototypes (close to prototypes belonging to different class). The selection process in POC-NN is based on the nearest prototypes to the mean of opposite classes.

A variant of CNN is the Generalized Condensed Nearest Neighobor (GCNN) method [5] which is similar to CNN but GCNN includes in S prototypes according to the *Absorption*(p) criterion, which is calculated in terms of the nearest neighbor and the nearest enemy of p in S. The selection process finishes when all prototypes in T have been strongly absorbed, that is, when their *Absorption* satisfies a threshold value given by the user.

Some authors [3,17,19] mentioned the idea of using clustering for prototype selection; this idea consists in after splitting T in n clusters, S is the set of centers of each cluster. In [2], the CLU (CLU*stering*) prototype selection method is based on this rule and it was applied to the signature recognition problem. Another method that follows the same idea is NSB (*Nearest Sub-class Classifier*) [7] which allows selecting different number of prototypes (centers) per class via the Maximum Variance Cluster algorithm [6].

Most of the prototype selection methods previously discussed (excluding GA, TS, CLU and NSB) are based on the *k-NN* rule. In the next section, we propose prototype selection methods based on sequential search, which allow using, during the selection stage, the classifier that will be used in the classification stage.

3. Proposed methods

In this section, three new prototype selection methods based on sequential search that can use any classifier in the selection process are introduced.

3.1. Backward sequential prototype selection

The first prototype selection method we propose is called Backward Sequential Prototype Selection (BSPS). The idea of this method consists in adapting the backward sequential selection used for feature selection (a treatment of this method can be found in [21]) but now for prototype selection.

Given a training set T, frequently occurs that there are some prototypes in T with a null or even negative contribution for classification accuracy, so it is necessary to identify and eliminate those prototypes i.e. prototype selection. We can see this problem as a search problem where the goal is to find the optimal training subset of prototypes S for a classifier. If we have m prototypes, the size of the search space is 2^m-1 then it is necessary to have a non-exhaustive search method for avoiding exponential complexity. The sequential search is a non exhaustive method with quadratic complexity ($O(m^2)$) that allows to get a suboptimal training set.

In order to evaluate the subsets of prototypes along the search, we propose to apply a classifier (the evaluation function in the BSPS method) using the subset that is being evaluated as training set and estimating the accuracy rate over a test set.

It is important to highlight that most of the prototype selection methods are much related to the nearest neighbor rule. Our method discards a prototype p if the accuracy (over the testing set) of the classifier (anyone) after discarding p is the same or better than without eliminating p. Notice that, in this way, the prototype selection depends on the classifier accuracy and not necessarily on the neighbors' relationship.

The main idea of BSPS is to search the best single prototype elimination, and repeating this process until no more prototypes can be eliminated.

The proposed method starts with the whole set T and in each step, the worst prototype is discarded. In this case, as worst prototype p we mean the one that if we eliminate it from T the resultant subset $S = T–\{p\}$ provides a classification accuracy better than or equal to T does and better than or equal to eliminating any other prototype. If there is more than one worst prototype, only the last one is eliminated.

If we have m prototypes and F attributes, in each step, the complexity of removing the worst prototype in the training set is $O(m)$ and at most $m - 1$ prototypes can be removed. Therefore the complexity of BSPS is $O(m^2 J(m, F))$, where $J(m, F)$ is the complexity of the evaluation function (classifier) along the search.

The BSPS method is as follows:

```
BSPS(Training set: T): Prototype subset S
S = T
BestEval=Classifier(S)
Repeat
  Worstp = None
  For each prototype $p$ in S
    S'= S – $\{p\}$
    Eval = Classifier(S')
    If Eval $\geqslant$ BestEval
      Then Worstp = $p$
        BestEval = Eval
  If Worstp $\neq$ None
    Then S = S – {Worstp}
Until Worstp == None or |S| == 1
Return S
```

Our method stops when it cannot eliminate more prototypes and its behavior depends on the function that evaluates the subset quality (which is based on a classifier).

3.1.1. Experiment description

From the UCI Repository [1] nine datasets were chosen in order to study the behavior of the methods introduced in this document.

In order to handle mixed databases, those with both numeric and nominal attributes, we used the heterogeneous distance function HVDM [11], which is defined as follows:

$$HVDM(x,y) = \sqrt{\sum_{a=1}^{F} d_a^2(x,y)} \tag{1}$$

where $d_a(x,y)$ is the distance for the values of the feature a and it is as follows:

$$d_a(x,y) = \begin{cases} 1 & \text{if } x \text{ or } y \text{ is unknown} \\ vdm_a(x,y) & \text{if } a \text{ is nominal} \\ \frac{|x-y|}{4\sigma_a} & \text{if } a \text{ is numeric} \end{cases} \tag{2}$$

where σ_a is the standard deviation in T of the feature a and *vdm*$_a(x,y)$ is defined as:

$$vdm_a(x,y) = \sum_{c=1}^{C} \left(\frac{N_{a,x,c}}{N_{a,x}} - \frac{N_{a,y,c}}{N_{a,y}} \right)^2 \tag{3}$$

Where $N_{a,x}$ is the number of times that the feature a had value x in the training set; $N_{a,x,c}$ is the number of times that the feature a had value x in the class c; and C is the number of classes.

For all the experiments shown in this document, 10-fold cross-validation was used. Each dataset was divided into 10 blocks. A training set T consisting of nine of the blocks (i.e., 90% of the data) was used to train each prototype selection method (for BSPS and TS we used the whole training set, 90% of the data, as testing set during the selection process), from which each method returned a subset S. The remaining block (i.e., the other 10% of the data) was classified using only the prototypes in S. For each dataset, with each prototype selection method 10 trials were run. The average classification accuracy (*Acc.*) over the 10 trials for each method on each dataset is reported. The average percentage of prototypes in T that were included in S is reported for each experiment under the column "%", that is, % = 100*|S|/|T|. In addition, the accuracy obtained using the whole training set (*Orig.*) is included in the tables, for comparison. Also at the bottom of each table, the averages over the nine datasets are shown.

It is important to emphasize that in all the experiments for each prototype selection method; the same training/testing sets and distance function (HVDM) were used on the same computer.

3.1.2. Comparison of BSPS against other prototype selection methods

In this section, a comparison among BSPS, ICF, DROPs, GCNN, TS, POP and POC-NN prototype selection methods using *k-NN* ($k = 3$, value where DROPs have the best performance), is shown (Tables 1–4). We have compared against DROPs because according to [12], they outperforms to previously proposed prototype selection methods. In addition, we have considered other recently proposed methods (ICF, TS, POP, POC-NN and GCNN).We have omitted GA because according to the results reported in [19], TS outperforms GA.

The POP and POC-NN methods cannot be applied over all the datasets because some of them have nominal or missing values. Therefore, in Tables 3–4 we show the results obtained over the five datasets that can be used by POP and POC-NN.

In Fig. 1, we show the scatter graphic of retention (vertical axis) versus accuracy (horizontal axis) for the average results obtained by the methods shown in Tables 3 and 4.

Table 1
Accuracy (*Acc.*) and retention (%) results obtained by: *Orig.*, *DROPs* and *BSPS* using *k-NN*, $k = 3$

Dataset	Orig.		DROP1		DROP2		DROP3		DROP4		DROP5		BSPS	
	Acc.	%	*Acc.*	%	*Acc.*	%	*Acc.*	%	*Acc.*	%	*Acc.*	%	*Acc.*	%
Bridges	66.09	100	39.91	25.17	61.51	28.53	56.36	14.78	56.73	21.08	62.82	20.66	60.18	58.80
Echocardiogram	95.71	100	88.93	12.16	93.04	15.16	92.86	13.95	92.86	13.80	88.75	14.87	91.96	25.18
Glass	71.42	100	66.84	23.21	65.89	31.05	66.28	24.35	67.77	29.39	62.16	25.91	65.02	46.36
Iris	94.66	100	89.33	9.56	94.66	16.96	95.33	15.33	94.66	15.26	94.00	12.44	94.00	15.26
Liver	65.22	100	57.98	31.53	64.08	39.26	67.82	26.83	66.41	31.95	63.46	30.59	57.85	58.90
New Thyroid	95.45	100	85.56	8.32	90.78	14.88	93.98	9.77	93.51	10.39	94.46	8.84	91.25	14.09
Tae	51.08	100	55.75	30.54	53.67	33.56	49.75	26.27	53.71	31.71	52.42	32.01	51.54	54.01
Wine	94.44	100	94.97	10.36	95.52	14.80	94.41	15.04	94.41	15.04	93.86	10.55	94.44	13.76
Zoo	93.33	100	87.78	16.79	86.67	20.37	90.00	20.37	91.11	21.36	95.56	18.77	91.55	20.86
Average	**80.60**	**100**	**74.12**	**18.63**	**78.42**	**23.84**	**78.53**	**18.52**	**79.02**	**21.08**	**78.61**	**19.40**	**77.53**	**34.14**

Table 2
Accuracy (*Acc.*) and retention (%) results obtained by: *Orig.*, *TS*, *ICF*, *GCNN* and *BSPS* using *k-NN*, $k = 3$

Dataset	Orig.		TS		ICF		GCNN		BSPS	
	Acc.	%	*Acc.*	%	*Acc.*	%	*Acc.*	%	*Acc.*	%
Bridges	66.09	100	45.90	18.94	40.45	8.33	68.20	88.20	60.18	58.80
Echocardiogram	95.71	100	85.71	7.46	75.00	11.94	93.39	22.67	91.96	25.18
Glass	71.42	100	62.59	15.98	63.63	8.93	69.61	61.62	65.02	46.36
Iris	94.66	100	70.66	6.50	64.66	10.37	96.00	38.00	94.00	15.26
Liver	65.22	100	64.13	5.21	52.94	1.83	66.09	83.70	57.85	58.90
New Thyroid	95.45	100	83.05	5.40	76.23	3.61	93.96	30.33	91.25	14.09
Tae	51.08	100	55.00	14.93	40.00	2.86	45.95	32.50	51.54	54.01
Wine	94.44	100	79.44	6.10	59.01	5.49	94.44	78.89	94.44	13.76
Zoo	93.33	100	88.88	14.12	75.55	9.38	95.55	26.17	91.55	20.86
Average	**80.60**	**100**	**70.60**	**10.52**	**60.83**	**6.97**	**80.35**	**51.34**	**77.53**	**34.14**

Table 3
Results obtained by: *Orig.*, *DROPs* and *BSPS* for numeric datasets using *k-NN*, $k = 3$

Dataset	Orig.		DROP1		DROP2		DROP3		DROP4		DROP5		BSPS	
	Acc.	%	*Acc.*	%	*Acc.*	%	*Acc.*	%	*Acc.*	%	*Acc.*	%	*Acc.*	%
Glass	71.42	100	66.84	23.21	65.89	31.05	66.28	24.35	67.77	29.39	62.16	25.91	65.02	46.36
Iris	94.66	100	89.93	9.56	94.67	16.96	95.33	15.33	94.67	15.26	94.00	12.44	94.00	15.26
Liver	65.22	100	57.98	31.53	64.08	39.26	67.82	26.83	66.41	31.65	63.46	30.59	57.85	58.90
New Thyroid	95.45	100	85.56	8.32	90.78	14.88	93.98	9.77	93.51	10.39	94.46	8.84	91.25	14.09
Wine	94.44	100	94.97	10.36	95.52	14.80	94.41	15.04	94.41	15.04	93.86	10.55	94.44	13.76
Average	**84.24**	**100**	**79.06**	**16.60**	**82.19**	**23.39**	**83.56**	**18.26**	**83.35**	**20.35**	**81.59**	**17.67**	**80.51**	**29.67**

From the results in Tables 1–4 and Fig. 1, we can see that in the average case, the classification accuracy obtained by BSPS is better than the obtained by DROP1, TS, ICF, POP and POC-NN. The best method in this experiment was GCNN.

We evaluated the accuracy changing of the different subsets obtained during the BSPS selection process, that is, we evaluated some subsets saved during the selection process. For this experiment, the datasets were divided in three partitions, one for training, the second for testing during the selection process and the last for evaluating the prototype sets obtained. The results obtained are depicted in Figs 2–3, the subset size (horizontal axis) varies from the original training set and the size of the final subset selected by BSPS.

Figure 2 shows how the accuracy over the testing set (second partition) changes during the selection.

