Prototype Selection Via Prototype Relevance

J. Arturo Olvera-López, J. Ariel Carrasco-Ochoa, and J. Fco. Martínez-Trinidad

Computer Science Department
National Institute of Astrophysics, Optics and Electronics
Luis Enrique Erro No. 1, Sta. María Tonantzintla, Puebla, CP: 72840, Mexico
{aolvera,ariel,fmartine}@ccc.inaoep.mx

Abstract. In Pattern recognition, the supervised classifiers use a training set *T* for classifying new prototypes. In practice, not all information in *T* is useful for classification therefore it is necessary to discard irrelevant prototypes from *T*. This process is known as prototype selection, which is an important task for classifiers since through this process the time in the training and/or classification stages could be reduced. Several prototype selection methods have been proposed following the Nearest Neighbor (*NN*) rule; in this work, we propose a new prototype selection method based on the prototype relevance and border prototypes, which is faster (over large datasets) than the other tested prototype selection methods. We report experimental results showing the effectiveness of our method and compare accuracy and runtimes against other prototype selection methods.

Keywords: Prototype selection, border prototypes, supervised classification, data reduction.

1 Introduction

In Pattern Recognition, supervised classification is a process that assigns a class or label to new prototypes using a set of previously assessed prototypes, commonly, this set is called training set *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. The main goal of a prototype selection method is to obtain a set $S \subset T$ such that *S* does not contain superfluous prototypes.

Through prototype selection, the training set size is reduced, which could be useful for reducing classification runtimes, particularly for instance-based classifiers.

There are two strategies [1] for reducing the training set:

Selection. Some prototypes from *T* are retained while ruling out those that do not significantly contribute to the classification accuracy.

Replacement. The original training set is replaced by some prototypes that do not necessarily coincide with the prototypes in *T*.

J. Ruiz-Shulcloper and W.G. Kropatsch (Eds.): CIARP 2008, LNCS 5197, pp. 153–160, 2008.

In a training set, some prototypes could be more relevant than others, then for prototype selection, it could be useful to determine the relevance of each prototype and select the most relevant prototypes for each class. In this work, we propose a new prototype selection method which, according to a prototype relevance function, selects the most relevant prototypes in the training set and through them some border prototypes (prototypes located in a region where there are prototypes from different classes) are selected since these last give useful information to the classifier for preserving the class discrimination regions [2, 3].

In order to show the performance of our method, we present an experimental comparison among our method and some other prototype selection methods using the obtained prototype sets as training for the *k-NN* [4], *C4.5* [5] and *Naive Bayes* [6, 7] classifiers. In addition, we report the runtimes for each method in order to show how fast is our method with respect to the other tested methods, mainly for large datasets, where prototype selection is more useful.

This paper is structured as follows: in section 2, some works related to the prototype selection are described. Section 3 introduces our method for prototype selection and section 4 shows the experimental results. Finally, in section 5, some conclusions and directions for future work are given.

2 Related Works

Several methods have been proposed for solving the prototype selection problem, in this section, some of the most relevant methods are briefly described.

The *Condensed Nearest Neighbor* (*CNN*) [8] and the *Edited Nearest Neighbor* (*ENN*) [9] rules are two of the first prototype selection methods. The *CNN* method starts with $S = \varnothing$ and its initial step consists in randomly including in S one prototype belonging to each class. 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 all prototypes in T are correctly classified. This method ensures that S correctly classifies all prototypes in T, this is, S is consistent but does not guarantee to find a minimal consistent subset. A variant of *CNN* is the *Generalized Condensed Nearest Neighbor Rule* (*GCNN*) [10], 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 (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.

The *ENN* method consists in discarding from T those prototypes that do not belong to their k nearest neighbors' class. This method is used as noise filter because it deletes noisy prototypes, that is, prototypes with a different class in a neighborhood. A variant of this method is the Repeated *ENN* (*RENN*) where *ENN* is repeatedly applied until all prototypes in S have the same class that the majority of their k nearest neighbors. Another extension of *ENN* is the *All k-NN* prototype selection method [11]. 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 prototype flagged as bad.

Devijver and Kittler [12] proposed the *Multiedit* method for prototype selection, which creates m random partitions ($P_1 \ldots P_m$) from T. After that, *ENN* (using 1-NN) is applied over each partition P_i finding the neighbors of P_i in $P_{(i+1) \bmod m}$. This process is repeated until there are not changes (eliminations) in f successive iterations.

Wilson and Martinez [2] presented five methods *DROP1… DROP5* (*Decremental Reduction Optimization Procedure*) for prototype selection. These 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.

