Computación y Sistemas
ISSN: 1405-5546
computacion-y-sistemas@cic.ipn.mx
Instituto Politécnico Nacional
México

Olvera López, José Arturo; Carrasco Ochoa, Jesús Ariel; Martínez Trinidad, José Francisco
Prototype Selection Methods
Computación y Sistemas, vol. 13, núm. 4, junio, 2010, pp. 449-462
Instituto Politécnico Nacional
Distrito Federal, México

RESUMEN DE TESIS DOCTORAL
Prototype Selection Methods
Métodos para la selección de prototipos

Graduated: José Arturo Olvera López
National Institute of Astrophysics, Optics and Electronics
Luis Enrique Erro # 1, Santa María Tonantzintla, C.P. 72840, Puebla, México.
aolvera@inaoep.mx
Graduated on April 16th, 2009

Advisor: Jesús Ariel Carrasco Ochoa
National Institute of Astrophysics, Optics and Electronics
Luis Enrique Erro # 1, Santa María Tonantzintla, C.P. 72840, Puebla, México.
ariel@inaoep.mx

Advisor: José Francisco Martínez Trinidad
National Institute of Astrophysics, Optics and Electronics
Luis Enrique Erro # 1, Santa María Tonantzintla, C.P. 72840, Puebla, México.
fmartine@inaoep.mx

Abstract
In pattern recognition, supervised classifiers assign a class to unseen objects or prototypes. For classifying new prototypes a training set is used which provides information to the classifiers during the training stage. In practice, not all information in a training set is useful therefore it is possible to discard some irrelevant prototypes. This process is known as prototype selection and it is the main topic of this thesis. Through prototype selection the training set size is reduced which allows reducing the runtimes in the classification and/or training stages of classifiers.
Several methods have been proposed for selecting prototypes however their performance is strongly related to the use of a specific classifier and most of the methods spend long time for selecting prototypes when large datasets are processed.
In this thesis, four methods for selecting prototypes, which solve drawbacks of some methods in the state of the art are proposed. The first two methods are based on the sequential floating search and the two remaining methods are based on clustering and prototype relevance respectively.
Keywords: Prototype selection, Data Reduction, Sequential Selection, Border Prototypes.

Resumen
En reconocimiento de patrones, los clasificadores supervisados asignan una clase a nuevos objetos o prototipos. Para clasificar prototipos se usa un conjunto de entrenamiento el cual proporciona información a los clasificadores durante la etapa de entrenamiento. En la práctica, no toda la información en los conjuntos de entrenamiento es útil, por lo que se pueden descartar prototipos irrelevantes. A este proceso se le denomina selección de prototipos, el cual es el tema central de esta tesis.
Mediante la selección de prototipos se reduce el tamaño de los conjuntos de entrenamiento, lo cual permite una reducción en los tiempos de ejecución en las fases de clasificación o entrenamiento de los clasificadores.
Se han propuesto diversos métodos para la selección de prototipos cuyo desempeño depende del uso de un clasificador particular, por otra parte, la mayoría de los métodos para la selección de prototipos son costosos, principalmente cuando se procesan grandes conjuntos de datos.
En esta tesis se presentan cuatro métodos para la selección de prototipos; dos de ellos se basan en la búsqueda secuencial flotante y los dos restantes en agrupamientos y relevancia de prototipos respectivamente.
Palabras clave: Selección de Prototipos, Reducción de Datos, Selección Secuencial, Prototipos Frontera.

1 Introduction

The supervised classification task is a process for assigning a class to unseen prototypes[1]. In order to assign the class to a new prototype, a previously assessed prototype set is provided to the classifiers, 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 prototypes which can be noisy or redundant) therefore a process to discard them from *T* is needed. This selection process is known as prototype selection (Figure 1). The main goal of a prototype selection method is to obtain a set $S \subset T$ such that *S* does not contain superfluous prototypes and $Acc(S) \cong Acc(T)$ where $Acc(X)$ is the classification accuracy obtained using *X* as training set. Through prototype selection, the training set size is reduced, which could be useful for reducing the runtimes in the classification process, particularly for instance-based classifiers.

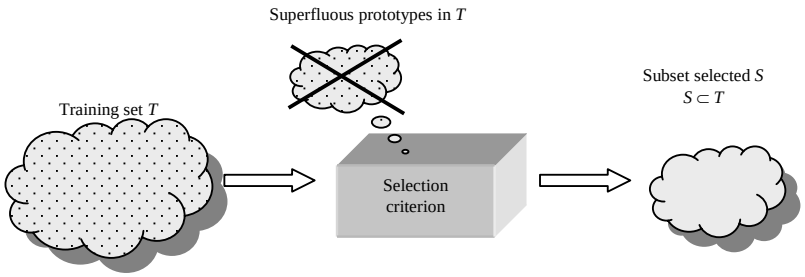

Fig. 1. Prototype selection process

There are two approaches (like in feature selection) for prototype selection: *Wrapper* and *Filter*.

-*Wrapper*. The selection criterion is based on the accuracy obtained by a classifier. In this approach the selection can be done for a specific classifier (*FSC*) or for any classifier (*FAC*). The first one uses a particular classifier during the selection process; the second one is not restricted to use a specific classifier.

-*Filter*. The selection criterion does not use a classifier for discarding prototypes.

In this thesis, four methods for selecting prototypes are presented, two of them are wrapper *FAC* and the other two methods are filter. The wrapper *FAC* methods are based on the sequential floating search; the filter methods are based on clustering and prototype relevance respectively.

2 Prototype selection methods

Several methods have been proposed in the literature for solving the prototype selection problem following either wrapper or filter strategies. In table 1, some of the most successful prototype selection methods (according to their author's results) are shown. In this table, the kind of each method (wrapper *FSC*, *FAC* or filter) and the kind of features they can deal with (numeric, non numeric) are indicated (*).

According to table 1, we can observe that most of the methods are wrapper *FSC*; all of them (excluding *SV-kNNC*) can be applied over mixed features (i.e. numeric and non numeric) and the best methods are *DROP* (*Decremental Reduction Optimization Procedure*) and *GCNN* (*Generalized Condensed Nearest Neighbor*).

The *DROP* methods are based on the concept of *associate*. The *associates* of a prototype *p* are those prototypes such that *p* is one of their *k*-Nearest Neighbors. These methods discard the prototype *p* if its associates can be correctly classified without *p*. According to the authors' results, the best methods are *DROP3* and *DROP5*.

[1] In this thesis, each element in a training set is named prototype.

The *GCNN* method starts with $S = \varnothing$ and its initial step consists in randomly including in *S* one prototype belonging to each class. Then, prototypes are included in *S* according to the *Absorption*(*p*) criterion, which is calculated in terms of the nearest neighbor and nearest enemy (nearest prototype with different class) 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.