Table 4
Results obtained by: *Orig.*, *TS*, *ICF*, *POP*, *POC-NN*, *GCNN* and *BSPS* for numeric datasets using *k-NN*, $k = 3$

Dataset	Orig.		TS		ICF		POP		POC-NN		GCNN		BSPS	
	Acc.	%	*Acc.*	%	*Acc.*	%	*Acc.*	%	*Acc.*	%	*Acc.*	%	*Acc.*	%
Glass	71.42	100	62.59	15.98	63.63	8.93	59.09	29.17	71.47	57.42	69.61	61.62	65.02	46.36
Iris	94.66	100	70.66	6.50	64.66	10.37	79.33	16.00	77.33	12.88	96.00	38.00	94.00	15.26
Liver	65.22	100	64.13	5.21	52.94	1.83	55.88	7.08	55.68	58.09	66.09	83.70	57.85	58.90
New Thyroid	95.45	100	83.05	5.40	76.23	3.61	89.22	13.90	93.05	15.34	93.96	30.33	91.25	14.09
Wine	94.44	100	79.44	6.10	59.01	5.49	92.18	34.51	94.93	40.82	94.44	78.89	94.44	13.76
Average	**84.24**	**100**	**71.97**	**7.84**	**63.29**	**6.05**	**75.14**	**20.13**	**78.49**	**36.91**	**84.02**	**58.51**	**80.51**	**29.67**

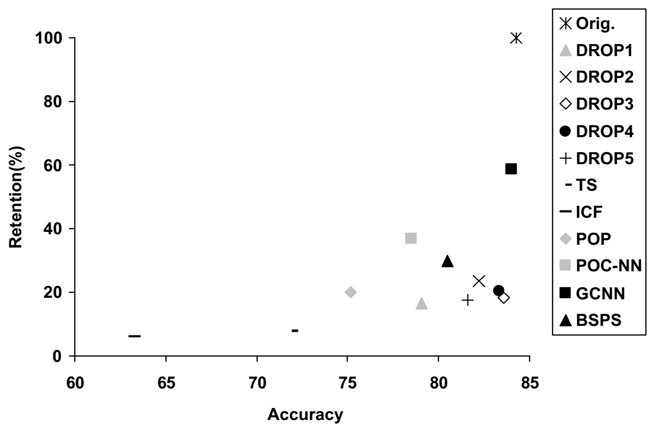

Fig. 1. Scatter graphic from average results shown in Tables 3 and 4.

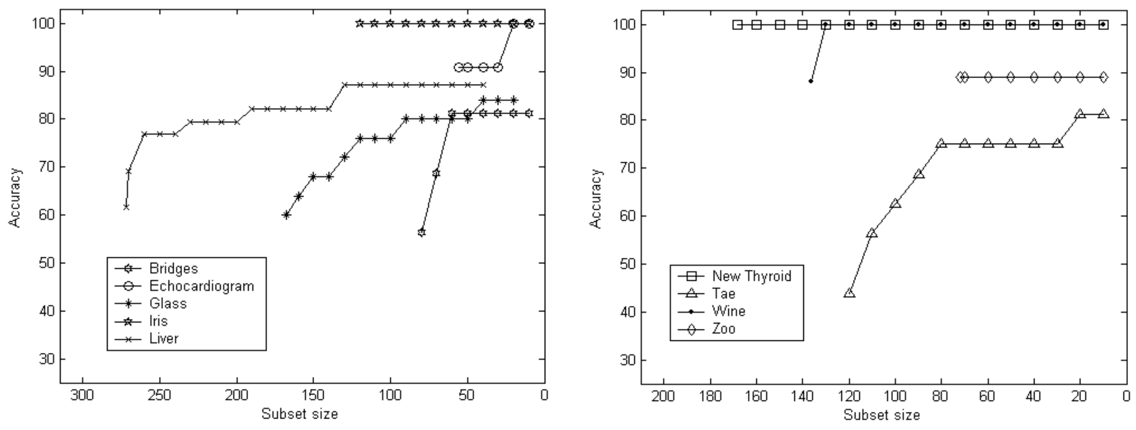

Fig. 2. Accuracy obtained by the different subsets using the second partition as testing set.

This figure shows that BSPS has an incremental behavior during the selection process since only prototypes that contribute for obtaining better classification (according to the testing set) are selected.

Figure 3 shows the accuracy of the subsets obtained during the selection but over the evaluation set (third partition) unseen during the selection. In these results, BSPS has not an incremental behavior because the accuracy obtained by the different subsets is evaluated over the unseen testing set.

Table 5
Accuracy (*Acc.*) and retention (%) results obtained by: *Orig.*, *DROPs* and *BSPS* using *k-NN*, $k = 1$

Dataset	Orig.		DROP1		DROP2		DROP3		DROP4		DROP5		BSPS	
	Acc.	%	*Acc.*	%	*Acc.*	%	*Acc.*	%	*Acc.*	%	*Acc.*	%	*Acc.*	%
Bridges	64.00	100	73.18	23.16	49.73	26.32	44.36	14.78	40.05	21.05	48.36	20.66	53.25	50.12
Echocardiogram	86.78	100	83.11	8.56	88.64	17.43	90.07	13.82	88.64	18.01	90.57	7.65	88.64	20.98
Glass	70.50	100	62.60	17.96	68.29	23.78	68.64	19.78	67.27	22.38	69.59	19.38	67.22	41.74
Iris	94.00	100	82.67	5.41	95.33	14.00	96.00	11.11	96.00	11.63	94.00	8.44	94.00	13.11
Liver	65.24	100	58.26	22.83	62.58	28.60	60.58	24.83	63.72	27.15	64.04	26.09	62.02	54.61
New Thyroid	95.80	100	83.87	7.85	92.14	12.30	92.62	11.42	92.14	11.73	93.53	8.53	93.07	13.12
Tae	58.29	100	45.71	39.00	47.17	41.35	45.75	36.57	47.75	38.56	51.71	38.12	59.62	49.52
Wine	94.93	100	91.11	7.30	94.41	11.86	95.52	11.05	95.52	11.24	94.97	6.18	94.96	11.67
Zoo	98.89	100	90.00	16.91	90.00	18.64	90.00	18.64	90.00	18.64	95.56	16.67	93.33	24.07
Average	**80.94**	**100**	**71.17**	**16.55**	**76.48**	**21.59**	**75.95**	**18.00**	**75.68**	**20.04**	**78.04**	**16.86**	**78.46**	**30.99**

Table 6
Accuracy (*Acc.*) and retention (%) results obtained by: *Orig.*, *TS*, *ICF*, *GCNN* and *BSPS* using *k-NN*, $k = 1$

Dataset	Orig.		TS		ICF		GCNN		BSPS	
	Acc.	%	*Acc.*	%	*Acc.*	%	*Acc.*	%	*Acc.*	%
Bridges	64.00	100	27.00	23.37	38.36	22.21	47.00	91.92	53.25	50.12
Echocardiogram	86.78	100	88.92	7.67	96.21	41.75	93.39	27.62	88.64	20.98
Glass	70.50	100	68.16	10.95	68.39	32.91	72.38	61.16	67.22	41.74
Iris	94.00	100	94.66	5.92	94.00	45.03	95.33	36.22	94.00	13.11
Liver	65.24	100	70.44	2.05	59.68	27.63	64.94	84.05	62.02	54.61
New Thyroid	95.80	100	93.96	3.82	92.05	53.22	95.36	30.28	93.07	13.12
Tae	58.29	100	58.20	9.19	56.25	5.59	51.11	45.55	59.62	49.52
Wine	94.93	100	94.93	4.80	89.82	44.44	94.96	79.02	94.96	11.67
Zoo	98.89	100	93.33	11.23	81.22	16.54	98.89	26.04	93.33	24.07
Average	**80.49**	**100**	**79.55**	**8.83**	**76.04**	**29.85**	**79.25**	**53.54**	**78.46**	**30.99**

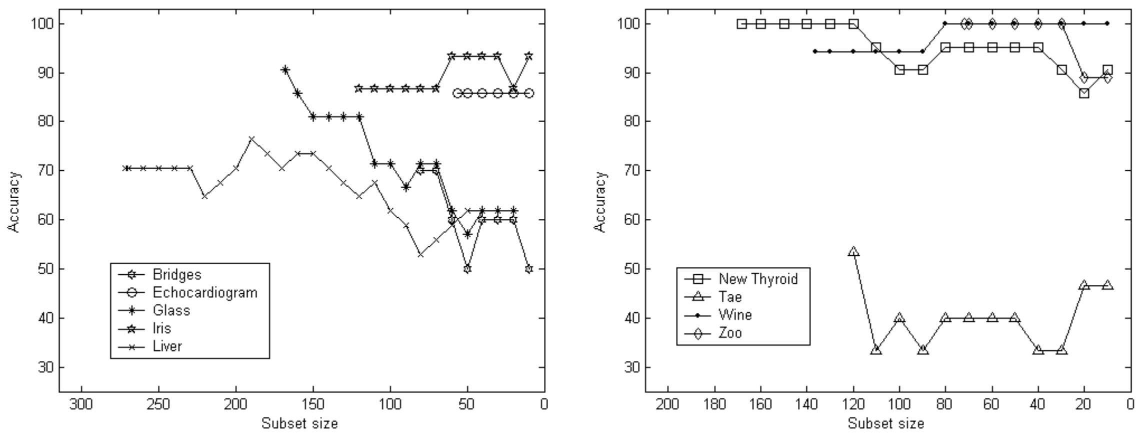

Fig. 3. Accuracy obtained by the different subsets using the unseen (third partition) as testing set.

In the previous experiments, the value for *k-NN* was $k = 3$ but it is important to know the performance of the prototype selection methods varying the k's value, so we report results obtained using $k = 1$ (Tables 5–8, Fig. 4) and $k = 5$ (Tables 9–12, Fig. 5).

Table 7
Results obtained by: *Orig.*, *DROPs* and *BSPS* for numeric datasets using *k-NN*, $k = 1$

Dataset	Orig.		DROP1		DROP2		DROP3		DROP4		DROP5		BSPS	
	Acc.	%	*Acc.*	%	*Acc.*	%	*Acc.*	%	*Acc.*	%	*Acc.*	%	*Acc.*	%
Glass	70.50	100	62.60	17.96	68.29	23.78	68.64	19.78	67.27	22.38	69.59	19.38	67.22	41.74
Iris	94.00	100	82.67	5.41	95.33	14.00	96.00	11.11	96.00	11.63	94.00	8.44	94.00	13.11
Liver	65.24	100	58.26	22.83	62.58	28.60	60.58	24.83	63.72	27.15	64.04	26.09	62.02	54.61
New Thyroid	95.80	100	83.87	7.85	92.14	12.30	92.62	11.42	92.14	11.73	93.53	8.53	93.07	13.12
Wine	94.93	100	91.11	7.30	94.41	11.86	95.52	11.05	95.52	11.24	94.97	6.18	94.96	11.67
Average	**84.09**	**100**	**75.70**	**12.27**	**82.55**	**18.11**	**82.67**	**15.64**	**82.93**	**16.83**	**83.23**	**13.72**	**82.25**	**26.85**

Table 8
Results obtained by: *Orig.*, *TS*, *ICF*, *POP*, *POC-NN*, *GCNN* and *BSPS* for numeric datasets using *k-NN*, $k = 1$

Dataset	Orig.		TS		ICF		POP		POC-NN		GCNN		BSPS	
	Acc.	%	*Acc.*	%	*Acc.*	%	*Acc.*	%	*Acc.*	%	*Acc.*	%	*Acc.*	%
Glass	70.50	100	68.16	10.95	68.39	32.91	59.09	47.18	71.88	57.42	72.38	61.16	67.22	41.74
Iris	94.00	100	94.66	5.92	94.00	45.03	84.66	16.00	86.00	12.88	95.33	36.22	94.00	13.11
Liver	65.24	100	70.44	2.05	59.68	27.63	58.82	65.24	55.13	58.09	64.94	84.05	62.02	54.61
New Thyroid	95.80	100	93.96	3.82	92.05	53.22	87.98	13.90	93.03	15.34	95.36	30.28	93.07	13.12
Wine	94.93	100	94.93	4.80	89.82	44.44	93.85	34.51	93.23	40.82	94.96	79.02	94.96	11.67
Average	**84.09**	**100**	**84.43**	**5.51**	**80.79**	**40.65**	**76.88**	**35.37**	**79.85**	**36.91**	**84.59**	**58.15**	**82.25**	**26.85**

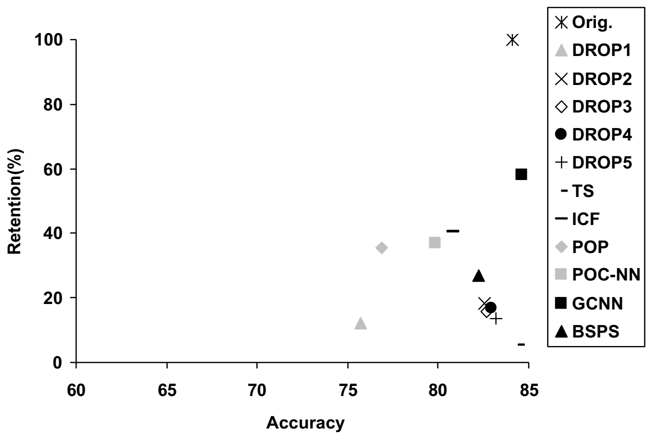

Fig. 4. Scatter graphic from average results shown in Tables 7 and 8.

According to the results shown in Tables 1–12, for $k = 3$ and $k = 5$, the method that obtained the lowest accuracy was ICF which had good performance for $k = 1$. The performance of the other methods was only slightly affected varying the value of k. For $k = 1$ and 3, in the average case for all the datasets (Tables 1–2, 5–6), the best methods were TS and GCNN respectively, and for $k = 5$ (Tables 9–10) the best method was DROP2 followed by BSPS.