The *Iterative Case Filtering algorithm* (*ICF*) was proposed in [3]. *ICF* is based on the *Coverage* and *Reachable* sets which are the neighborhood set and associates set respectively. In this method, a prototype p is flagged for removal if $|Reachable(p)|>|Coverage(p)|$, which means that other prototypes can correctly classify to p (or prototypes similar to p) without p. After, all prototypes flagged for removal are deleted.

The methods described above and most of the prototype selection methods have been proposed based on the *k-NN* rule, in this work we propose a prototype selection method which is not based on this rule and, for large datasets, it is faster than the other tested methods.

3 Proposed Method

In a training set, there are some prototypes which are more similar than others in the same class; the most similar prototypes could be more representative or relevant than the less similar ones, then it makes sense for prototype selection to retain the most relevant prototypes. In this paper, the relevance of each prototype is given in terms of the average similarity that it has with the others thus the most similar to all the prototypes (in the same class) the most relevant in the class.

In this paper, we propose 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. As we mentioned before, in this work, the relevance of a prototype p is given in terms of the average similarity (A_N) which is computed as follows:

$$A_N(p) = \frac{\sum_{p' \in C, p \neq p'} S(p, p')}{|C| - 1} \tag{1}$$

Where:

C is the set of training prototypes belonging to the same class than p.

$S(p, p')$ is a similarity function for comparing prototypes. In particular, in this work we used *HVDM* (*Heterogeneous Value Difference Metric*) [13] as similarity function. This function works over numeric, non numeric and missing features.

The initial phase of *PSR* consists in 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. Notice that depending on the training set and/or the relevance function, some relevant prototypes could be border at the same time. *PSR* finds the border prototypes as follows: for each prototype among the *r* chosen, the nearest prototype belonging to each class different from *i* is selected as a border prototype.

Finally, the prototype set obtained by *PSR* contains the *r* most relevant prototypes found through the relevance criterion described above and the border prototypes found from them.

4 Experimental Results

In this section, we compare *PSR* against the *DROP3* and *DROP5* methods, which are two of the most successful prototype selection methods (according to experiments reported in [2] and [3]). The *GCNN* method is also considered in our comparison since according to results reported by its authors, this method is competitive against the *DROP* methods. We applied these methods over ten small datasets and three medium-large datasets taken from the UCI Repository [14]. For all the experiments, 10-fold cross validation was used.

For *PSR*, in the initial phase, it is necessary to choose the *r* most relevant prototypes per class in the training set, therefore some experiments using different values for *r* were done. In table 1 we show the accuracy results (using *k-NN*, *k=3*) obtained by *PSR* selecting different percentage of relevant prototypes per class, the tested values for *r* were *r= 10%, 20%, 30%,* and *40%* of the prototypes in each class. The classification accuracy is reported under the column *Acc* and *Ret* corresponds to the percentage of prototypes (retention) in *T* that were included in *S*, that is, $Ret = 100|S|/|T|$. The averages of accuracy and retention are shown at the bottom of the table.

Table 1. Classification accuracy (*Acc*) and retention (*Ret*) obtained by *PSR* selecting different number of relevant prototypes in the initial phase

Dataset	Percentage (*r*) of prototypes per class							
	r=10%		*r=20%*		*r=30%*		*r=40%*	
	Acc	*Ret*	*Acc*	*Ret*	*Acc*	*Ret*	*Acc*	*Ret*
Bridges	48.90	26.31	50.90	37.62	57.63	47.79	57.54	55.65
Echocardiogram	83.21	14.26	83.92	26.57	90.53	37.68	90.53	48.05
Glass	61.12	23.67	64.37	32.81	64.85	42.36	65.80	51.24
Heart Cleveland	75.81	14.22	75.84	25.70	79.18	36.96	78.20	49.39
Heart Swiss	92.88	11.01	92.88	21.94	93.71	30.53	93.71	41.10
Hepatitis	74.95	13.11	82.58	23.58	83.16	33.40	81.91	43.22
Iris	86.66	14.88	88.66	25.70	91.33	38.07	89.33	47.11
Letter	79.99	25.22	87.91	37.74	88.17	49.19	90.39	58.73
Liver	61.96	14.36	62.85	24.66	63.77	35.55	64.92	44.15
Segmentation	88.38	17.00	90.66	28.94	91.95	38.34	92.57	50.56
UPS	77.62	22.18	81.99	35.43	85.27	46.44	87.48	56.12
Wine	94.00	18.10	92.12	30.96	92.18	42.94	92.74	52.24
Zoo	93.33	29.50	93.33	39.75	93.33	51.11	93.33	59.13
Average	**78.37**	**18.76**	**80.62**	**30.11**	**82.70**	**40.80**	**82.96**	**50.51**

Based on the table1, in the average case, the best accuracy was obtained using *r=40%* but a very similar accuracy was obtained using *r=30%* which produced a better percentage of reduction over the datasets, therefore we used *r=30%* in the next experiments.