Only some methods: *BSE* (*Backward Sequential Edition*), *Genetic algorithms*, and *Tabu Search* (*TS*) are wrapper *FAC* but they are expensive methods. On the other hand, only few filter methods have been proposed and all of them can only be applied over numeric features. *TS* starts over an initial subset S_i (named initial solution) from *T*. *TS* finds a subset $S \subset S_i$ such that *S* contains the prototypes with the best classification accuracy. In order to decide the prototypes added to *S*, *TS* evaluates all the neigboring subsets to S_i i. e. the prototype sets that differ from S_i by just one element.

Table 1. Characteristics of some prototype selection methods

Method	Wrapper		Filter	Features	
	FSC	*FAC*		Numeric	Non numeric
CNN (Hart, 1968)	*			*	*
ENN (Wilson, 1972)	*			*	*
SNN (Ritter et al., 1975)	*			*	*
Multiedit (Devijver and Kittler, 1980)	*			*	*
IB (Aha, 1991)	*			*	*
DROP (Wilson and Martínez, 2000)	*			*	*
ICF (Brighton and Mellish, 2002)	*			*	*
SETRED (Li et al., 2005)	*			*	*
GCNN (Chien-Hsing et al., 2006)	*			*	*
SV-kNNC (Srisawat et al., 2006)	*			*	*
CNNDD (Angiulli, 2007)	*			*	*
Genetic Algorithms (Kuncheva and Bezdek, 1998; 2001)		*		*	*
Tabu search (Zhang and Sun, 2002)		*		*	*
BSE (Olvera et al., 2005)		*		*	*
SCE (Eick et al., 2004)			*	*	
POC-NN (Raicharoen and Lursinsap, 2005)			*	*	
kd-trees (Narayan et al., 2006)			*	*	
CLU (Lumini and Nani, 2006)			*	*	

Some authors (Bezdek and Kuncheva, 2001; Liu and Motoda, 2002; Spillman, 2006) have suggested the idea of using clustering for prototype selection; this idea consists in: after splitting *T* in *n* clusters, the selected prototypes will be the centers of the *n* clusters. The *CLU* prototype selection method is based on this idea.

Based on these drawbacks of prototype selection methods, this thesis is focused on proposing wrapper *FAC* and filter methods. In particular four methods are proposed: *Restricted Floating Prototype Selection* (*RFPS*), *Restricted Floating Prototype Selection-Inverse* (*RFPS-Inv*), *Prototype Selection by Clustering* (*PSC*), and *Prototype Selection by Relevance* (*PSR*). These methods are described in the following sections.

2.1 Restricted Floating Prototype Selection (RFOS)

This method is based on the idea of the Sequential Floating Search (*SFS*) (Pudil et al., 1994), which allows backtracking, i.e. reconsiders the inclusion/exclusion (in the prototype subset) of prototypes previously discarded/included. *SFS* consists in applying conditional inclusion/exclusion steps after each exclusion/inclusion in

S. This kind of search (like the sequential search) can be done in backward and forward directions.

The backward *SFS* consists in applying a number of inclusion steps after each exclusion step 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 pre-processing step (in order to reduce the training set size) before applying our prototype selection method based on the *SFS*. Our method named *Restricted Floating Prototype Selection* (*RFPS*) is restricted since it applies the floating search just once in each direction: conditional exclusion followed by the conditional inclusion.

RFPS starts applying a pre-processing step followed by the conditional exclusion and finally the conditional inclusion is applied (figure 2). The conditional exclusion sequentially discards prototypes of the training set. This process analyzes the classification contribution of each prototype and at each step it excludes the prototype with the smallest contribution for the subset quality, in terms of the accuracy of a classifier. The conditional inclusion consists in analyzing the prototypes discarded from *T* (prototypes discarded in both pre-processing and conditional exclusion) for including in *S* those that their inclusion improves the classification accuracy, that is, a prototype *p* will be included if the classification accuracy obtained using *p* is better than the one obtained without it.

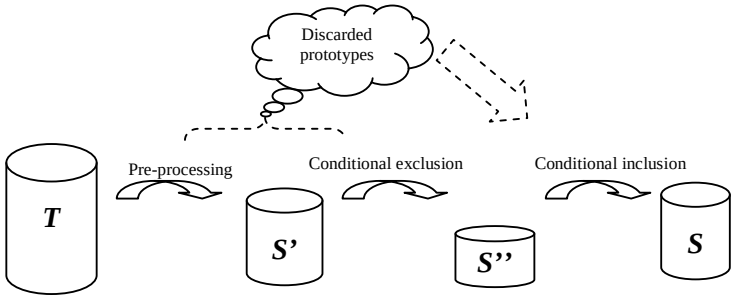

Fig. 2. *RFPS* method for prototype selection

This restricted floating method can be done in the inverse direction (*RFOS-Inv*), that is, first applying the conditional inclusion (from the prototype discarded in the pre-processing) followed by the conditional exclusion (figure 3).

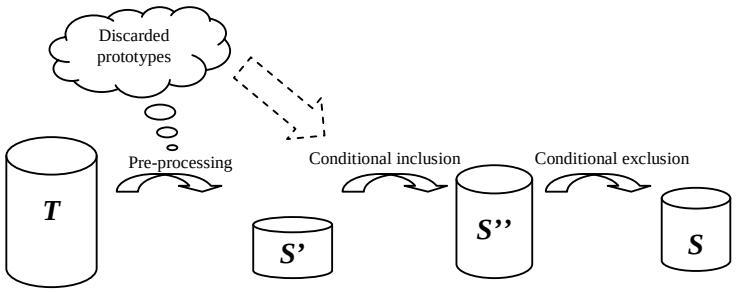

Fig. 3. *RFPS-Inv* method for prototype selection

2.2 Prototype Selection by Clustering (PSC)

In a training set, the border prototypes of a class are located in a region where there are prototypes from different classes. These prototypes contain useful information for preserving the class discrimination regions (Wilson and Martínez, 2000; Brighton and Mellish, 2002). On the other hand, interior prototypes of a class (those that are not

border) could be discarded without affecting the classification quality of the set. Although interior prototypes could be discarded, some of them are necessary for representing interior regions.

In this section, we introduce a prototype selection method named *PSC* (*Prototype Selection by Clustering*) which is based on clustering. When clusters are created from a training set they can be either homogeneous (all the prototypes belong to the same class) or non homogeneous (the prototypes belong to two or more classes). In order to find border prototypes *PSC* analyzes the non homogeneous clusters.