BSPS is a prototype selection method which uses a classifier as evaluation function in the internal selection process. In the above experiments we used *k-NN* as evaluation function, but in order to show the performance of the prototype selection obtained for other classifiers, we carried out some experiments using the subsets obtained by the prototype selection methods in the previous experiments (using *k-NN*, $k = 3$) as training for: LWLR (Locally Weighted-Linear Regression) [4] which is an instance based classifier, SVM (Support Vector Machines) [32] and feedforward backpropagation Neural Networks [8],

Table 9
Accuracy (*Acc.*) and retention (%) results obtained by: *Orig.*, *DROPs* and *BSPS* using *k-NN*, $k = 5$

Dataset	Orig.		DROP1		DROP2		DROP3		DROP4		DROP5		BSPS	
	Acc.	%	*Acc.*	%	*Acc.*	%	*Acc.*	%	*Acc.*	%	*Acc.*	%	*Acc.*	%
Bridges	64.18	100	64.18	27.37	65.73	31.58	61.91	12.67	62.27	23.16	62.55	25.26	60.00	62.10
Echocardiogram	93.21	100	92.50	14.93	92.89	19.70	92.89	18.18	91.71	18.18	90.70	19.40	93.21	20.42
Glass	65.41	100	60.63	28.14	65.35	33.80	61.10	19.42	63.51	29.07	59.72	28.66	61.30	46.00
Iris	93.33	100	91.67	9.04	93.00	15.48	93.33	14.30	93.33	14.37	91.67	11.78	93.33	16.82
Liver	66.37	100	65.17	28.86	65.77	38.20	66.61	23.8	64.60	32.82	68.70	31.18	67.63	59.48
New Thyroid	93.54	100	86.02	13.08	93.51	18.34	92.62	13.59	91.62	16.74	90.69	14.11	90.77	15.07
Tae	47.79	100	47.17	38.27	43.13	44.00	37.83	26.71	45.13	35.76	41.08	32.60	41.87	53.64
Wine	94.33	100	90.08	24.97	94.33	20.84	91.71	14.92	91.33	23.78	93.61	20.97	94.33	16.53
Zoo	93.33	100	91.11	26.42	91.11	28.52	87.78	24.81	90.00	27.53	93.33	25.56	93.33	24.69
Average	**79.05**	**100**	**76.50**	**23.45**	**78.31**	**27.83**	**76.20**	**18.71**	**77.06**	**24.60**	**76.89**	**23.28**	**77.31**	**34.97**

Table 10
Accuracy (*Acc.*) and retention (%) results obtained by: *Orig.*, *TS*, *ICF*, *GCNN* and *BSPS* using *k-NN*, $k = 5$

Dataset	Orig.		TS		ICF		GCNN		BSPS	
	Acc.	%	*Acc.*	%	*Acc.*	%	*Acc.*	%	*Acc.*	%
Bridges	64.18	100	48.84	14.73	41.36	9.47	50.63	80.45	60.00	62.10
Echocardiogram	93.21	100	82.32	12.12	75.00	13.63	90.89	30.26	93.21	20.42
Glass	65.41	100	50.90	13.19	57.14	3.99	58.39	60.85	61.30	46.00
Iris	93.33	100	68.66	7.92	80.00	7.11	75.33	37.25	93.33	16.82
Liver	66.37	100	69.85	9.11	51.42	2.44	57.94	83.47	67.63	59.48
New Thyroid	93.54	100	92.66	7.23	81.81	4.29	93.98	30.64	90.77	15.07
Tae	47.79	100	42.00	19.93	40.20	6.91	46.25	34.42	41.87	53.64
Wine	94.33	100	90.49	9.04	74.70	4.68	94.33	78.71	94.33	16.53
Zoo	93.33	100	85.18	16.46	66.66	10.24	71.11	26.29	93.33	24.69
Average	**79.05**	**100**	**70.10**	**12.19**	**63.14**	**6.97**	**70.98**	**51.37**	**77.31**	**34.97**

Table 11
Results obtained by: *Orig.*, *DROPs* and *BSPS* for numeric datasets using *k-NN*, $k = 5$

Dataset	Orig.		DROP1		DROP2		DROP3		DROP4		DROP5		BSPS	
	Acc.	%	*Acc.*	%	*Acc.*	%	*Acc.*	%	*Acc.*	%	*Acc.*	%	*Acc.*	%
Glass	65.41	100	60.63	28.14	65.35	33.80	61.10	19.42	63.51	29.07	59.72	28.66	61.30	46.00
Iris	93.33	100	91.67	9.04	93.00	15.48	93.33	14.30	93.33	14.37	91.67	11.78	93.33	16.82
Liver	66.37	100	65.17	28.86	65.77	38.20	66.61	23.8	64.60	32.82	68.70	31.18	67.63	59.48
New Thyroid	93.54	100	86.02	13.08	93.51	18.34	92.62	13.59	91.62	16.74	90.69	14.11	90.77	15.07
Wine	94.33	100	90.08	24.97	94.33	20.84	91.71	14.92	91.33	23.78	93.61	20.97	94.33	16.53
Average	**82.60**	**100**	**78.71**	**20.82**	**82.39**	**25.33**	**81.07**	**17.21**	**80.88**	**23.36**	**80.88**	**21.34**	**81.47**	**30.78**

two non instance based classifiers.

In Tables 13–15 and Figs 6–8 the results are shown. In this experiment we have tested only numeric datasets because LWLR, SVM and Neural Networks are restricted to this kind of data. For each dataset, the accuracy results obtained by the original training set (*Orig.*) and BSPS are reported, as well as the retention results (%). According to results reported in Table 13 and Fig. 6, using LWLR, BSPS had a good behavior since GCNN and BSPS obtained the best accuracy.

Based on Table 14 and Fig. 7, the best accuracy results using SVM was obtained by GCNN.

For Neural Networks (Table 15 and Fig. 8), in the average case, the best method in accuracy was GCNN. For this classifier, BSPS was better than the other methods (excluding GCNN).

Almost all methods tested in Tables 13–15 are based on the *k-NN* rule but TS and BSPS allow us to

Table 12
Results obtained by: *Orig., TS, ICF, POP, POC-NN, GCNN* and *BSPS* for numeric datasets using *k-NN*, $k = 5$

Dataset	Orig.		TS		ICF		POP		POC-NN		GCNN		BSPS	
	Acc.	%	*Acc.*	%	*Acc.*	%	*Acc.*	%	*Acc.*	%	*Acc.*	%	*Acc.*	%
Glass	65.41	100	50.90	13.19	57.14	3.99	54.54	29.17	62.12	57.42	58.39	60.85	61.30	46.00
Iris	93.33	100	68.66	7.92	80.00	7.11	77.33	16.00	86.66	12.88	75.33	37.25	93.33	16.82
Liver	66.37	100	69.85	9.11	51.42	2.44	60.00	61.46	54.49	58.09	57.94	83.47	67.63	59.48
New Thyroid	93.54	100	92.66	7.23	81.81	4.29	87.42	13.90	93.46	15.34	93.98	30.64	90.77	15.07
Wine	94.33	100	90.49	9.04	74.70	4.68	91.07	34.51	94.33	40.82	94.33	78.71	94.33	16.53
Average	**82.60**	**100**	**74.51**	**9.30**	**69.01**	**4.50**	**74.07**	**31.01**	**78.21**	**36.91**	**75.99**	**58.18**	**81.47**	**30.78**

Table 13
Accuracy results obtained using *LWLR*

Dataset	Orig.	DROP1	DROP2	DROP3	DROP4	DROP5	TS	ICF	POP	POC-NN	GCNN	BSPS
Glass	57.85	46.58	48.48	51.66	53.18	54.06	50.86	53.56	53.48	50.41	56.88	54.54
Iris	98.00	82.66	93.33	92.00	92.00	92.00	93.33	91.33	87.33	85.33	95.33	93.33
Liver	70.13	68.63	68.68	68.63	67.51	68.95	68.40	68.66	63.62	68.68	70.13	68.68
New Thyroid	91.16	77.83	78.28	75.43	78.88	79.18	79.15	77.27	73.85	64.02	76.19	79.15
Wine	92.15	80.26	80.65	78.53	78.23	83.63	80.93	85.88	78.59	86.40	92.15	83.26
Average	**81.86**	**71.19**	**73.88**	**73.25**	**73.96**	**75.56**	**74.53**	**75.34**	**71.37**	**70.97**	**78.14**	**75.79**

Table 14
Accuracy results obtained using *SVM*

Dataset	Orig.	DROP1	DROP2	DROP3	DROP4	DROP5	TS	ICF	POP	POC-NN	GCNN	BSPS
Glass	65.34	58.72	60.17	62.48	62.03	56.99	59.09	59.97	54.93	57.46	66.80	61.90
Iris	96.00	93.33	95.33	94.00	93.33	95.33	94.66	96.00	85.33	94.00	94.66	94.00
Liver	69.91	68.26	68.26	68.26	67.97	68.56	69.73	67.97	60.10	68.56	69.73	69.73
New Thyroid	93.54	90.47	90.90	91.19	90.90	93.07	88.21	91.73	90.72	93.54	89.78	89.93
Wine	97.18	91.63	93.82	95.33	95.52	91.01	83.26	92.96	97.18	91.01	94.82	92.15
Average	**84.39**	**80.48**	**81.70**	**82.25**	**81.95**	**80.99**	**78.99**	**81.73**	**77.65**	**80.91**	**83.16**	**81.54**

Table 15
Accuracy results obtained using *Neural Networks*

Dataset	Orig.	DROP1	DROP2	DROP3	DROP4	DROP5	TS	ICF	POP	POC-NN	GCNN	BSPS
Glass	60.79	45.28	50.88	55.17	49.89	49.95	57.00	38.09	34.95	45.69	46.66	59.00
Iris	96.00	80.66	84.00	90.00	90.66	80.66	72.59	95.33	79.33	72.00	91.33	83.33
Liver	72.45	62.36	65.42	63.51	65.51	65.26	64.16	61.11	55.07	67.25	67.69	65.42
New Thyroid	94.37	74.97	84.45	86.16	81.88	82.68	76.25	84.02	86.40	93.05	89.23	87.01
Wine	96.63	83.66	89.83	88.13	84.83	89.37	76.94	84.37	84.83	84.83	96.63	88.88
Average	**84.05**	**69.39**	**74.92**	**76.59**	**74.55**	**73.58**	**69.39**	**72.58**	**68.12**	**72.56**	**78.31**	**76.73**

use any classifier during the selection process. Therefore, some experiments with TS and BSPS using other classifiers were done. In Table 16 we report results obtained by TS and BSPS using LWLR, SVM and Neural Networks in the selection process. We can notice that using the same classifier in both selection and classification, the accuracy results are better than those obtained in the previous experiment (Tables 13–15). Based on the results, we can observe that for these classifiers, in the average case, BSPS had the best performance.

In following sections, for BSPS and TS, we report the results obtained when the same classifier is used in both selection and classification stages.

Table 16
Results obtained by *TS* and *BSPS* using *LWLR, SVM* and *Neural Networks* classifiers during the selection process

Dataset	LWLR						SVM						Neural Networks					
	Orig.		TS		BSPS		Orig.		TS		BSPS		Orig.		TS		BSPS	
	Acc.	%	*Acc.*	%	*Acc.*	%	*Acc.*	%	*Acc.*	%	*Acc.*	%	*Acc.*	%	*Acc.*	%	*Acc.*	%
Glass	57.85	100	52.38	6.64	55.99	88.48	65.34	100	60.75	11.36	59.09	57.14	60.79	100	42.96	6.64	52.81	88.40
Iris	98.00	100	98.00	6.22	98.00	22.22	96.00	100	93.33	3.62	93.33	56.79	96.00	100	78.66	6.14	95.99	51.66
Liver	70.13	100	50.00	0.96	71.87	69.45	69.91	100	66.87	3.25	69.77	87.98	72.45	100	64.86	3.25	70.73	95.35
New Thyroid	91.16	100	92.92	7.12	92.59	49.48	93.54	100	70.45	3.82	92.61	56.42	94.37	100	79.43	4.08	92.09	75.52
Wine	92.15	100	88.88	2.49	93.26	46.25	97.18	100	93.88	4.93	98.14	10.18	96.63	100	84.90	7.42	93.30	49.48
Average	**81.86**	**100**	**76.44**	**4.69**	**82.34**	**55.18**	**84.39**	**100**	**77.06**	**5.40**	**82.59**	**53.70**	**84.05**	**100**	**70.16**	**5.51**	**80.98**	**72.08**

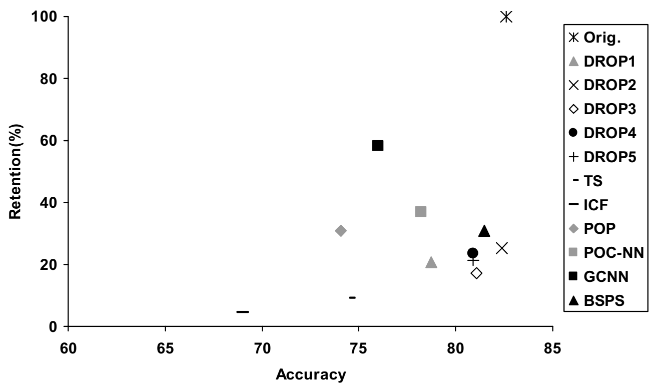

Fig. 5. Scatter graphic from average results shown in Tables 11 and 12.

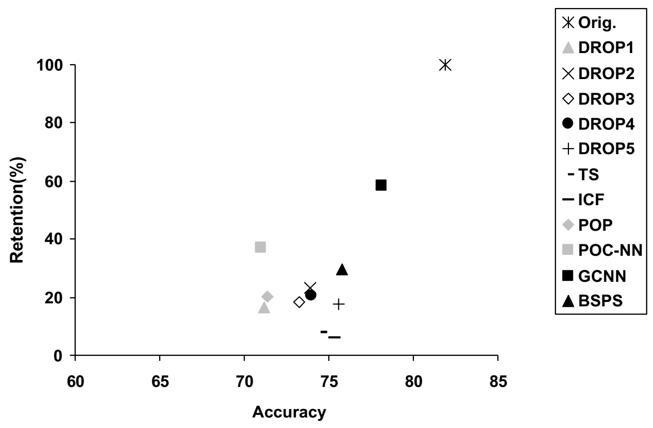

Fig. 6. Scatter graphic from average results shown in Table 13.