Once the *PSR* initial parameter has been fixed, a comparison among *PSR*, the *DROP* methods and *GCNN* was done. The results are shown in table 2, (using *k-NN*, *k*=3, the best value for the *DROP* methods [2]). In addition, we show the accuracy obtained by the original datasets (O*rig*). In figure 1, the classification accuracy (horizontal axis) versus retention (vertical axis) scatter graphic of the results shown in the table 2 is depicted.

Table 2. Classification (*Acc*) and retention (*Ret*) results obtained by: the original training set (*Orig*), *DROP3, DROP5, GCNN* and *PSR* using *k-NN*

Dataset	Orig		DROP3		DROP5		GCNN		PSR	
	Acc	*Ret*	*Acc*	*Ret*	*Acc*	*Ret*	*Acc*	*Ret*	*Acc*	*Ret*
Bridges	66.09	100	56.36	14.78	62.82	20.66	68.20	88.20	57.63	47.79
Echocardiogram	95.71	100	92.86	13.95	94.82	14.87	93.39	22.67	90.53	37.68
Glass	71.42	100	66.28	24.35	62.16	25.91	69.61	61.62	64.85	42.36
Heart Cleveland	82.49	100	78.89	11.44	79.87	14.59	67.63	9.09	79.18	36.96
Heart Swiss	93.72	100	93.72	1.81	93.72	1.81	75.76	62.45	93.71	30.53
Hepatitis	79.29	100	78.13	11.47	75.42	15.05	60.66	17.75	83.16	33.40
Iris	94.66	100	95.33	15.33	94.00	12.44	96.00	38.00	91.33	38.07
Letter	95.00	100	92.68	16.33	92.17	13.63	95.29	34.08	88.17	49.19
Liver	65.22	100	67.82	26.83	63.46	30.59	66.09	83.70	63.77	35.55
Segmentation	95.10	100	92.19	15.94	91.86	14.30	92.71	13.82	91.95	38.34
UPS	96.48	100	94.59	10.27	93.99	7.96	94.78	34.53	85.27	46.44
Wine	94.44	100	94.41	15.04	93.86	10.55	94.44	78.89	92.18	42.94
Zoo	93.33	100	90.00	20.37	95.56	18.77	95.55	26.17	93.33	51.11
Average	**86.38**	**100**	**84.10**	**15.22**	**84.13**	**15.47**	**82.32**	**43.92**	**82.70**	**40.80**

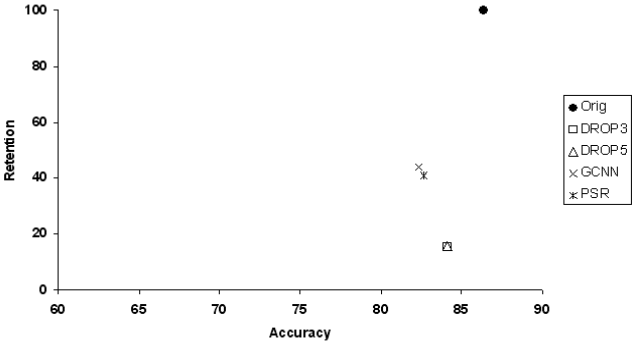

Fig. 1. Scatter graphic from the average results shown in table 2

Based on the results in table 2 and figure 1, we can observe that in the average case, the best methods were *DROP3* and *DROP5*. The classification accuracy obtained by *PSR* was smaller than those obtained by the *DROP* methods but *PSR* outperformed *GCNN*.

The best methods in table 2 were the *DROP*, however, they obtained good results since their selection criterion is related to the Nearest Neighbor rule and the same rule was used for evaluating the obtained prototype sets. For this reason, other experiments were done using the prototype sets obtained by the *DROP* methods, *GCNN* and *PSR* as training for other classifiers different from *k-NN*. In particular, we used *C4.5* (decision trees) and *Naive Bayes* (*NB*) classifiers. The *C4.5* and *NB* results are reported in tables 3-4 and figures 2-3 respectively.