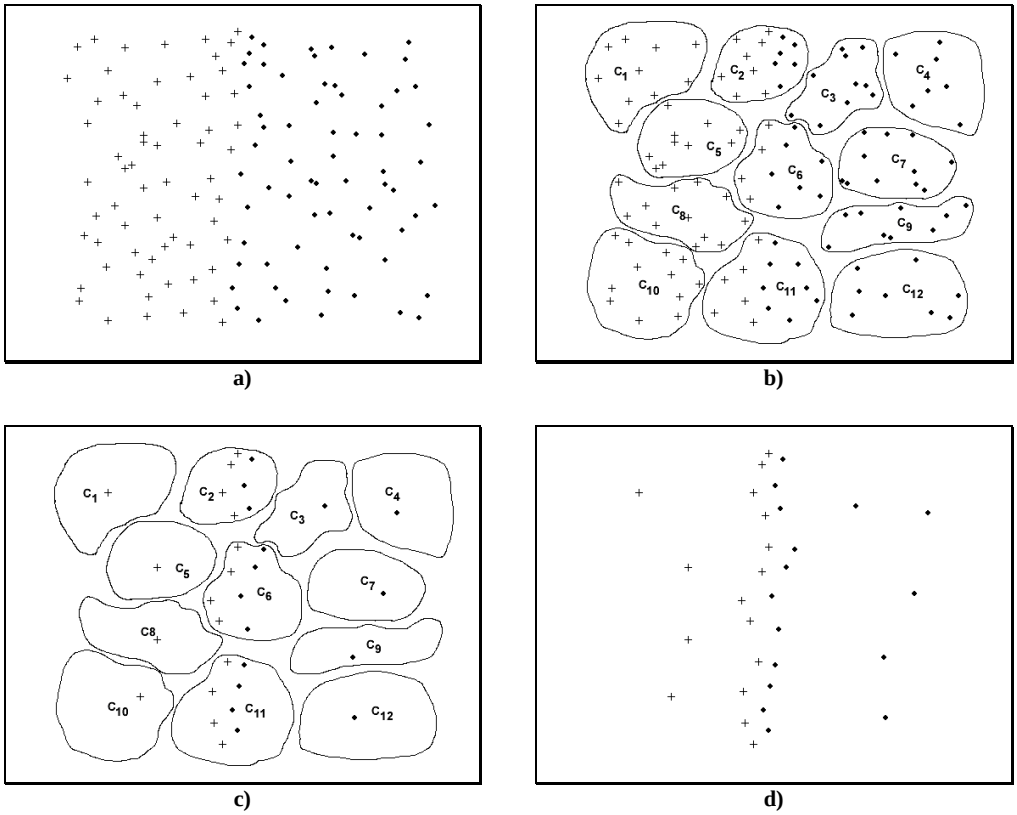

Fig. 4. a) Dataset with classes "+" and "●". **b)** Clusters created from the dataset. **c)** Prototypes selected in each cluster. **d)** Subset selected by *PSC*

The *PSC* works as follows: first, clusters are generated from a training set, after that, *PSC* works over each cluster. If the cluster C_j is non homogeneous then C_j contains some prototypes located at critical regions, i.e. border prototypes. In order to find the border prototypes, *PSC* finds the majority class among the cluster prototypes. The border prototypes in the majority class are those prototypes that are the most similar to a prototype belonging to a different class, and the border prototypes in the other classes are those prototypes that are the most similar to each border prototype in the majority class. If C_j is homogeneous then the prototypes in C_j are interior prototypes and they could be discarded from T without affecting the classification accuracy. Therefore, *PSC* finds the representative prototype (the prototype that is in average the most similar to all other prototypes in C_j) and discards the remaining prototypes so that C_j is reduced to its representative prototype. Finally, the prototypes selected by *PSC* are the representative ones from each homogeneous cluster and the border from each non homogeneous cluster.

In order to illustrate the *PSC* idea let us consider the dataset shown in figure 4a which is a bi-dimensional dataset with prototypes belonging to the classes "+" and "•". In figure 4b, the clusters ($C_1 \ldots C_{12}$) created from the dataset are depicted, the non homogeneous clusters are C_2, C_6 and C_{11} whereas the remaining clusters are homogeneous. In the clusters C_6 and C_{11} the minority class is "+", then the border prototypes in the most frequent class (•) are the most similar prototypes to each minority class prototype (+). On the other hand, the border prototypes in class "+" are the most similar (belonging to class "+") to each border in class "•".

The same process described before is applied to cluster C_2 where the minority class is "•". The prototypes selected in each cluster are depicted in figure 4c and the subset selected by *PSC* is depicted in figure 4d.

2.3 Prototype Selection by Relevance (PSR)

In a training set, among the prototypes belonging to the same class only some of them could be considered as representative or relevant prototypes in the class. For prototype selection, it makes sense to select the most relevant prototypes and discard the remaining. In this thesis, the relevance of each prototype is computed in terms of the average similarity that it has with all other prototypes in the same class thus the most similar to all other prototypes, the most relevant in the class.

In this section, we present the *PSR* (*Prototype Selection by Relevance*) method which computes the relevance of each prototype and retains the most relevant ones. Additionally, in order to preserve the discrimination regions between classes, *PSR* also retain border prototypes which are found through the most relevant prototypes.