3.2. Combining BSPS with other prototype selection methods

In order to reduce the BSPS runtimes we combined BSPS with other prototype selection methods. We first apply a prototype selection method for reducing the sample and then we apply BSPS, that is, BSPS is applied to previously reduced samples. Notice that, this pre-reduction is needed mainly in large dataset since BSPS runtime depends on the number of prototypes in the dataset.

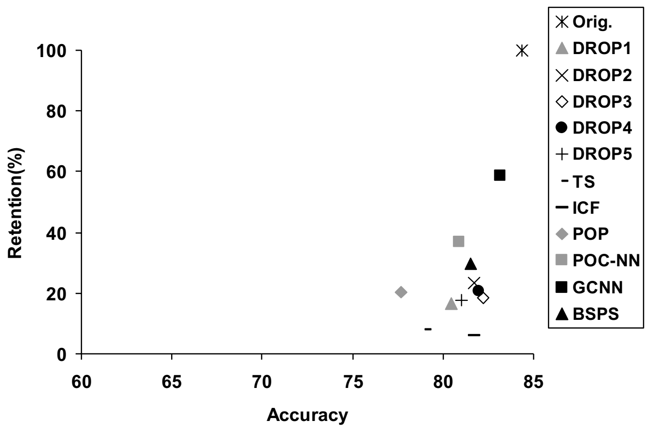

Fig. 7. Scatter graphic from average results shown in Table 14.

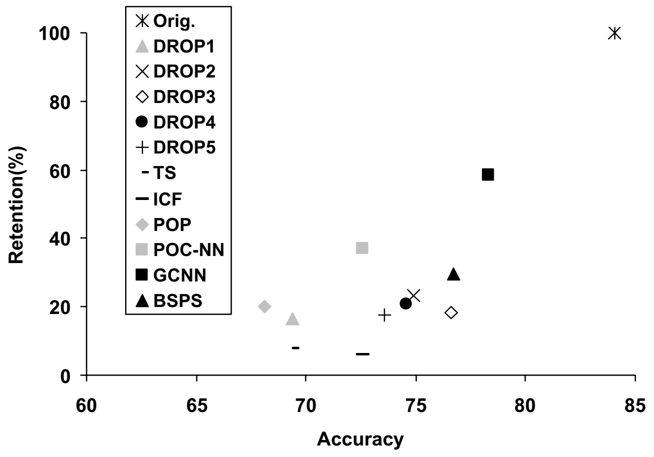

Fig. 8. Scatter graphic from average results shown in Table 15.

For reducing the sample, we used ENN, DROP2...DROP5 and GCNN, that is, we applied the combinations ENN + BSPS, DROP + BSPS and GCNN + BSPS.

3.2.1. Experimental comparison among BSPS combinations

In this section, we report the results obtained by our combinations using *k-NN*, $k = 3$, these results are shown in Table 17 and Fig. 9.

The ENN + BSPS, DROP + BSPS and GCNN + BSPS retention results were better than BSPS. The BSPS accuracy results outperformed DROP + BSPS but they reduce the BSPS runtimes.

In Table 18, we show the average runtimes[1] spent by these methods for each dataset. The results in Table 18 show that the combination of BSPS with other methods indeed reduces the BSPS runtimes.

In Tables 19–21 and Figs 10–12, we show the results obtained by TS and the combinations of BSPS using LWLR, SVM and Neural Networks in the selection process.

[1]These runtimes were obtained using an Intel Celeron CPU 2.4 GHz, 512 MB RAM.

Table 17
Accuracy (*Acc.*) and retention (%) results obtained by: *Orig.*, *ENN + BSPS*, *DROP + BSPS* and *GCNN + BSPS* using *k-NN*, $k = 3$

Dataset	Orig.		ENN + BSPS		DROP2 + BSPS		DROP3 + BSPS		DROP4 + BSPS		DROP5 + BSPS		GCNN + BSPS		BSPS	
	Acc.	%	*Acc.*	%	*Acc.*	%	*Acc.*	%	*Acc.*	%	*Acc.*	%	*Acc.*	%	*Acc.*	%
Bridges	66.09	100	58.00	23.46	60.00	7.21	62.90	6.26	68.54	6.57	66.18	8.11	56.00	28.21	60.18	58.80
Echocardiogram	95.71	100	86.48	6.00	91.60	6.00	95.43	6.00	95.53	6.30	95.71	6.00	86.42	9.60	91.96	25.18
Glass	71.42	100	69.41	21.81	60.36	17.70	59.78	14.95	59.78	17.18	54.24	15.21	61.27	40.13	65.02	46.36
Iris	94.66	100	93.00	8.00	74.6	8.18	88.00	6.42	88.00	6.64	89.33	6.39	86.66	12.22	94.00	15.26
Liver	65.22	100	57.67	26.69	60.80	14.26	59.77	10.91	61.21	12.36	57.95	11.75	57.36	53.20	57.85	58.90
New Thyroid	95.45	100	93.09	5.63	88.96	7.07	91.19	4.28	91.16	4.39	88.29	3.51	92.55	11.93	91.25	14.09
Tae	51.08	100	46.66	43.85	50.29	17.44	47.70	14.93	50.00	18.17	46.66	20.08	42.37	25.30	51.54	54.01
Wine	94.44	100	92.74	8.17	87.67	4.93	90.49	5.05	90.49	5.05	91.04	4.43	94.41	10.67	94.44	13.76
Zoo	93.33	100	91.11	12.59	83.33	10.00	77.77	11.72	77.77	11.97	83.33	7.76	91.11	16.54	91.55	20.86
Average	**80.60**	**100**	**76.46**	**17.36**	**73.07**	**10.31**	**74.78**	**8.95**	**75.83**	**9.85**	**74.75**	**9.25**	**74.24**	**23.09**	**77.53**	**34.14**

Table 18
Runtimes (in seconds) spent by *ENN + BSPS*, *DROP + BSPS*, *GCNN + BS* and *BSPS*

Dataset	ENN + BSPS	DROP2 + BSPS	DROP3 + BSPS	DROP4 + BSPS	DROP5 + BSPS	GCNN + BSPS	BSPS
Bridges	1075.89	21.30	4.80	13.10	8.20	1402.33	1942.50
Echocardiogram	450.12	4.20	2.40	3.05	3.60	8.34	510.28
Glass	426.35	75.20	39.80	78.60	45.12	216.99	560.20
Iris	358.69	9.03	5.06	5.60	5.80	28.33	465.82
Liver	1420.08	3.06	144.56	145.73	120.09	2321.98	3504.81
New Thyroid	1265.23	15.20	5.60	8.40	4.60	43.18	1396.76
Tae	56.18	36.08	17.60	26.40	32.20	12.88	105.33
Wine	836.74	16.73	17.40	13.80	7.08	641.76	980.61
Zoo	1504.67	7.70	6.20	6.10	5.30	9.30	1980.55
Average	**821.55**	**20.94**	**27.05**	**33.42**	**25.78**	**520.57**	**1271.87**

Table 19
Accuracy (*Acc.*) and retention (%) results obtained by: *Orig.*, *TS*, *ENN + BSPS*, *DROP + BSPS* and *GCNN + BSPS* using *LWLR*

Dataset	Orig.		TS		ENN + BSPS		DROP2 + BSPS		DROP3 + BSPS		DROP4 + BSPS		DROP5 + BSPS		GCNN + BSPS	
	Acc.	%	*Acc.*	%	*Acc.*	%	*Acc.*	%	*Acc.*	%	*Acc.*	%	*Acc.*	%	*Acc.*	%
Glass	57.85	100	52.38	6.64	56.84	50.26	48.92	31.05	50.71	20.83	55.18	25.54	53.72	21.97	57.83	56.34
Iris	98.00	100	98.00	6.22	96.66	20.73	90.67	14.22	93.33	10.89	88.67	11.19	88.67	8.15	96.66	30.96
Liver	70.13	100	50.00	0.96	66.34	36.00	70.25	18.94	68.99	17.13	68.08	19.00	68.68	16.59	70.69	80.51
New Thyroid	91.16	100	92.92	7.12	89.95	51.06	79.24	12.15	74.52	7.80	77.27	8.01	79.78	7.34	85.10	29.50
Wine	92.15	100	88.88	2.49	92.68	57.51	86.61	14.23	78.51	14.30	76.14	14.30	85.03	8.93	92.68	63.98
Average	**81.86**	**100**	**76.44**	**4.69**	**80.47**	**43.11**	**75.18**	**18.12**	**73.21**	**14.19**	**73.07**	**15.61**	**75.18**	**12.60**	**80.59**	**52.26**

Table 20
Accuracy (*Acc.*) and retention (%) results obtained by: *Orig.*, *TS*, *ENN + BSPS*, *DROP + BSPS* and *GCNN + BSPS* using *SVM*

Dataset	Orig.		TS		ENN + BSPS		DROP2 + BSPS		DROP3 + BSPS		DROP4 + BSPS		DROP5 + BSPS		GCNN + BSPS	
	Acc.	%	*Acc.*	%	*Acc.*	%	*Acc.*	%	*Acc.*	%	*Acc.*	%	*Acc.*	%	*Acc.*	%
Glass	65.34	100	60.75	11.36	66.82	40.82	60.30	22.48	61.31	17.29	64.87	23.73	61.90	15.42	66.82	58.51
Iris	96.00	100	93.33	3.62	96.00	8.89	95.33	6.44	93.33	3.70	94.00	4.07	94.67	3.33	96.00	25.03
Liver	69.91	100	66.87	3.25	68.67	45.84	64.36	19.10	62.64	16.07	64.10	17.30	62.66	18.42	69.71	80.31
New Thyroid	93.54	100	76.94	1.06	93.54	7.23	91.79	5.37	89.34	3.20	89.20	3.36	88.27	3.31	91.64	16.02
Wine	97.18	100	93.88	4.93	96.63	21.68	96.01	3.50	93.89	3.62	94.97	3.87	92.09	2.75	96.11	17.59
Average	**84.39**	**100**	**77.06**	**5.40**	**84.33**	**24.89**	**81.56**	**11.38**	**80.10**	**8.78**	**81.43**	**10.47**	**79.92**	**8.65**	**84.06**	**39.49**

Table 21
Accuracy (*Acc.*) and retention (%) results obtained by: *Orig.*, *TS*, *ENN + BSPS*, *DROP + BSPS* and *GCNN + BSPS* using *Neural Networks*

Dataset	Orig.		TS		ENN + BSPS		DROP2 + BSPS		DROP3 + BSPS		DROP4 + BSPS		DROP5 + BSPS		GCNN + BSPS	
	Acc.	%	*Acc.*	%	*Acc.*	%	*Acc.*	%	*Acc.*	%	*Acc.*	%	*Acc.*	%	*Acc.*	%
Glass	60.79	100	42.96	6.64	54.56	67.87	47.57	30.01	48.03	23.05	50.36	28.28	50.38	24.66	47.61	60.23
Iris	96.00	100	78.66	6.14	88.88	18.51	86.66	11.33	84.66	7.33	86.66	7.33	81.33	4.37	90.00	27.48
Liver	72.45	100	64.86	3.25	64.93	63.74	67.27	26.73	65.84	19.71	61.49	21.44	63.74	21.77	71.61	82.99
New Thyroid	94.37	100	79.43	4.08	88.05	31.77	87.87	10.33	82.27	4.39	79.95	4.49	77.66	3.66	86.08	17.47
Wine	96.63	100	84.90	7.42	92.75	37.56	81.96	9.36	78.56	10.17	78.77	10.17	82.51	6.17	94.44	38.45
Average	**84.05**	**100**	**70.16**	**5.51**	**77.83**	**43.89**	**74.27**	**17.55**	**71.87**	**12.93**	**71.45**	**14.34**	**71.12**	**12.13**	**77.95**	**45.32**

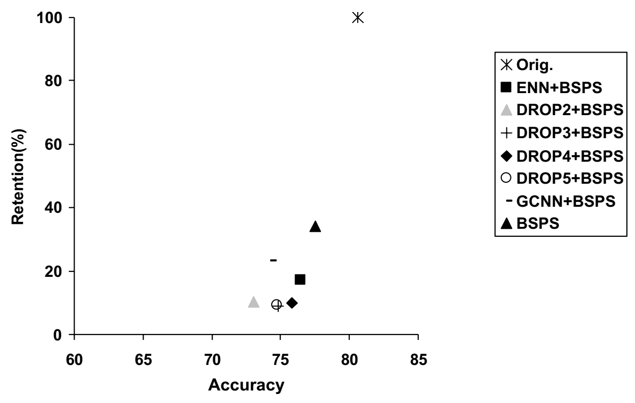

Fig. 9. Scatter graphic from average results shown in Table 17.