Table 3. Classification results obtained using the original training set (*Orig.*) and the prototype sets obtained by *DROPs, GCNN* and *PSR* as training for the *C4.5* classifier

Dataset	Orig.	DROP3	DROP5	GCNN	PSR
Bridges	65.81	47.90	39.54	52.36	51.09
Echocardiogram	95.71	84.10	92.85	91.78	95.71
Glass	67.29	60.19	53.76	60.75	63.48
Heart Cleveland	71.96	68.59	72.16	66.00	71.35
Heart Swiss	93.71	93.71	93.71	81.59	92.05
Hepatitis	76.70	63.33	63.41	65.16	83.20
Iris	93.99	92.66	90.66	88.66	93.33
Letter	88.29	72.96	73.00	81.21	78.65
Liver	63.67	59.48	63.67	61.76	65.21
Segmentation	96.02	81.61	88.75	85.71	89.00
UPS	87.79	74.42	74.35	85.77	80.92
Wine	94.44	84.43	78.88	95.55	94.44
Zoo	93.33	81.10	88.88	81.10	95.55
Average	**83.75**	**74.19**	**74.89**	**76.72**	**81.08**

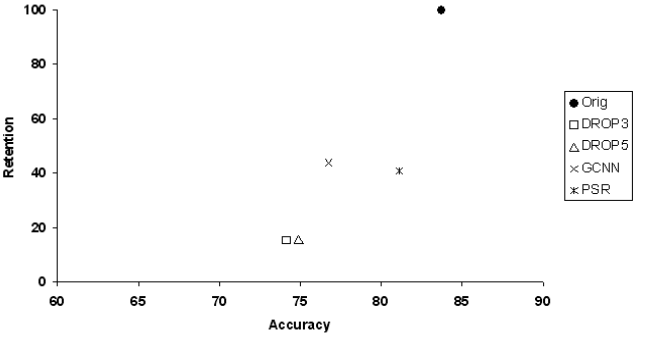

Fig. 2. Scatter graphic from the average results shown in table 3

According to the results obtained using *C4.5* and *NB*, in the average case, in accuracy, the best prototype selection method was *PSR* followed by *GCNN*. Based on these results, we can observe that using other classifiers, different from *k-NN*, the prototype subsets obtained by the *DROP3, DROP5* and *GCNN* are not as good as those obtained by *PSR*.

The dataset sizes and runtimes[1] spent by each prototype selection method tested in our experiments are shown in table 5. Based on the runtimes, we can observe that

[1] These runtimes do not depend on the used classifier and were obtained using an Intel Celeron CPU 2.4GHz, 512MB RAM.

Table 4. Classification results obtained using the original training set (*Orig.*) and the prototype sets obtained by *DROPs, GCNN* and *PSR* as training for the *NB* classifier

Dataset	Orig.	DROP3	DROP5	GCNN	PSR
Bridges	64.00	49.61	39.81	44.36	50.90
Echocardiogram	97.14	78.31	77.32	91.78	90.71
Glass	48.05	49.56	47.74	47.57	60.71
Heart Cleveland	83.81	78.20	81.16	75.52	81.86
Heart Swiss	92.05	93.71	93.71	61.02	69.23
Hepatitis	84.58	61.83	54.20	65.87	79.37
Iris	95.33	91.99	93.99	95.33	91.99
Letter	64.00	53.88	56.09	45.50	65.30
Liver	56.02	61.50	61.77	56.88	66.94
Segmentation	80.19	75.23	71.76	76.66	84.00
UPS	77.21	71.58	70.51	72.27	74.77
Wine	98.81	61.11	66.66	96.66	92.22
Zoo	95.55	88.88	83.33	93.33	95.55
Average	**79.75**	**70.41**	**69.08**	**70.98**	**77.20**