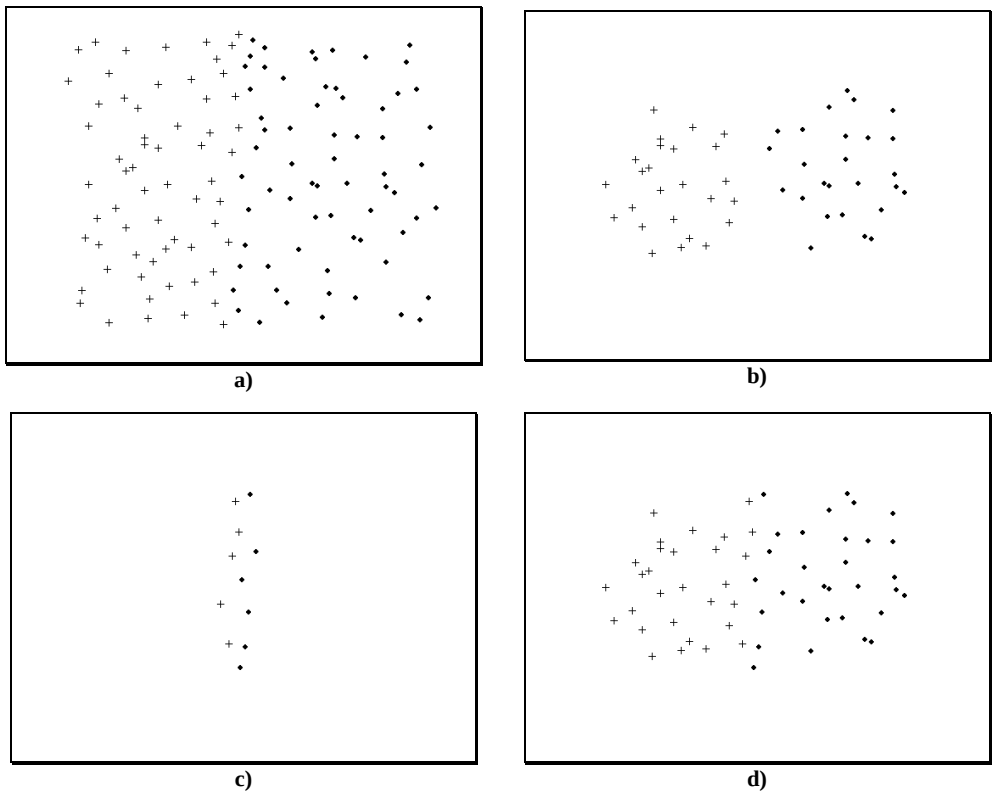

Fig. 5. a) Dataset with classes "+" and "●". **b)** 30% of the most relevant prototypes per class. **c)** Border prototypes found through figures 5a and 5b**. d)** Subset selected by *PSR*

PSR starts computing the relevance weight (average similarity) of each prototype in the training set. Once the prototype relevance weights have been computed, for each class *i*, the *r* most relevant prototypes are chosen and through them some border prototypes are selected. It can be noticed that depending on the training set and/or the relevance function, some relevant prototypes could also be border prototypes. *PSR* finds the border prototypes as follows: for each prototype of the *r* chosen, the most similar prototype belonging to each class different from *i* is selected as a border prototype. Finally, *S* contains the *r* most relevant prototypes found through the relevance criterion described above and the border prototypes found from them.

To illustrate the *PSR* selection process let us consider the dataset shown in figure 5a (the same shown in figure 4a). Figure 5b shows the 30% of the most relevant prototypes per class (according to the average similarity). Through the prototypes depicted in figures 5a-5b, the border prototypes are selected (figure 5c) and finally the subset selected by *PSR* is depicted in the figure 5d.

3 Experimental results

In this section, we report some experimental results obtained by applying our methods presented in section 2 over different datasets taken from the UCI repository (Asunción and Newman, 2007); we used numeric (*Glass, Iris, Liver, Wine*), and mixed (*Bridges, Echocardiogram, Hearth Cleveland, Hepatitis, Zoo*) datasets. In our experiments we used *10-fold cross validation* and the average over the ten trials is reported. The average results over all the tested datasets are depicted on scatter graphs of classification accuracy (*Acc*) versus storage (*Str*), where *Str* is the percentage of prototypes retained by each method with respect to the training set size, that is *Str*=100|S|/|T|. In these graphs, the Pareto front is shown, which is formed by those methods such that they are the best in either accuracy or storage.

In our experimentation, we included some of the most successful prototype selection methods, in particular *DROP3, DROP5, GCNN, TS* and *CLU* methods.

For *PSC, PSR* and *CLU* methods it is necessary to fix the initial parameter: number of cluster to create (*PSC* and *CLU*) and the number *r* of relevant prototypes to choose in the initial phase (*PSR*). We carried out some experiments varying the value of the initial parameters. The results are reported in tables 2-3. Table 2 shows the classification accuracy obtained using as training set (for *k-NN, k=3*) the selected prototypes by *PSC* and *CLU* creating different number of clusters *n*, in particular the values *n= 4c, 6c, 8c 10c* and *12c* (where *c* is the number of classes in the training set) were tested. In this table, the *Zoo* dataset was omitted because *n=12c* is higher than the total prototypes in the training set.

Table 2. Classification accuracy obtained by *PSC* and *CLU* creating different number of clusters in the initial phase

Dataset	Number of clusters									
	n=4c		*n=6c*		*n=8c*		*n=10c*		*n=12c*	
	CLU	**PSC**	**CLU**	**PSC**	**CLU**	**PSC**	**CLU**	**PSC**	**CLU**	**PSC**
Bridges	51.63	51.63	53.54	56.54	58.38	59.45	61.27	61.09	61.07	60.03
Echocardiogram	85.90	86.42	90.71	86.42	94.10	91.42	90.71	85.53	88.10	85.67
Glass	53.26	56.21	52.83	59.09	55.58	56.16	56.08	63.52	57.63	56.72
Heart Cleveland	74.61	71.26	73.00	73.27	75.29	72.26	76.33	74.00	75.18	73.54
Hepatitis	77.50	73.12	75.00	75.37	75.87	79.29	75.66	75.54	76.45	76.95
Iris	84.00	84.66	86.66	94.66	87.33	88.88	89.00	90.00	85.43	94.00
Liver	51.57	55.32	52.18	55.36	52.58	54.54	51.87	54.78	53.97	55.36
Wine	85.91	88.30	88.11	92.67	87.37	88.41	88.30	88.19	89.52	91.00
Average	**70.55**	**70.87**	**71.50**	**74.17**	**73.31**	**73.80**	**73.65**	**74.08**	**73.42**	**74.16**

Table 3. Classification accuracy obtained by *PSR* selecting different number of relevant prototypes in the initial phase

Dataset	Percentage (*r*) of relevant prototypes per class				
	r=10%	*r=20%*	*r=30%*	*r=40%*	*r=50%*
Bridges	48.90	50.90	57.63	57.54	61.18
Echocardiogram	83.21	83.92	90.53	90.53	89.28
Glass	61.12	64.37	64.85	65.80	69.17
Heart Cleveland	75.81	75.84	79.18	78.20	77.21
Hepatitis	74.95	82.58	83.16	81.91	79.91
Iris	86.66	88.66	91.33	89.33	92.00
Liver	61.96	62.85	63.77	64.92	64.61
Wine	94.00	92.12	92.18	92.74	93.00
Zoo	93.33	93.33	93.33	93.33	93.33
Average	**75.55**	**77.17**	**79.55**	**79.37**	**79.97**

Table 3 shows the accuracy obtained by *PSR* selecting different number *r* of relevant prototypes per class, the tested values for *r* were *r= 10%, 20%, 30%, 40%* and *50%* of the prototypes in each class.

According to table 2, in the average case, the best value for the *n* parameter in *PSC* is *n=6c* and the best one for *CLU* is *n=10c*. We used these values in the experiments reported in the following sections. For *PSR* we decided to use *r=30%* since its accuracy is similar to the best obtained (*r=50%*) but with less storage.