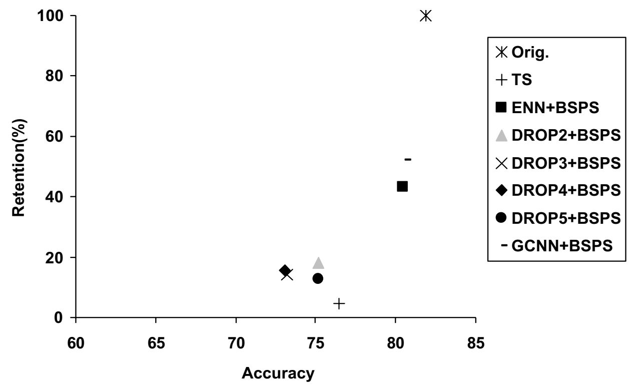

Fig. 10. Scatter graphic from average results shown in Table 19.

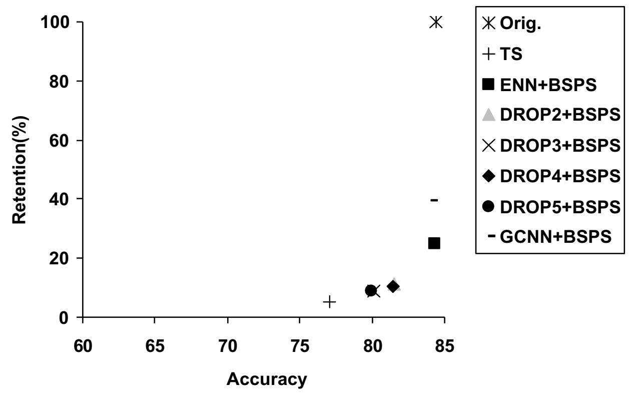

Fig. 11. Scatter graphic from average results shown in Table 20.

We can notice that although the BSPS combinations use methods based on the *k-NN* rule in the initial step, BSPS allows obtaining good results by using other classifiers in the last step of the selection process. In these experiments, for SVM and Neural Networks, all the BSPS combinations outperformed TS.

BSPS is a backward sequential technique for prototype selection, the counterpart of BSPS is FSPS

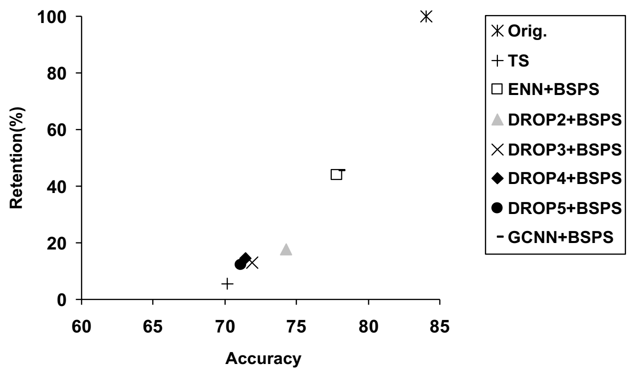

Fig. 12. Scatter graphic from average results shown in Table 21.

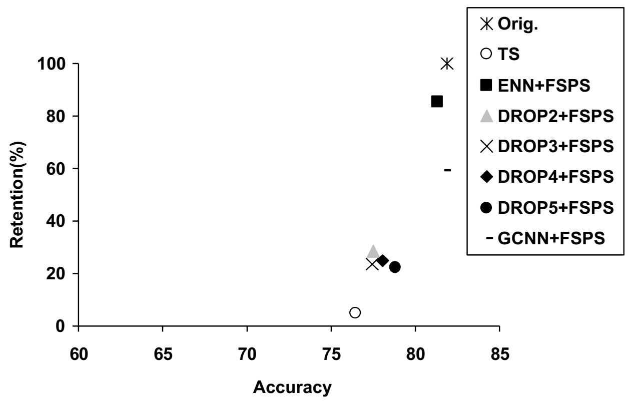

Fig. 13. Scatter graphic from average results shown in Table 23.

(Forward Sequential Prototype Selection) which starts with an empty set and at each step it includes those prototypes that strictly enhance the accuracy (according to the evaluation function).

For prototype selection, starting from an empty set using FSPS will lead us to obtain bad accuracy results because of the inclusion criterion of FSPS. Instead, we apply FSPS starting from an initial set, which was obtained by ENN, DROP and GCNN methods, that is, as in the above experiment, we applied ENN + FSPS, DROP + FSPS and GCNN + FSPS. The results are reported in Table 22.

Comparing Tables 17 and 22, we can observe that the accuracy obtained by the combinations of FSPS was better than the obtained by BSPS and its combinations (Table 17). In addition, we can notice that the subset sizes obtained by FSPS combinations were bigger than the obtained by the combinations of BSPS.

In Tables 23–25 and Figs 13–15 we show the results obtained by the FSPS combinations and TS using LWLR, SVM and Neural Networks in the selection process.

In these experiments, TS obtained the best reduction results but in accuracy, the FSPS combinations outperformed TS. Among the FSPS combinations, the best accuracy results were obtained by ENN + FSPS and GCNN + FSPS.

Table 22
Accuracy (*Acc.*) and retention (%) results obtained by: *Orig.*, *ENN + FSPS*, *DROP + FSPS* and *GCNN + FSPS*

Dataset	Orig.		ENN + FSPS		DROP2 + FSPS		DROP3 + FSPS		DROP4 + FSPS		DROP5 + FSPS		GCNN + FSPS	
	Acc.	%	*Acc.*	%	*Acc.*	%	*Acc.*	%	*Acc.*	%	*Acc.*	%	*Acc.*	%
Bridges	66.09	100	60.00	64.94	62.09	36.04	62.90	20.61	62.09	26.80	54.45	32.98	48.72	88.23
Echocardiogram	95.71	100	93.21	91.04	93.21	16.41	93.21	19.69	93.21	16.41	89.10	19.40	90.53	23.42
Glass	71.42	100	70.29	71.96	68.67	43.29	69.13	35.46	64.41	44.65	66.32	39.92	67.77	62.09
Iris	94.66	100	93.33	95.04	95.33	18.29	95.33	16.81	93.33	16.74	94.66	13.48	94.66	38.44
Liver	65.22	100	65.22	66.44	63.52	39.97	60.98	29.43	65.29	33.11	61.17	31.65	65.22	84.02
New Thyroid	95.45	100	93.07	95.03	94.02	15.97	94.41	11.99	94.00	12.35	95.41	11.31	93.96	30.75
Tae	51.08	100	51.08	56.07	40.25	47.45	41.62	39.30	40.54	44.89	33.50	42.90	50.29	32.44
Wine	94.44	100	94.44	96.06	94.44	16.23	94.07	16.54	94.44	16.54	94.07	11.92	94.44	79.21
Zoo	93.33	100	92.22	94.32	90.33	23.82	88.88	24.81	90.33	24.46	83.33	25.55	92.22	26.79
Average	**80.60**	**100**	**79.21**	**81.21**	**77.98**	**28.61**	**77.84**	**23.85**	**77.52**	**26.22**	**74.67**	**25.46**	**77.53**	**51.71**

Table 23
Accuracy (*Acc.*) and retention (%) results obtained by: *Orig.*, *TS*, *ENN + FSPS*, *DROP + FSPS* and *GCNN + FSPS* using *LWLR*

Dataset	Orig.		TS		ENN + FSPS		DROP2 + FSPS		DROP3 + FSPS		DROP4 + FSPS		DROP5 + FSPS		GCNN + FSPS	
	Acc.	%	*Acc.*	%	*Acc.*	%	*Acc.*	%	*Acc.*	%	*Acc.*	%	*Acc.*	%	*Acc.*	%
Glass	57.85	100	52.38	6.64	57.14	73.05	50.82	33.74	54.43	28.13	57.33	33.28	55.00	28.18	56.88	62.30
Iris	98.00	100	98.00	6.22	98.00	94.74	97.33	19.48	95.33	17.77	95.33	17.77	97.33	14.22	97.33	38.15
Liver	70.13	100	50.00	0.96	69.71	66.25	69.83	49.56	67.64	32.60	68.55	35.41	68.97	35.54	70.13	84.44
New Thyroid	91.16	100	92.92	7.12	90.62	95.35	86.06	21.39	86.99	16.95	86.55	16.22	88.39	15.09	91.16	30.49
Wine	92.15	100	88.88	2.49	90.98	96.50	83.56	18.85	82.61	22.03	82.61	22.03	84.54	17.16	93.24	79.52
Average	**81.86**	**100**	**76.44**	**4.69**	**81.29**	**85.18**	**77.52**	**28.60**	**77.40**	**23.50**	**78.07**	**24.94**	**78.85**	**22.04**	**81.75**	**58.98**

Table 24
Accuracy (*Acc.*) and retention (%) results obtained by: *Orig.*, *TS*, *ENN* + *FSPS* and *DROP* + *FSPS* using *SVM*

Dataset	Orig.		TS		ENN + FSPS		DROP2 + FSPS		DROP3 + FSPS		DROP4 + FSPS		DROP5 + FSPS		GCNN + FSPS	
	Acc.	%	*Acc.*	%	*Acc.*	%	*Acc.*	%	*Acc.*	%	*Acc.*	%	*Acc.*	%	*Acc.*	%
Glass	65.34	100	60.75	11.36	67.27	71.44	61.30	32.08	60.80	27.05	61.90	34.71	61.30	26.99	65.87	62.15
Iris	96.00	100	93.33	3.62	96.00	95.19	95.33	17.69	94.00	16.29	93.33	15.55	95.33	13.03	96.00	38.81
Liver	69.91	100	66.87	3.25	69.83	65.73	57.36	40.66	59.91	28.97	57.97	35.88	58.25	32.80	69.91	84.77
New Thyroid	93.54	100	76.94	1.06	93.10	94.99	85.23	15.39	90.65	12.95	90.20	11.91	87.54	9.32	91.62	30.85
Wine	97.18	100	93.88	4.93	97.22	96.01	93.82	14.80	95.23	15.35	95.52	15.04	91.01	11.23	97.18	79.15
Average	**84.39**	**100**	**77.06**	**5.40**	**84.68**	**84.67**	**78.61**	**24.12**	**80.12**	**20.12**	**79.78**	**22.62**	**78.69**	**18.67**	**84.12**	**59.15**

Table 25
Accuracy (*Acc.*) and retention (%) results obtained by: *Orig.*, *TS*, *ENN* + *FSPS*, *DROP* + *FSPS* and *GCNN* + *FSPS* using *Neural Networks*

Dataset	Orig.		TS		ENN + FSPS		DROP2 + FSPS		DROP3 + FSPS		DROP4 + FSPS		DROP5 + FSPS		GCNN + FSPS	
	Acc.	%	*Acc.*	%	*Acc.*	%	*Acc.*	%	*Acc.*	%	*Acc.*	%	*Acc.*	%	*Acc.*	%
Glass	60.79	100	42.96	6.64	53.66	71.76	49.45	33.74	50.34	27.20	45.73	31.15	48.16	29.02	49.76	63.19
Iris	96.00	100	78.66	6.14	95.33	95.11	94.00	19.03	90.00	17.40	88.00	18.07	87.33	14.66	95.33	38.96
Liver	72.45	100	64.86	3.25	66.34	65.73	66.77	28.65	68.72	22.15	62.06	23.64	64.97	23.96	69.84	84.77
New Thyroid	94.37	100	79.43	4.08	94.03	95.04	91.73	16.69	87.05	11.26	86.60	11.57	89.32	10.85	92.99	30.80
Wine	96.63	100	84.90	7.42	94.67	96.13	87.09	16.17	92.09	17.54	96.07	17.59	89.28	12.98	96.63	79.21
Average	**84.05**	**100**	**70.16**	**5.51**	**81.21**	**84.75**	**77.81**	**22.84**	**77.64**	**19.11**	**75.69**	**20.40**	**75.81**	**18.29**	**80.91**	**59.39**

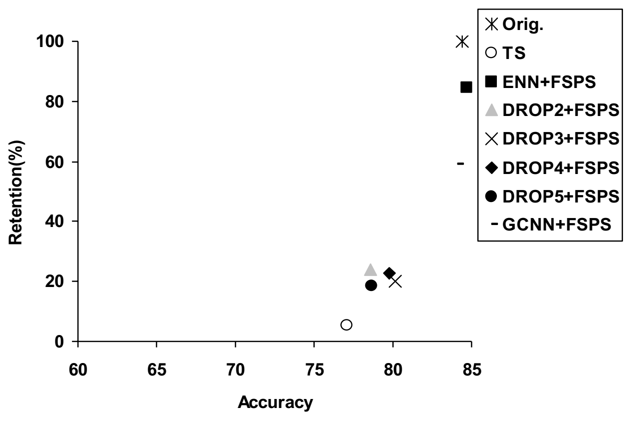

Fig. 14. Scatter graphic from average results sown in Table 24.

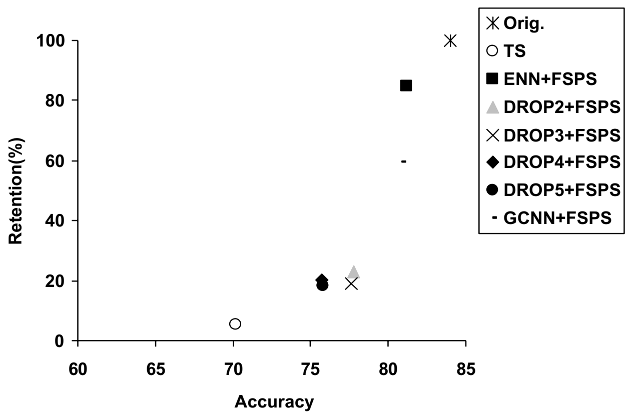

Fig. 15. Scatter graphic from average results sown in Table 25.