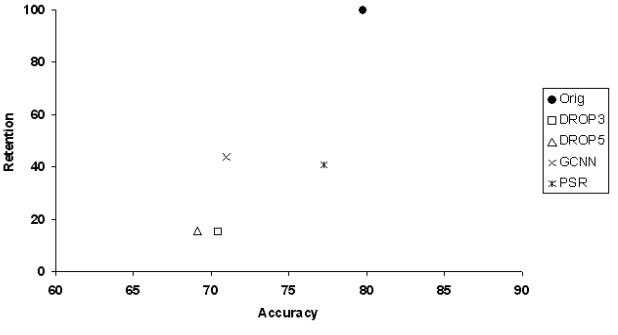

Fig. 3. Scatter graphic from the average results shown in table 4

Table 5. Datasets sizes and runtimes (in seconds) spent by the tested methods

Dataset	Size			Runtimes			
	Prototypes	Features	Classes	DROP3	DROP5	GCNN	PSR
Echocardiogram	74	9	2	0.07	0.08	1.02	0.42
Zoo	90	16	7	0.26	0.27	1.15	0.75
Bridges	108	11	7	0.16	0.14	15.27	1.15
Heart Swiss	123	12	2	0.22	0.36	16.63	1.47
Iris	150	4	3	0.19	0.16	1.25	0.37
Hepatitis	155	19	2	0.46	0.50	5.33	2.42
Wine	178	13	3	0.84	0.59	3.03	0.46
Glass	214	9	6	0.46	0.47	4.14	0.87
Heart Cleveland	303	13	5	1.32	1.34	12.62	6.57
Liver	345	6	2	0.76	0.76	19.45	0.73
Segmentation	2100	19	7	208.39	208.91	716.28	59.49
Letter	20000	16	26	15213.13	14339.61	64120.25	912.28
UPS	9000	255	10	37854.95	64788.06	211200.26	3125.42

for small datasets, the fastest methods were *DROP3* and *DROP5*, however, for large datasets, *PSR* were much faster than all the other methods; therefore, *PSR* is particularly useful for large datasets where the other prototype selection methods require very long time, and in some cases, these methods could be inapplicable.

5 Conclusions

In this work, we have proposed, compared and tested the *PSR* method for prototype selection. *PSR* selects the most relevant prototypes per class in a training set and through them some border prototypes are selected in order to preserve discrimination capability between classes.

The experimental results showed that *PSR* is a good method for solving the prototype selection problem mainly when a classifier different from *k-NN* is used. In our experiments, *PSR* obtained the best results using *C4.5* and *NB* classifiers. When *k-NN* was used, in accuracy, *PSR*, *DROP3, DROP5* and *GCNN* were similar.

When prototype selection is required for small datasets, other methods outperform *PSR* in runtime but an important characteristic of our method is that it is faster for large datasets (datasets where prototype selection is particularly useful). This characteristic is very important because other successful prototype selection methods spent a lot of time (or could be inapplicable) for processing these kind of datasets.

In this work, we computed the prototypes relevance based on the average similarity, but as future work we are interested in proposing another way for computing the prototypes relevance.

References

1. Bezdek, J.C., Kuncheva, L.I.: Nearest prototype classifier designs: An experimental study. International Journal of Intelligent Systems 16(12), 1445–1473 (2001)
2. Wilson, D.R., Martínez, T.R.: Reduction Techniques for Instance-Based Learning Algorithms. Machine Learning 38, 257–286 (2000)
3. Brighton, H., Mellish, C.: Advances in Instance Selection for Instance-Based Learning Algorithms. Data Mining and Knowledge Discovery 6, 153–172 (2002)
4. Cover, T., Hart, P.: Nearest Neighbor Pattern Classification. IEEE Transactions on Information Theory 13, 21–27 (1967)
5. Quinlan, J.R.: C4.5 Programs for Machine Learning. Morgan Kaufmann, San Francisco (1993)
6. Mitchell, T.M.: Machine Learning. WCB McGraw-Hill, Boston (1997)
7. Duda, R.O., Hart, P.E., Stork, D.G.: Pattern Classification, 2nd edn. John Wiley & Sons, New York (2001)
8. Hart, P.E.: The Condensed Nearest Neighbor Rule. IEEE Transactions on Information Theory 14(3), 515–516 (1968)
9. Wilson, D.L.: Asymptotic Properties of Nearest Neighbor Rules Using Edited Data. IEEE Transactions on Systems, Man, and Cybernetics 2(3), 408–421 (1972)
10. Chien-Hsing, C., Bo-Han, K., Fu, C.: The Generalized Condensed Nearest Neighbor Rule as A Data Reduction Method. In: 18th International Conference on Pattern Recognition, vol. 2, pp. 556–559. IEEE press, Washington (2006)
11. Tomek, I.: An Experiment with the Edited Nearest-Neighbor Rule. IEEE Transactions on Systems, Man, and Cybernetics 6-6, 448–452 (1976)
12. Devijver, P.A., Kittler, J.: On the edited nearest neighbor rule. In: 5th International Conference on Pattern Recognition. The Institute of Electrical and Electronics Engineers, pp. 72–80 (1980)
13. Wilson, D.R., Martínez, T.R.: Improved Heterogeneous Distance Functions. Journal of Artificial Intelligence Research 6-1, 1–34 (1997)
14. UCI Machine Learning Repository, University of California, School of Information and Computer Science, Irvine CA, `http://www.ics.uci.edu/~mlearn/MLRepository.html`