3.1 RFPS and RFPS-Inv results

The results obtained by applying *RFPS* and *RFPS-Inv* over the datasets are reported in tables 4-5 and figure 6. In order to compare our methods, the results obtained by *DROPs*, *GCNN* and *Tabu Search* (*TS*) are also reported; the classifier used was *k-NN* (*k=3*, the best value for the *DROPs* according to their authors' results). In these tables, the accuracy obtained by the original training set without applying prototype selection (*Orig*) is shown; *ENN+RFPS*, *DROP3+RFPS* and *DROP5+RFPS* mean that *ENN*, *DROP3*, *DROP5* methods were used in the pre-processing step of the restricted floating methods.

From the obtained results, the best methods in accuracy were *DROP3*, *DROP5* and *ENN+RFPS*. Among our methods *RFPS* outperformed to *RFPS-Inv* and both of them were better than *TS*; the best in accuracy were *ENN+RFPS*, *DROP3+RFPS* and the last is part of the Pareto front.

Table 4. Accuracy (*Acc*) and Storage (*Str*) results obtained by: Original training set (*Orig*), *RFPS* and *RFPS-Inv*

Dataset	Orig.		ENN+RFPS		DROP3+RFPS		DROP5+RFPS		ENN+RFPS-Inv		DROP3+RFPS-Inv		DROP5+RFPS-Inv	
	Acc.	Str.	Acc.	Str.	Acc.	Str.	Acc.	Str.	Acc.	Str.	Acc.	Str.	Acc.	Str.
Bridges	66.09	100	60.00	48.10	56.45	14.28	53.63	23.71	51.54	26.80	55.45	12.37	54.54	19.58
Echocardiogram	95.71	100	93.23	86.56	91.96	9.85	87.67	8.16	93.21	11.94	87.85	8.95	89.10	7.46
Glass	71.42	100	69.43	29.34	64.48	25.75	67.74	26.11	58.35	19.47	43.50	23.36	55.49	16.45
Heart Cleveland	82.49	100	79.52	22.41	77.21	12.61	79.22	12.53	78.74	13.60	77.90	7.77	76.48	9.12
Hepatitis	79.29	100	79.00	18.71	78.20	8.38	76.66	8.67	77.95	11.47	78.45	6.66	74.08	5.23
Iris	94.66	100	93.33	10.07	93.00	9.92	93.33	10.00	80.66	5.33	92.66	7.18	86.00	6.29
Liver	65.22	100	59.98	33.68	61.70	16.94	60.03	19.64	58.84	21.80	59.75	19.25	60.30	19.54
Wine	94.44	100	93.63	10.23	94.44	8.17	93.85	8.30	90.00	4.36	88.23	5.18	91.04	4.42
Zoo	93.33	100	91.33	71.14	91.33	14.81	91.11	14.93	91.11	15.92	90.00	13.58	80.00	14.19
Average	**82.52**	**100**	**79.94**	**36.70**	**78.75**	**13.41**	**78.14**	**14.67**	**75.60**	**14.52**	**74.87**	**11.59**	**74.11**	**11.36**

Table 5. Accuracy (*Acc*) and Storage (*Str*) results obtained by: Original training set (*Orig*), *DROPs*, *GCNN* and *TS*

Dataset	Orig.		DROP3		DROP5		GCNN		TS	
	Acc.	Str.	Acc.	Str.	Acc.	Str.	Acc.	Str.	Acc.	Str.
Bridges	66.09	100	56.36	14.78	62.82	20.66	68.20	88.20	45.90	18.94
Echocardiogram	95.71	100	92.86	13.95	98.42	14.87	93.39	22.67	85.71	7.46
Glass	71.42	100	66.28	24.35	62.16	25.91	69.61	61.62	62.59	15.98
Heart Cleveland	82.49	100	78.89	11.44	79.87	14.59	67.63	9.09	72.63	4.54
Hepatitis	79.29	100	78.13	11.47	75.42	15.05	60.66	17.75	74.33	5.73
Iris	94.66	100	95.33	15.33	94.00	12.44	96.00	38.00	70.66	6.50
Liver	65.22	100	67.82	26.83	63.46	30.59	66.09	83.70	64.13	5.21
Wine	94.44	100	94.41	15.04	93.86	10.55	94.44	78.89	79.44	6.10
Zoo	93.33	100	90.00	20.37	95.56	18.77	95.55	26.17	88.88	14.12
Average	**82.52**	**100**	**80.01**	**17.06**	**80.22**	**18.16**	**79.06**	**47.34**	**71.59**	**9.40**

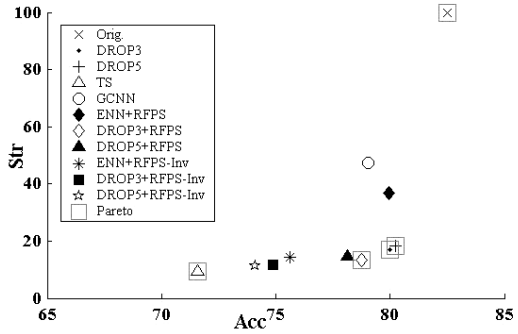

Fig. 6. Scatter graphic from the average results shown in tables 4-5

In the above results the used classifier was *k-NN*; in order to know the performance of the prototype selection methods using other classifiers, the subsets selected by each method were used as training for *C4.5* (decision trees) and *Naïve Bayes* (*NB*). The results obtained are depicted in figures 7-8. We can notice that using these classifiers, the *DROPs* are not as good in accuracy as for *k-NN*. In both classifiers, *ENN+RFPS* and *ENN+RFPS-Inv* are on the Pareto front and *ENN+RFPS* was the best in accuracy.