3.3. *Restricted floating search for prototype selection*

Since in BSPS there are not backtrack steps, it does not try to improve the quality of a subset by adding prototypes. The backtrack steps can be done by the Sequential Floating Search (SFS) strategies [27] which provide a good compromise between accuracy and reduction. SFS consists in applying conditional inclusion/exclusion steps after each exclusion/inclusion in the set. This kind of search (as sequential search) can be done in the backward and forward directions.

The backward SFS applies (after each exclusion step) a number of inclusion steps as long as the classification results are better than the previously evaluated ones. The forward SFS is the counterpart of backward SFS. These floating searches are very expensive therefore we propose a prototype selection method based on the backward SFS but in a restricted way. Our method named Restricted Floating Prototype Selection (RFPS) applies an exclusion process followed by the conditional inclusion of discarded prototypes.

RFPS starts applying a pre-processing step followed by the exclusion process and finally the conditional inclusion is applied over the prototype set previously selected (S, $S \subset T$).The exclusion process is carried out by applying BSPS.

The selection process in RFPS consists in analyzing (conditional inclusion) the prototypes discarded from T (prototypes in the set $D = T - S$) for including in S those prototypes such that their inclusion improves the classification, that is, a prototype $p \in D$ is included in S only if the classification obtained using $S \cup \{p\}$ is better than the obtained using S. We used ENN, DROPs or GCNN methods for the pre-processing step.

The RFPS algorithm is as follows:

```
RFPS (Training set T): prototype subset S
//pre-proocessing
Let S = subset obtained after applying ENN, DROPs or GCNN over T
Best_val = Classif(S)
Repeat                          //exclusion process
            Worst = None
            For each prototype $p$ in S
              S' = S-$\{p\}$
              Eval = Classifier(S')
              If Eval $\geqslant$ Best_val
                Then Worst = $p$
                Best_val = Eval
            If Worst $\neq$ None
                Then S = S-$\{$Worst$\}$
Until Worst == None or |S| == 1
D = T-S
For each prototype $p_i$ in D       //conditional inclusion
  S" = S$\cup\{p_i\}$
  Eval = Classifier(S")
  If Eval > Best_val
    Then Best_val = Eval
    S = S$\cup\{p_i\}$
Return S
```

The RFPS is a restricted floating search method because first, it applies only an exclusion process followed by the conditional inclusion.

This restricted floating method can be done in the inverse direction (RFPS-*Inv*), that is, first applying an inclusion process followed by the conditional exclusion.

3.3.1. Experimental comparison of RFPS and RFPS-Inv

We have applied both RFPS and RFPS-*Inv* using *k-NN*, $k = 3$ over the datasets used for testing BSPS. The results are reported in Tables 26 and 27. In Table 26, ENN + RFPS is the RFPS method using ENN for the pre-processing step and in the same way, for DROP2 + RFPS, ..., DROP5 + RFPS, GCNN + RFPS, the DROP2, ..., DROP5 and GCNN methods were respectively used.

We can observe that RFPS outperformed to RFPS-*Inv* because this last method discards relevant prototypes in the final exclusion step. In addition, the accuracy obtained by RFPS is better than the obtained by the BSPS combinations (Table 17); this is because RFPS includes relevant prototypes discarded in the exclusion steps.

As a consequence of the final inclusion step, the prototype sets obtained by RFPS are slightly bigger than those obtained by RFPS-*Inv* and BSPS combinations.

As in previous experiments, in Tables 28–30 and Figs 16–18, the results obtained by RFPS using LWLR, SVM and Neural Networks in the selection process are reported. Based on these results, we can notice that because of the conditional inclusion, again RFPS outperforms BSPS combinations (Tables 19–21).

Table 26
Accuracy (*Acc.*) and retention (%) results obtained by: *Orig.* and *RFPS* using *k-NN*, $k = 3$

Dataset	Orig.		ENN + RSPS		DROP2 + RSPS		DROP3 + RSPS		DROP4 + RSPS		DROP5 + RSPS		GCNN + RSPS	
	Acc.	%	*Acc.*	%	*Acc.*	%	*Acc.*	%	*Acc.*	%	*Acc.*	%	*Acc.*	%
Bridges	66.09	100	60.00	48.12	50.45	24.74	56.45	14.28	61.09	14.43	53.63	23.71	59.00	37.49
Echocardiogram	95.71	100	93.23	86.56	93.21	8.95	91.96	9.85	91.96	7.46	87.67	8.16	93.21	12.30
Glass	71.42	100	69.43	29.34	63.54	26.12	64.48	25.75	65.41	27.46	67.74	26.11	69.67	44.70
Iris	94.66	100	93.33	10.07	92.00	12.12	93.00	9.92	93.33	10.29	93.33	10.00	92.00	14.59
Liver	65.22	100	59.98	33.68	63.48	19.09	61.70	16.94	65.00	10.39	60.03	19.64	59.41	56.07
New Thyroid	95.45	100	94.04	7.02	93.98	9.30	93.98	6.25	94.45	6.77	90.47	5.94	93.98	13.17
Tae	51.08	100	50.70	48.88	53.33	33.18	47.70	25.45	50.00	27.29	53.33	31.34	53.33	36.85
Wine	94.44	100	93.63	10.23	91.03	8.30	94.44	8.17	94.44	8.17	93.85	8.30	93.85	11.23
Zoo	93.33	100	91.33	71.14	91.11	13.58	91.33	14.81	91.33	14.69	91.11	14.93	91.33	17.90
Average	**80.60**	**100**	**78.41**	**38.34**	**76.90**	**17.26**	**77.23**	**14.60**	**78.56**	**14.11**	**76.80**	**16.46**	**78.42**	**27.14**

Table 27
Accuracy (*Acc.*) and retention (%) results obtained by: *Orig.* and *RFPS-Inv* using *k-NN*, $k = 3$

Dataset	Orig.		ENN + RFPS-Inv		DROP2 + RFPS-Inv		DROP3 + RFPS-Inv		DROP4 + RFPS-Inv		DROP5 + RFPS-Inv		GCNN + RFPS-Inv	
	Acc.	%	*Acc.*	%	*Acc.*	%	*Acc.*	%	*Acc.*	%	*Acc.*	%	*Acc.*	%
Bridges	66.09	100	51.54	26.80	60.09	23.71	55.45	12.37	58.27	15.30	54.54	19.58	42.36	26.18
Echocardiogram	95.71	100	93.21	11.94	91.78	8.95	87.85	8.95	87.85	7.46	89.10	7.46	89.10	8.40
Glass	71.42	100	58.35	19.47	64.00	20.00	43.50	23.36	54.95	20.77	55.49	16.45	61.75	40.34
Iris	94.66	100	80.66	5.33	86.00	9.18	92.66	7.18	92.00	7.25	86.00	6.29	91.33	12.88
Liver	65.22	100	58.84	21.80	59.47	24.38	59.75	19.25	61.20	20.58	60.30	19.54	56.50	53.43
New Thyroid	95.45	100	88.83	3.82	86.12	8.37	94.55	4.85	93.03	5.27	86.96	4.18	92.55	11.93
Tae	51.08	100	50.97	14.27	51.33	25.31	46.66	20.27	45.54	31.86	54.20	30.17	45.00	25.74
Wine	94.44	100	90.00	4.36	89.41	5.30	88.23	5.18	89.44	5.36	91.04	4.42	94.41	10.67
Zoo	93.33	100	91.11	15.92	78.88	12.96	90.00	13.58	78.88	13.45	80.00	14.19	80.00	17.28
Average	**80.60**	**100**	**73.72**	**13.75**	**74.12**	**15.35**	**73.18**	**12.78**	**73.46**	**14.14**	**73.07**	**13.59**	**72.56**	**22.98**

Table 28
Accuracy (*Acc.*) and retention (%) results obtained by: *Orig.*, *TS* and *RFPS* using *LWLR*

Dataset	Orig.		TS		ENN + RFPS		DROP2 + RFPS		DROP3 + RFPS		DROP4 + RFPS		DROP5 + RFPS		GCNN + RFPS	
	Acc.	%	*Acc.*	%	*Acc.*	%	*Acc.*	%	*Acc.*	%	*Acc.*	%	*Acc.*	%	*Acc.*	%
Glass	57.85	100	52.38	6.64	57.79	52.18	50.82	33.74	53.30	25.13	58.33	26.26	54.54	27.09	56.41	57.12
Iris	98.00	100	98.00	6.22	97.33	22.00	96.66	16.59	96.00	13.40	95.33	13.77	95.33	10.00	97.33	31.48
Liver	70.13	100	50.00	0.96	65.50	37.61	70.10	20.32	73.31	18.64	71.27	21.22	71.30	18.77	70.70	80.74
New Thyroid	91.16	100	92.92	7.12	91.67	52.00	84.13	18.65	91.21	22.58	91.62	25.93	91.19	19.90	87.45	32.51
Wine	92.15	100	88.88	2.49	92.12	58.76	88.64	18.41	82.58	16.22	81.56	16.87	87.50	16.15	92.15	64.30
Average	**81.86**	**100**	**76.44**	**4.69**	**80.88**	**44.51**	**78.07**	**21.54**	**79.28**	**19.19**	**79.62**	**20.81**	**79.97**	**18.38**	**80.81**	**53.23**

Table 29
Accuracy (*Acc.*) and retention (%) results obtained by: *Orig.*, *TS* and *RFPS* using *SVM*

Dataset	Orig.		TS		ENN + RFPS		DROP2 + RFPS		DROP3 + RFPS		DROP4 + RFPS		DROP5 + RFPS		GCNN + RFPS	
	Acc.	%	*Acc.*	%	*Acc.*	%	*Acc.*	%	*Acc.*	%	*Acc.*	%	*Acc.*	%	*Acc.*	%
Glass	65.34	100	60.75	11.36	69.18	43.24	64.02	25.44	62.26	20.16	64.95	25.50	63.87	69.18	66.84	60.49
Iris	96.00	100	93.33	3.62	96.00	9.78	95.66	7.81	93.33	4.14	94.00	4.14	94.67	96.00	96.00	25.78
Liver	69.91	100	66.87	3.25	69.83	47.68	65.84	20.74	62.95	20.52	67.03	19.62	67.83	69.83	70.42	80.93
New Thyroid	93.54	100	76.94	1.06	93.54	8.16	91.79	5.89	90.07	5.42	90.23	3.77	90.59	93.54	93.54	17.36
Wine	97.18	100	93.88	4.93	96.75	22.75	96.60	6.05	95.55	5.80	95.55	5.86	92.64	96.75	96.08	18.16
Average	**84.39**	**100**	**77.60**	**5.40**	**85.06**	**26.32**	**82.78**	**13.19**	**80.83**	**11.21**	**82.35**	**11.78**	**81.92**	**11.23**	**84.58**	**40.54**

Table 30
Accuracy (*Acc.*) and retention (%) results obtained by: *Orig.*, *TS* and *RFPS* using *Neural Networks*

Dataset	Orig.		TS		ENN + RFPS		DROP2 + RFPS		DROP3 + RFPS		DROP4 + RFPS		DROP5 + RFPS		GCNN + RFPS	
	Acc.	%	*Acc.*	%	*Acc.*	%	*Acc.*	%	*Acc.*	%	*Acc.*	%	*Acc.*	%	*Acc.*	%
Glass	60.79	100	42.96	6.64	63.64	70.27	52.38	32.34	50.88	25.39	56.47	30.47	53.54	27.10	46.21	58.37
Iris	96.00	100	78.66	6.14	87.78	29.01	86.00	14.51	84.66	10.29	86.66	10.37	85.33	7.25	96.00	31.78
Liver	72.45	100	64.86	3.25	68.08	65.54	67.26	28.47	62.12	20.90	67.49	22.86	64.07	23.41	73.62	81.90
New Thyroid	94.37	100	79.43	4.08	91.69	33.68	82.03	12.14	86.51	6.51	85.58	6.14	86.55	5.94	94.37	31.01
Wine	96.63	100	84.90	7.42	96.62	38.95	86.56	12.96	88.20	12.85	85.94	12.67	87.64	9.23	96.11	64.61
Average	**84.05**	**100**	**70.16**	**5.51**	**81.56**	**47.49**	**74.85**	**20.09**	**74.47**	**15.19**	**76.43**	**16.50**	**75.43**	**14.59**	**81.26**	**53.53**

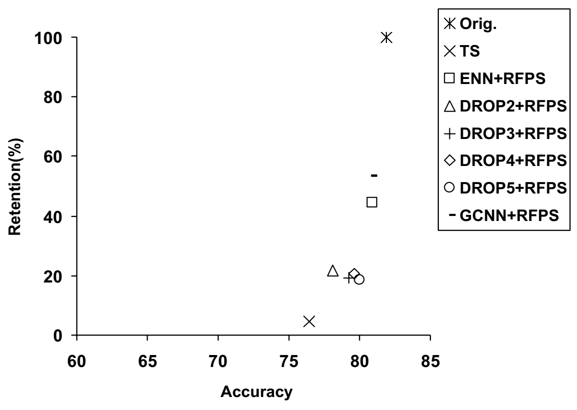

Fig. 16. Scatter graphic from average results shown in Table 28.