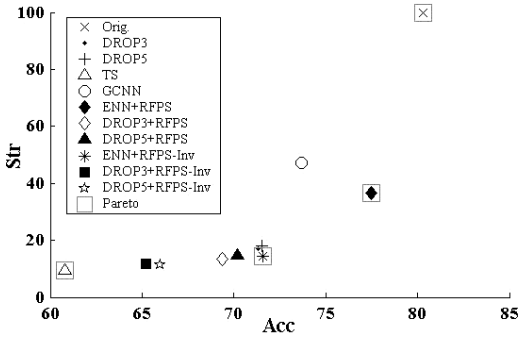

Fig.7. Results obtained using the subsets selected by *DROPs, TS, GCNN, RFOS, RFOS-Inv* as training for *C4.5*

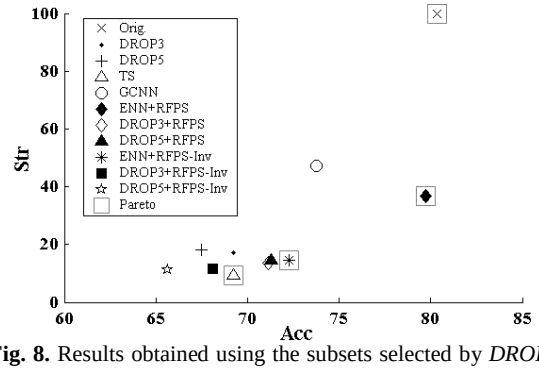

Fig. 8. Results obtained using the subsets selected by *DROPs, TS, GCNN, RFOS, RFOS-Inv* as training for *NB*

Our restricted prototype selection methods and *TS* are wrapper *FAC* then they can use any classifier during the selection process; in figures 9-10, the results of applying *RFPS* (the best of our restricted floating methods) and *TS* results using *C4.5* and *NB* during the selection process are shown.

According to figures 9-10, using *C4.5* and *NB* in the selection process *TS* was outperformed by *RFPS* (excluding *DROP3+RFPS* for *C4.5*) and for *NB*, the accuracy obtained by the subsets selected by *ENN+RFPS* was better than those obtained by the original training set.

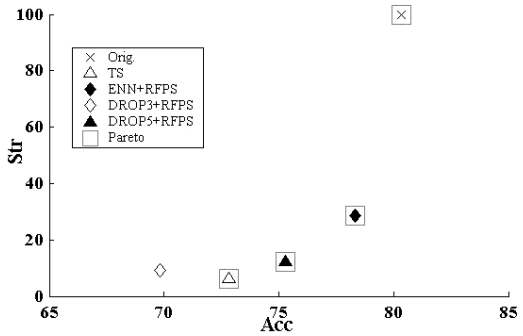

Fig. 9. Results obtained using *C4.5* in the selection process

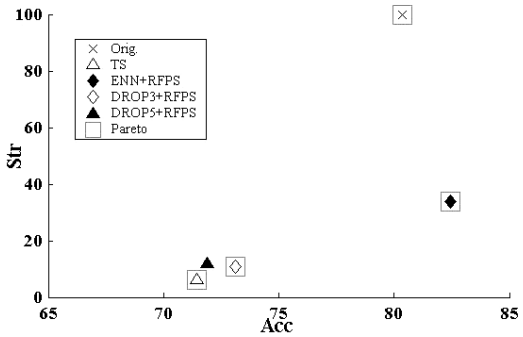

Fig. 10. Results obtained using *NB* in the selection process

3.2 PSC and PSR results

In this section, the results obtained by *PSC* and *PSR* are reported; for comparison we have included *DROP3*, *DROP5*, *GCNN* and *CLU* methods again. The obtained results using *k-NN* are shown in table 6 and figure 11.

Table 6. Accuracy (*Acc*) and Storage (*Str*) results obtained by: Original training set (*Orig*), *DROPs*, *GCNN*, *CLU*, *PSC* and *PSR*

Dataset	Orig.		DROP3		DROP5		GCNN		CLU		PSC		PSR	
	Acc.	Str.	Acc.	Str.	Acc.	Str.	Acc.	Str.	Acc.	Str.	Acc.	Str.	Acc.	Str.
Bridges	66.09	100	56.36	14.78	62.82	20.66	68.20	88.20	61.27	63.68	56.54	42.86	57.63	47.79
Echocardiogram	95.71	100	92.86	13.95	98.42	14.87	93.39	22.67	90.71	30.03	86.42	19.82	90.53	37.68
Glass	71.42	100	66.28	24.35	62.16	25.91	69.61	61.62	56.08	31.15	59.09	38.45	64.85	42.36
Heart Cleveland	82.49	100	78.89	11.44	79.87	14.59	67.63	9.09	76.33	18.33	73.27	22.51	79.18	36.96
Hepatitis	79.29	100	78.13	11.47	75.42	15.05	60.66	17.75	75.66	14.33	75.37	7.95	83.16	33.40
Iris	94.66	100	95.33	15.33	94.00	12.44	96.00	38.00	89.00	22.22	94.66	20.45	91.33	38.07
Liver	65.22	100	67.82	26.83	63.46	30.59	66.09	83.70	51.87	6.44	55.36	45.87	63.77	35.55
Wine	94.44	100	94.41	15.04	93.86	10.55	94.44	78.89	88.30	37.03	92.67	37.15	92.18	42.94
Zoo	93.33	100	90.00	20.37	95.56	18.77	95.55	26.17	90.00	76.41	92.22	41.48	93.33	51.11
Average	**82.52**	**100**	**80.01**	**17.06**	**80.22**	**18.16**	**79.06**	**47.34**	**75.47**	**33.29**	**76.18**	**30.73**	**79.55**	**40.65**

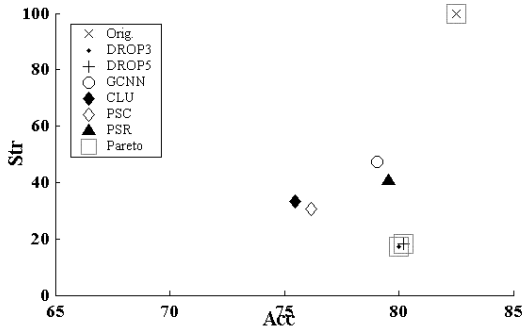

Fig. 11. Scatter graphic from the average results shown in table 6

From these results, we can notice that the best methods were the *DROPs* but it was expected since they are based on *k-NN*. In these results, *PSC* was better than *CLU*.

Like in the previous experiments, in figures 12-13 we report the results obtained using the subsets selected by the prototype selection methods as training for *C4.5* and *NB* classifiers. According to these figures, for both classifiers *PSR* and *PSC* were the best methods in accuracy respectively. Between them, the best one in accuracy was *PSR* and he best one in retention was *PSC.*

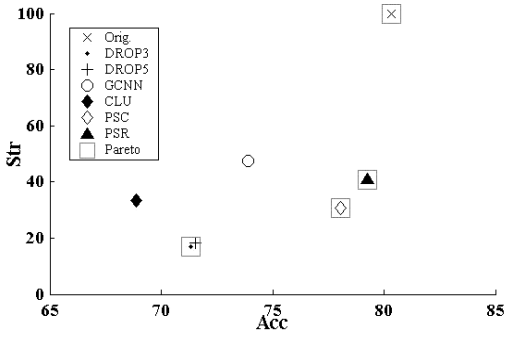

Fig. 12. Results obtained using the subsets selected by *DROPs*, *GCNN*, *CLU*, *PSC* and *PSR* as training for *C4.5*