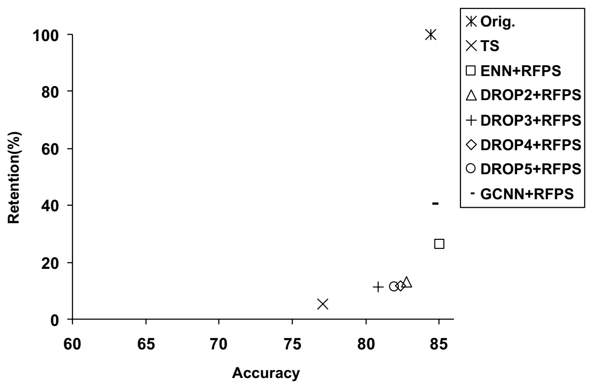

Fig. 17. Scatter graphic from average results shown in Table 29.

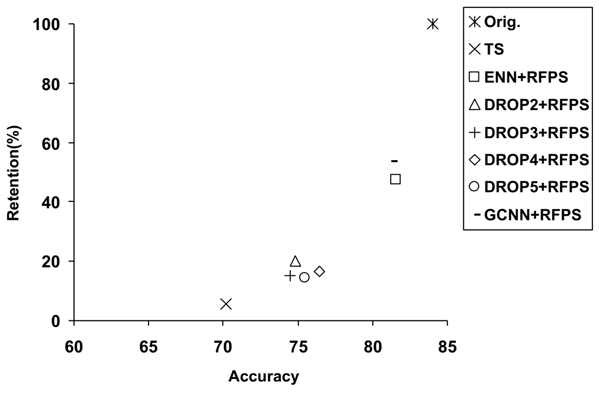

Fig. 18. Scatter graphic from average results shown in Table 30.

In Table 31 we report the runtimes spent by RFPS. We can observe that RFPS spent more runtime than the BSPS combinations (see Table 18) because of the conditional inclusion but there is a small runtime difference between them. In addition, both RFPS and BSPS combinations spent much less runtime than BSPS.

Table 31
Runtimes (in seconds) spent by *RFPS*

Dataset	ENN + RFPS	DROP2 + RFPS	DROP3 + RFPS	DROP4 + RFPS	DROP5 + RFPS	GCNN + RFPS
Bridges	1083.62	40.10	25.50	33.40	32.10	1453.68
Echocardiogram	465.21	8.50	7.10	7.50	8.70	15.13
Glass	431.65	84.47	51.30	87.90	55.20	225.84
Iris	366.31	13.80	11.20	10.30	12.60	33.12
Liver	1435.10	372.00	169.80	170.40	124.70	2345.08
New Thyroid	1279.55	24.90	16.30	17.50	12.60	52.18
Tae	57.03	39.70	21.30	30.60	36.90	16.31
Wine	843.92	19.20	24.50	21.30	13.90	649.83
Zoo	1532.62	19.37	19.60	17.80	14.70	20.57
Average	**832.78**	**69.12**	**38.51**	**44.08**	**34.60**	**534.64**

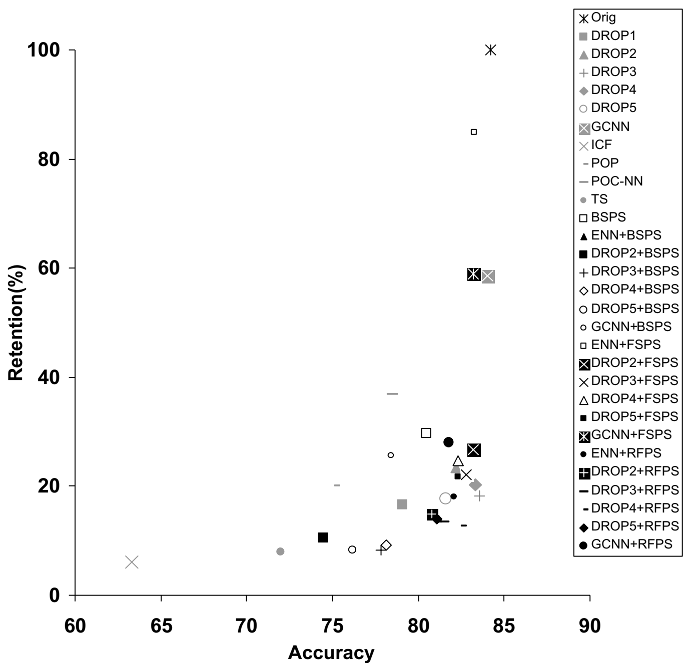

Fig. 19. Scatter graphic of the results obtained by all the prototype selection methods using k-NN, $k = 3$.

4. Discussion

In Section 3.1, we introduced the BSPS prototype selection method based on backward sequential search. In our experiments, BSPS obtained small subsets but with a slight reduction in the classification accuracy, on the average case.

Due to the high complexity of BSPS, this method is useful for small datasets. On the other hand, for medium-large datasets we can apply the combinations of BSPS that were proposed in Section 3.2. These combinations reduce the BSPS runtimes.

In BSPS and its combinations there are not backtrack steps, therefore in Section 3.3 we proposed two Restricted Floating Prototype Selection methods (RFPS and RFPS-Inv) which allow including/excluding prototypes previously discarded/included that contribute for a better accuracy.

In Figs 19–22, as a summary, we show the graphics of the average results obtained (using *k-NN* with $k = 3$, LWLR, SVM and Neural Networks respectively) by all the methods tested along the experimental section. In these results, the five datasets where all the methods can be applied were used (numeric datasets without missing values). Among our methods, we have omitted RFPS-*Inv* since it is outperformed by RFPS. On these figures, our methods are represented by the icons in dark color meanwhile the other methods by the icons in light color.

The best accuracy results using *k-NN* were obtained by GCNN and DROP3. Among our sequential methods, the best were ENN + FSPS and GCNN + FSPS.

Using classifiers different from *k-NN*, GCNN and the DROPs were outperformed in accuracy by some of our methods which can use the same classifier in both the selection and classification stages. For LWLR (Fig. 20) the best accuracy was obtained by BSPS, GCNN + RFPS and ENN + FSPS. For SVM and Neural Networks (Figs 21–22), the best accuracy was obtained by ENN + RFPS and ENN + FSPS.

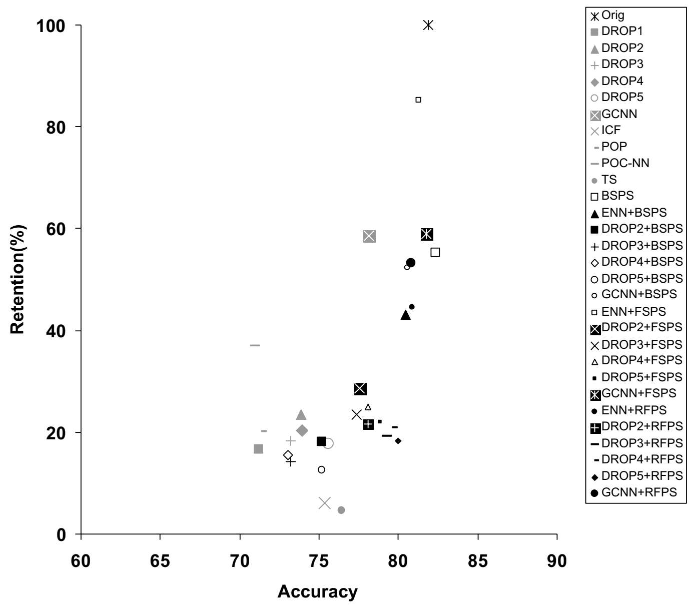

Fig. 20. Scatter graphic of the results obtained by all the prototype selection methods using *LWLR*.

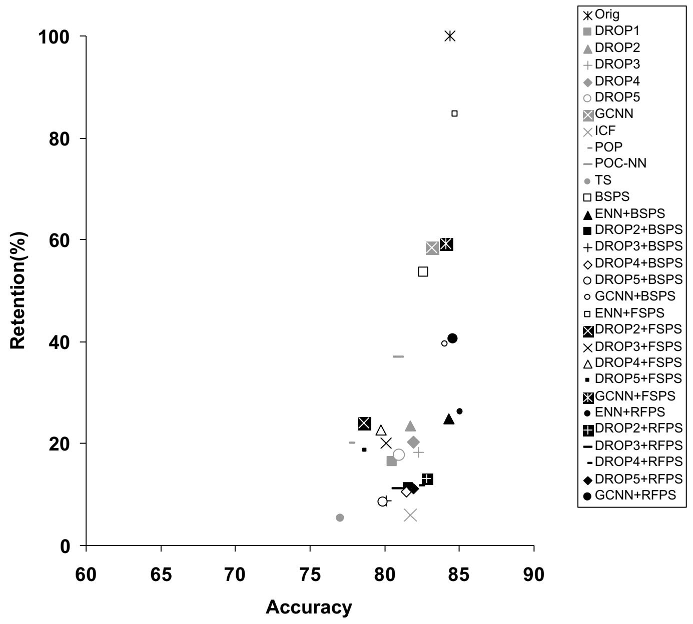

Fig. 21. Scatter graphic of the results obtained by all the prototype selection methods using *SVM*.

From Fig. 21, we can see that even though SVM is itself a prototype selection method, better accuracies can be obtained by applying a prototype selection method before SVM.

According to these results, we can notice that the best retention was obtained by ICF and TS but there is a trade off between retention and accuracy since the best ones in retention are not the best in accuracy and vice versa.

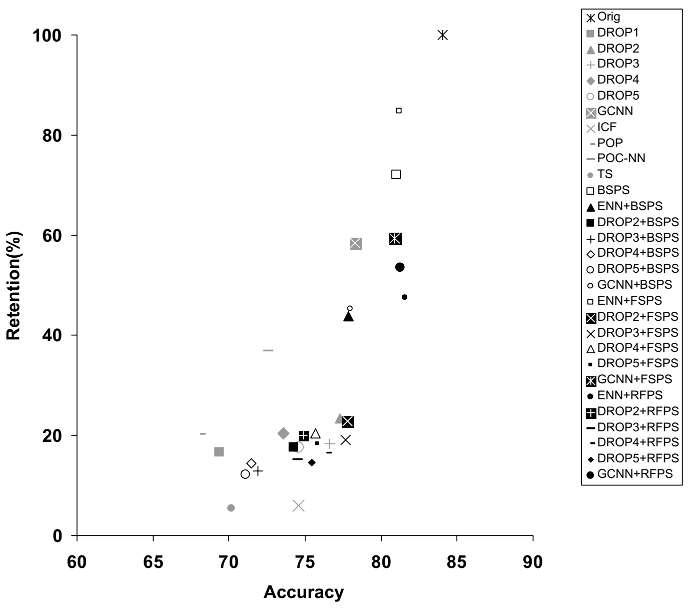

Fig. 22. Scatter graphic of the results obtained by all the prototype selection methods using *Neural Networks*.

Another experiment consisted in evaluating the performance of the methods with different missing values proportion in the data. In particular, we used the Iris data (which has not missing values) where 5%, 10%, 15%, 20% and 25% of the data were randomly changed as missing. We evaluated some of the prototype selection methods (GCNN, DROP3, POC-NN, TS) and some of our best methods (ENN + FSPS, DROP4 + RFPS) according to Fig. 19. The results obtained are shown in Table 32.

Based on these results, we can notice that missing values affects the accuracy since the more number of missing values, the lower accuracy is. The best methods and the less affected by the missing values were DROP3, ENN + FSPS and DROP4 + RFPS.

4.1. *Experiments with synthetic data*

Finally, we present some experiments to visualize the prototype selection in a two-dimensional problem. We tested an artificial dataset with a complex frontier, this frontier was created by hand and randomly we created prototypes over each class side. The dataset consists of 330 prototypes per class. The results obtained after applying some methods compared in previous sections are depicted in Figs 23–24. For each method, we show in parenthesis the obtained accuracy.

All methods shown in Figs 23–24 preserved prototypes near to the frontier. These border prototypes give useful information to preserve the class discrimination regions. These results just showed the kind of prototypes selected by some of the methods in a two dimensions problem. In problems with more than two dimensions, our methods also had a good performance as it can be seen in the results shown in Section 3.