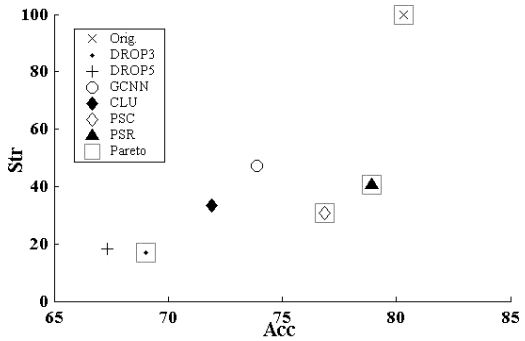

Fig. 13. Results obtained using the subsets selected by *DROPs*, *GCNN*, *CLU*, *PSC* and *PSR* as training for *NB*

Another experiment reported in this document is related to the runtimes spent by each prototype selection method. In this experiment we measured the runtimes spent by each method over different training set sizes constructed from the prototypes in the *Shuttle- Statlog* dataset (58000 prototypes, 9 features, 7 classes). The results are shown in figures 14-15. In figure 14, some points are not depicted because they are out of the vertical axis scale. We have not included *RFOS* because it is an expensive method (like *TS* and some other wrapper methods) when it is applied over medium/large datasets. Based on the runtimes from figures 14-15, clearly *CLU*, *PSC* and *PSR* are the fastest methods. The runtimes spent by *CLU* and *PSC* are very similar but *CLU* is outperformed in accuracy by both *PSC* and *PSR* (according to the results reported in previous tables).

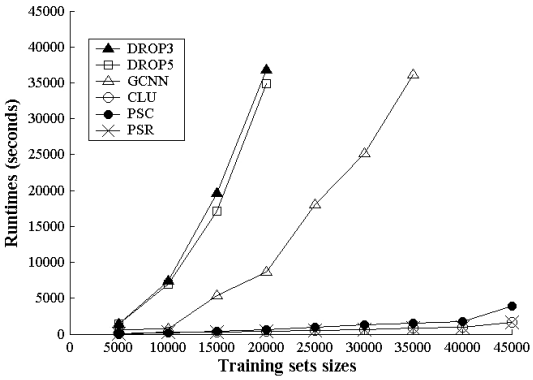

Fig. 14. Runtimes spent by *DROPs*, *GCNN*, *CLU*, *PSC* and *PSR* for different training sets built from the *Shuttle- Statlog* dataset

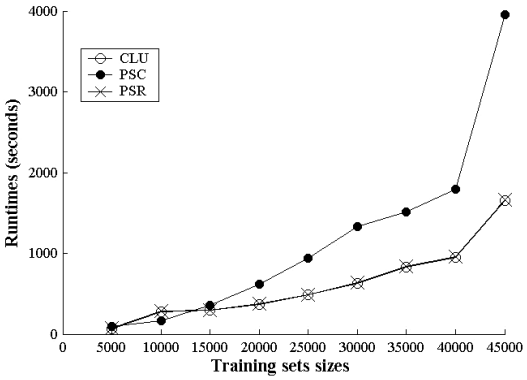

Fig. 15. Runtimes spent by *CLU*, *PSC* and *PSR* for different training sets built from the *Shuttle -Statlog* dataset

3.3 Comparison among the proposed methods

The last experiment consisted in comparing the methods introduced in this thesis. In figures 16-18 the results obtained by *RFOS*, *RFOS-Inv*, *PSC*, *PSR* using *k-NN*, *C4.5* and *NB* are shown. According to these figures, the best method in accuracy (excepting for *C4.5*) was *ENN+RFPS* followed by *PSR* and *PSC*. Among our methods, the best one in storage was *DROP5+RFPS-Inv*.

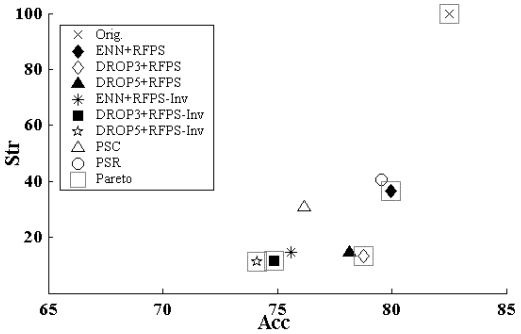

Fig. 16. Scatter graphic from the average results obtained by the proposed methods using *k-NN*

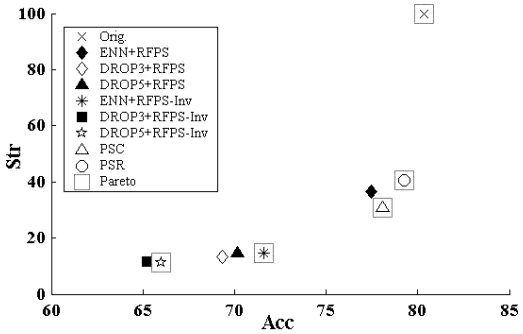

Fig. 17. Scatter graphic from the average results obtained by the proposed methods using *C4.5*

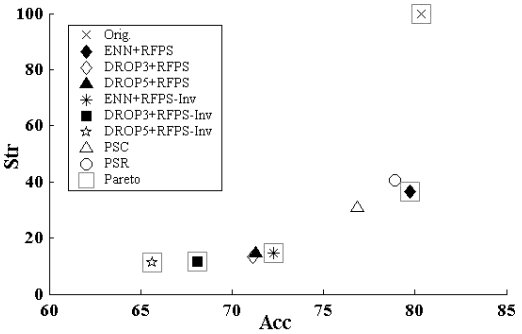

Fig. 18. Scatter graphic from the average results obtained by the proposed methods using *NB*

4 Conclusions

In this thesis, four prototype selection methods were introduced: *RFOS*, *RFOS-Inv* (wrapper *FAC* methods), *PSC* and *PSR* (filter methods). They were compared against other successful prototype selection methods (*DROPs*, *GCNN*, *TS* and *CLU*) and from our results we can conclude that our methods are competitive in accuracy when the *k-NN* classifier is used; in addition our methods are the best when *C4.5* and *NB* classifiers are used.

Based on the experimentation, among our methods, the best one in accuracy was *ENN+RFPS* but it is mainly recommended to be applied over small datasets because it is an expensive method (like other wrapper methods such as *TS*). On the other hand, when medium/large datasets are processed, *PSC* and *PSR* were clearly faster than all other methods. *PSC* is faster than *PSR* but the last is better in accuracy than *PSC*.

PSC and *PSR* require initial parameters given by the user: the number of clusters to create and the percentage of relevant prototypes per class respectively; as future work we are interested in proposing techniques for automatically fixing these parameters.