5. Conclusions and further extensions

The quality of the training set in supervised classification problems is very important because based on it new prototypes will be classified. Unfortunately, in the practice the samples contain prototypes with a

Table 32
Accuracy (*Acc.*) and retention (%) results obtained by: *Orig.*, *GCNN*, *DROP3*, *TS*, *POC-NN*, *ENN* + *FSPS* and *DROP4* + *RFPS* applied over the Iris dataset varying the proportion of missing values

Missing values (%)	Orig.		GCNN		DROP3		TS		POC-NN		ENN + FSPS		DROP4 + RFPS	
	Acc.	%	*Acc.*	%	*Acc.*	%	*Acc.*	%	*Acc.*	%	*Acc.*	%	*Acc.*	%
5	96.67	100	86.66	20.44	93.33	12.00	66.66	6.07	85.66	21.70	95.33	95.92	93.33	11.33
10	96.67	100	82.66	15.77	94.00	12.22	60.00	6.37	80.66	26.51	95.33	96.59	92.66	9.55
15	96.00	100	80.66	20.44	96.00	10.89	62.00	7.40	74.00	25.85	93.33	96.07	92.66	11.11
20	92.67	100	81.33	16.14	91.33	11.93	64.00	8.66	76.66	31.33	92.00	94.37	88.00	9.55
25	94.00	100	80.00	18.66	92.67	16.15	68.66	7.40	73.33	39.92	90.00	95.18	94.00	10.51

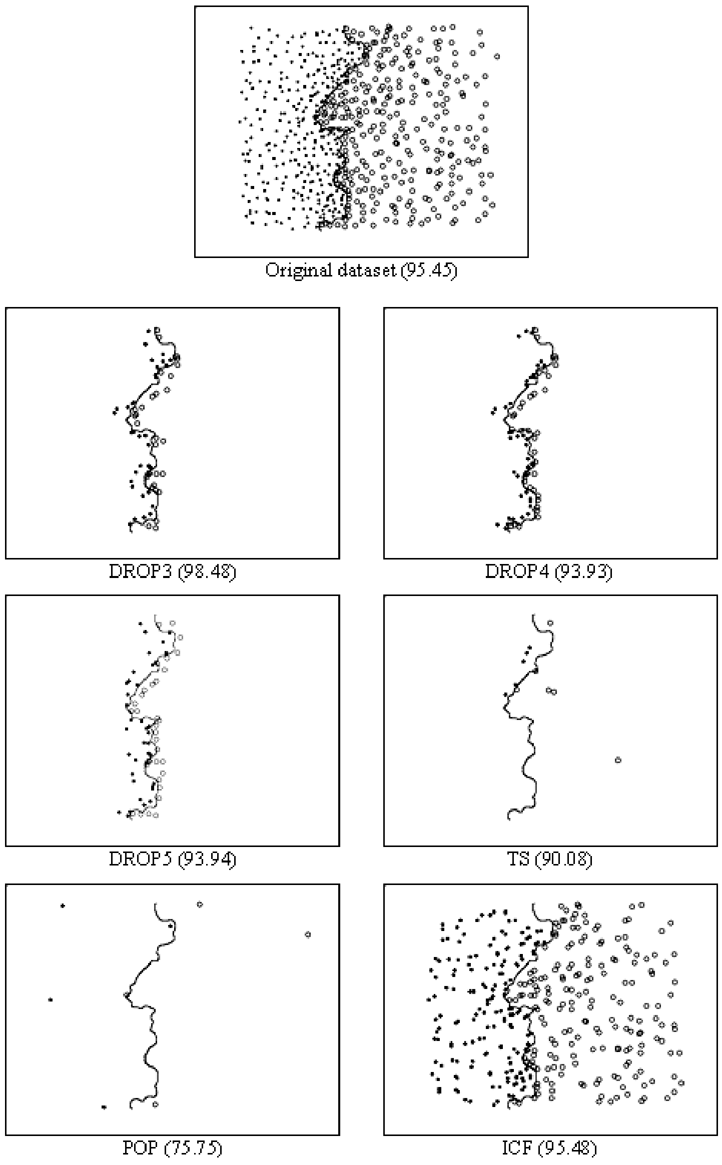

Fig. 23. Results obtained in the synthetic data experiment by: *DROP3, DROP4, DROP5, TS, POP* and *ICF.*

null or even negative contribution for classification accuracy; therefore, it is necessary to select from the original sample a training subset such that it does not contain irrelevant prototypes.

In this paper, we have explored the sequential search for prototype selection. We proposed three sequential prototype selection methods; the first of them is BSPS, based in backward sequential search that constitutes an alternative for reducing the training data. Based on our experimentation we can say that BSPS significantly reduces the size of the original sample preserving good classification accuracy. The first point is a desirable condition for classifiers, especially for instance-based classifiers, because if we provide a small training sample they will reduce their runtimes at the classification stage.

From the experimental results, when our proposed methods were compared against other methods,

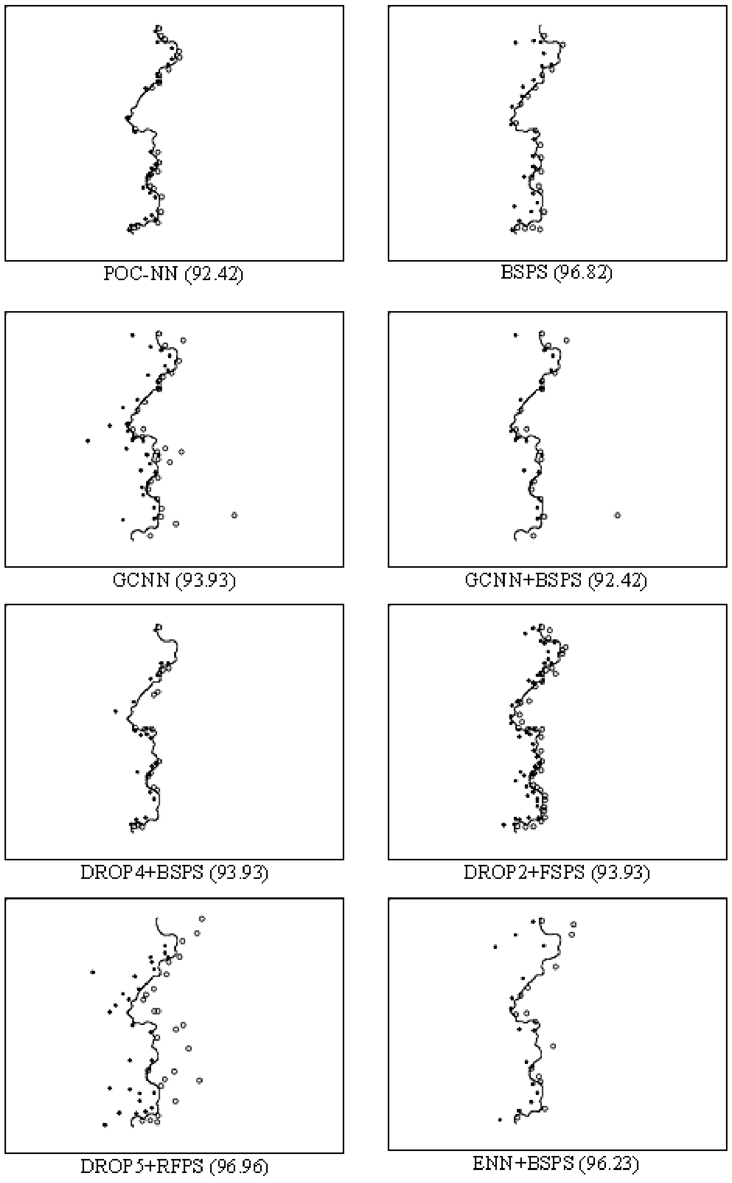

Fig. 24. Results obtained in the synthetic data experiment by: *POC-NN, BSPS, GCNN, GCNN + BSPS, DROP4 + BSPS, DROP2 + FSPS, DROP5 + RFPS* and *ENN + BSPS.*

we found that BSPS has a good compromise between classification accuracy and sample reduction. In addition, our method obtained better results than other well-known methods such as DROP1, ICF and tabu search.

In order to reduce the runtimes of BSPS, we proposed methods that consist in combining BSPS with other prototype selection methods (ENN, DROPs and GCNN). The results obtained show that our combinations slightly reduce the accuracy obtained by BSPS but they spend less runtime.

In addition, we have proposed two restricted floating search methods (RFPS and RFPS-*Inv*) for prototype selection. These methods allow backtrack steps, which cannot be done in BSPS and its combinations. According to our experimentation, RFPS was better than RFPS-*Inv*. Due to the conditional inclusion in RFPS, this method spends more runtime than BSPS combinations but both RFPS and BSPS combinations spend much less runtime than BSPS.

Finally, we want to remark that when *k-NN* is required for classification, the best methods for prototype selection in our experiments were DROPs and GCNN; but when we use other classifiers as LWLR, SVM or Neural Networks, some of our methods had the best performance. This is due to the fact that our methods (and tabu search) can use the same classifier for both selection and classification steps.

As future work, we are going to test some heuristics for reducing the order dependence of our methods.

References

[1] A. Asuncion and D.J. Newman, UCI Machine Learning Repository [http://www.ics.uci.edu/˜mlearn/MLRepository.html] Irvine CA: University of California, School of Information and Computer Science, 2007.

[2] A. Lumini and L. Nanni, A clustering method for automatic biometric template selection, *Pattern Recognition* **39** (2006), 495–497.

[3] B. Spillmann, M. Neuhaus, H. Bunke, E. Pękalska and R.P.W. Duin, in: *Transforming Strings to Vector Spaces Using Prototype Selection*, D.-Y. Yeung et al., eds, SSPR&SPR 2006 LNCS 4109, 2006, pp. 287–296.

[4] C.G. Atkeson, A.W. Moorel and S. Schaal, Locally Weighted Learning, *Artificial Intelligence Review* **11**(1–5) (1997), 11–73.

[5] C. Chien-Hsing, K. Bo-Han and C. Fu, *The Generalized condensed Nearest Neighbor Rule as a Data Reduction Method*, in: *Proceedings of the 18th International Conference on Pattern Recognition*, 2006, 556-559.

[6] C.J. Venmann, M.J.T. Reinders and E. Backer, A maximum variance clustering algorithm, *IEEE Transactions on Pattern Analysis and Machine Intelligence* **24**(9) (2002), 1273–1280.

[7] C.J. Venmann and M.J.T. Reinders, The Nearest Sub-class Classifier: a Compromise between the Nearest Mean and Nearest Neighbor Classifier, *IEEE Transactions on Pattern Analysis and Machine Intelligence* **27**(9) (2005), 1417–1429.

[8] D.E. Rumelhart, G.E. Hinton and R.J. Williams, in: *Learning Internal Representations by Error Propagation*, (Vol. 1), J.L. McClelland and D.E. Rumelhart, eds, Parallel distributed processing, 1986, pp. 318–362.

[9] D.G. Lowe, Similarity Metric Learning for a Variable-Kernel Classifier, *Neural Computation* **7**(1) (1995), 72–85.

[10] D.L. Wilson, Asymptotic Properties of Nearest Neighbor Rules using Edited Data, *IEEE Transactions on Systems, Man and Cybernetics* **2**(3) (1972), 408–421.

[11] D.R. Wilson and T.R. Martínez, Improved Heterogeneous Distance Functions, *Journal of Artificial Intelligence Research* **6**(1) (1997), 1–34.

[12] D.R. Wilson and T.R. Martínez, Reduction Techniques for Instance-Based Learning Algorithms, *Machine Learning* **38** (2000), 257–286.

[13] F. Glover, Future paths for Integer Programming and Links to Artificial Intelligence, *Computers and Operations Research* **5** (1986), 533–549.

[14] 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.

[15] G.W. Gates, The Reduced Nearest Neighbor Rule, *IEEE Transactions on Information Theory* **18**(3) (1972), 431–433.

[16] H. Brighton and C. Mellish, Advances in Instance Selection for Instance-Based Learning algorithms, *Data Mining and Knowledge Discovery* **6**(2) (2002), 153–172.

[17] H. Liu and H. Motoda, On Issues of Instance Selection, *Data Mining and Knowledge Discovery* **6** (2002), 115–130.

[18] I. Tomek, An Experiment with the Edited Nearest-Neighbor Rule, *IEEE Transactions on Systems, Man, and Cybernetics* **6**(6) (1976), 448–452.

[19] J.C. Bezdek and L.I. Kuncheva, Nearest prototype classifier designs: An experimental study, *International Journal of Intelligent Systems* **16**(12) (2001), 1445–1473.

[20] J.C. Riquelme, J.S. Aguilar-Ruíz and M. Toro, Finding representatives patterns with ordered projections, *Pattern Recognition* **36** (2003), 1009–1018.

[21] J. Kittler, in: *Feature Selection and Extraction*, Young and Fu, eds, Handbook of Pattern Recognition and Image Processing, 1986, pp. 203–217.

[22] L.I. Kuncheva, Editing for the k-nearest neighbors rule by a genetic algorithm, *Pattern Recognition Letters, Special Issue on Genetic Algorithms* **16** (1995), 809–814.

[23] L.I. Kuncheva, Fitness functions in editing k-NN reference set by genetic algorithms, *Pattern Recognition* **30** (1997), 1041–1049.

[24] L.I. Kuncheva and J.C. Bezdek, Nearest prototype classification: Clustering, genetic algorithms or random search?, *IEEE Transactions on Systems, Man, and Cybernetics* **C28**(1) (1998), 160–164.
[25] P.A. Devijver and J. Kittler, On the edited nearest neighbor rule, in: *Proceedings of the 5th International Conference on Pattern Recognition* 1980, pp. 72–80.
[26] P.E. Hart, The Condensed Nearest Neighbor Rule, *IEEE Transactions on Information Theory* **14** (1968), 515–516.
[27] P. Pudil, J. Novovicová and J. Kittler, Floating search methods in feature selection, *Pattern Recognition Letters* **15**(11) (1994), 1119–1125.
[28] R.J.S. Aguilar, J.A. Nepomuceno, D.N. Díaz and I. Nepomuceno, A measure for Data Set Editing by Ordered Projections, in: *IEA/AIE 2006 LNAI 4031*, M. Ali and R. Dapoigny, eds, 2006, pp. 1339–1348.
[29] T. Cover and P. Hart, Nearest Neighbor Pattern Classification, *IEEE Transactions on Information Theory* **13** (1967), 21–27.
[30] T. Raicharoen and C. Lursinsap, A divide-and-conquer approach to the pairwise opposite class-nearest neighbor (POC-NN) algorithm, *Pattern Recognition Letters* **26**(10) (2005), 1554–1567.
[31] V. Cerveron and F.J. Ferri, Another move toward the minimum consistent subset: a tabu search approach to the condensed nearest neighbor rule, *IEEE Transactions on Systems, Man and Cybernetics, Part B: Cybernetics* **31**(3) (2001), 408–413.
[32] V. Vapnik, The Nature of Statistical Learning Theory, Springer-Verlag, New York, 1995.