References

1 **Aha, D. W. (1991).** Instance based learning algorithms. *Machine Learning*, 6 (1), 37-66.

2 **Angiulli, F. (2007).** Condensed Nearest Neighbor Data Domain Description. *IEEE Transaction on Pattern Analysis and Machine Intelligence*, 29 (10), 1746-1758.

3 **Asunción, A., & Newman, D. J. (2007).** *UCI Machine Learning Repository*. Irvine, CA. University of California, School of Information and Computer Science. [http://www.ics.uci.edu/~mlearn/MLRepository.html].

4 **Bezdek, J. C., & Kuncheva, L. I. (2001).** Nearest Prototype Classifier Designs: An Experimental Study. *International Journal of Intelligent Systems*, 16 (12), 1445-1473.

5 **Brighton, H., & Mellish, C. (2002).** Advances in Instance-Based Learning Algorithms. *Data Mining and Knowledge Discovery*, 6, 153-172.

6 **Chien-Hsing, C., Bo-Han, K., & Fu, C. (2006).** The Generalized Condensed Nearest Neighbor Rule as a Data Reduction Method. *18th International Conference on Pattern Recognition*, Hong Kong, 2, 556-559.

7 **Devijver, P. A., & Kittler, J. (1980).** On the edited nearest neighbor rule. *5th International Conference on Pattern Recognition*, Miami, Florida, USA, 72-80.

8 **Eick, C. F., Zeidat, N., & Vilalta, R. (2004).** Using Representative-Based Clustering for Nearest Neighbor Dataset Editing. *4th IEEE International Conference on Data Mining*, Brighton, UK, 375-378.

9 **Hart, P. E. (1968).** The Condensed Nearest Neighbor Rule. *IEEE Transactions on Information Theory*, 14, 515-516.

10 **Kuncheva, L. I., & Bezdek, J. C. (1998).** Nearest prototype classification: clustering, genetic algorithms or random search?. *IEEE Transactions on Systems, Man and Cybernetics*, C28(1), 160 - 164.

11 **Li, M. & Zhi-Hua, Z. (2005).** SETRED: Self-training with Editing. *Pacific-Asia Conference on Knowledge Discovery and Data Mining PAKDD 2005. Lecture Notes in Artificial Intelligence*, 3518, 611-621.

12 **Liu, H., & Motoda, H. (2002).** On Issues of Instance Selection. *Data Mining and Knowledge Discovery*, 6, 115-130.

13 **Lumini, A., & Nanni, L. (2006).** A clustering method for automatic biometric template selection. *Pattern Recognition*, 39, 495-497.

14 **Narayan, B. L., Murthy, C. A. & Pal, S. K. (2006).** Maxdiff kd-trees for data condensation. *Pattern Recognition Letters*, 27, 187-200.

15 **Olvera-López, J. A., Carrasco-Ochoa, J. A. & Martínez-Trinidad, J. F. (2005).** Sequential Search for Decremental Edition. *Intelligent Data Engineering and Automated Learning IDEAL 2005. Lecture Notes in Computer Science*, 3578, 280-285.

16 **Pudil, P., Ferri, F.J., Novovicová, J. & Kittler, J. (1994).** Floating Search Methods for Feature Selection with Nonmonotonic Criterion Functions. *12th International Conference on Pattern Recognition*, Jerusalem, Israel, 279-283.

17 **Raicharoen, T. & Lursinsap, C. (2005).** A divide-and-conquer approach to the pairwise opposite class-nearest neighbor (POC-NN) algorithm. *Pattern Recognition Letters*, 26(10), 1554-1567.

18 **Ritter, G. L., Woodruff, H.B., Lowry, S. R. & Isenhour, T. L. (1975).** An Algorithm for a Selective Nearest Neighbor Decision Rule. *IEEE Transactions on Information Theory*, 21(6), 665-669.

19 **Spillmann, B., Neuhaus, M., Bunke, H., Pękalska, E. & Duin, R. P. W. (2006).** Transforming Strings to Vector Spaces Using Prototype Selection. *International workshop on Structural and Syntactic Pattern Recognition SSPR&SPR 2006. Lecture Notes in Computer Science*, 4109, 287–296.

20 **Srisawat, A., Phienthrakul, T., & Kijsirikul, B. (2006).** SV-kNNC: An Algorithm for Improving the Efficency of k-Nearest Neighbor. *Pacific Rim International Conference on Artificial Intelligence PRICAI 2006. Lecture Notes in Artificial Intelligence*, 4099, 975-979.

21 **Wilson, D. L. (1972).** Asymptotic Properties of Nearest Neighbor Rules Using Edited Data. *IEEE Transactions on Systems, Man, and Cybernetics*, 2(3), 408-421.

22 **Wilson, D. R., & Martínez, T. R. (2000).** Reduction Techniques for Instance-Based Learning Algorithms. *Machine Learning*, 38, 257-286.

23 **Zhang, H., & Sun, G. (2002).** Optimal reference subset selection for nearest neighbor classification by tabu search. *Pattern Recognition*, 35, 1481-1490.

José Arturo Olvera López *Received his B. S. degree in Computer Science from the faculty of Computer Science of the Autonomous University of Puebla, Mexico in 2004; his M.Sc. degree in Computer Science from the National Institute of Astrophysics, Optics and Electronics (INAOE), Mexico, in 2006 and his Ph.D. degree in Computer Science from INAOE, Mexico, in 2009. Currently, he is a full time researcher in the Faculty of Computer Science at the Autonomous University of Puebla (BUAP), Mexico. His Research interests are Logical Combinatorial Pattern Recognition, Machine Learning and Prototype Selection.*

Jesús Ariel Carrasco Ochoa *Received his Ph.D. degree in Computer Science form the Center for Computing Research of the National Polytechnic Institute (CIC-IPN), Mexico, in 2001. Currently, he is a full time researcher at the National Institute for Astrophysics, Optics and Electronics (INAOE) of Mexico. His current research interests include Sensitivity Analysis, Logical Combinatorial Patter Recognition, Testor Theory, Feature Selection, Prototype Selection and Clustering.*

José Francisco Martínez Trinidad *Received his B.S. degree in Computer Science from Physics and Mathematics School of the Autonomous University of Puebla (BUAP), Mexico in 1995, his M.Sc. degree in Computer Science from the faculty of Computers Science of the Autonomous University of Puebla, Mexico in 1997 and his Ph.D. degree in the Center for Computing Research of the National Polytechnic Institute (CIC, IPN), Mexico in 2000. Professor Martinez-Trinidad edited/authored four books and over fifty journal and conference papers, on subjects related to Pattern Recognition.*