Performance Evaluation of Prototype Selection Algorithms for Nearest Neighbor Classification

J.S. SÁNCHEZ[1]
R. BARANDELA[2]
R. ALEJO[2]
A.I. MARQUÉS[1]
[1]Universitat Jaume I, Av. Vicente Sos Baynat s/n, 12006 Castellón, Spain
sanchez@uji.es
[2]Instituto Tecnológico de Toluca, Av. Tecnológico s/n, 52140 Metepec, Mexico
rbarandela@hotmail.com

Abstract. Prototype selection is primarily effective in improving the classification performance of Nearest Neighbor (NN) classifier and also partially in reducing its storage and computational requirements. This paper reviews some prototype selection algorithms for NN classification and experimentally evaluates their performance using a number of real data sets. Finally, new approaches based on combining the NN and the Nearest Centroid Neighbor (NCN) of a sample [3] are also introduced.

1 Introduction

Much of the research work in the frame of supervised pattern recognition has been almost entirely devoted to the analysis and characteristics of classification algorithms and to the study of feature selection methods. Recently, however, an increasing emphasis is being given to the evaluation of procedures used to collect and clean the training sample, a critical aspect for effective automation of discrimination tasks.

The NN rule [4] is a well-known supervised non-parametric classifier that combines its conceptual simplicity and an asymptotic error rate conveniently bounded in terms of the optimal Bayes error. In its classical manifestation, given a set of n previously labeled prototypes or training set (TS), this classifier assigns any given sample to the class indicated by the label of the closest prototype in the TS. More generally, the k-NN rule maps any sample to the pattern class most frequently represented among its k closest neighbors.

Nevertheless, the NN classifiers also suffer from certain drawbacks. The performance of these rules, as with any non-parametric method, is extremely sensitive to incorrectness or imperfections in the TS. On the other hand, its applicability to real-time problems, with a large set of prototypes of high dimensionality, can become prohibitive because of the immense computational loads required for searching the nearest neighbors of each sample in the TS.

Extensive efforts have been devoted to the improvement of the NN classification rules in various aspects [4]. First, condensed NN rules [5, 12], fast NN search algorithms [9, 10] and adaptive learning rules [14, 16] try to alleviate the high computational cost of the NN rules. Second, alternative distance metrics [19, 22], adaptive decision rules [13], sample re-combination [11] and editing [1, 8, 20, 25] contribute to improving the classification performance of the NN rules.

Joint use of edited and condensed NN rules, generally referred to as prototype selection [7], has been proposed to pick out an appropriate subset of prototypes with computational efficiency and accuracy as ultimate goals [6]. While condensing aims at selecting a sufficiently small subset of prototypes that leads approximately the same performance than the NN rule using the whole TS, editing eliminates outliers from the original TS and "cleans" possible overlapping among regions of different classes.

Outlier has traditionally been defined as a case that does not follow the same model as the rest of the data [24]. Now the term outlier is being employed to cover a broad range of circumstances reflecting some confusion among different situations and a lack of rigorous and unified concept of outlier data. In general, there are three of these potential situations: noisy or atypical data that can be produced by errors (measuring, recording, etc) [26], new unidentified patterns appearing in the classification phase and that do not belong to any of the classes represented in the TS [18], and also mislabeled prototypes in the TS [2].

The present work compares the performance of existing prototype selection algorithms by running a set of experiments over several real databases. In particular, this

1530-1834/01 $10.00 © 2001 IEEE

paper focuses on improving the classification accuracy of the NN rule by means of editing approaches.

The rest of this paper is organized as follows. The prototype selection algorithms tested in the experimental study are outlined in Section 2. By combining different neighborhood concepts, new editing approaches are derived in Section 3. Databases and experiments are described in Section 4. Next, Section 5 provides the main experimental results. Finally, some concluding remarks and future extensions are drawn in Section 6.

2 Algorithms

As already mentioned in Section 1, this work is concerned with editing algorithms. All these methods are primarily aimed at improving classification accuracy of the NN rule by preprocessing the training prototypes. Nevertheless, they also obtain, as a byproduct, a decrease in the TS size and accordingly, a reduction in the computational burden of the classification rule.

In this section, the editing approaches tested in the empirical study are briefly described. In particular, these schemes correspond to the original Wilson's technique [25], the unlimited repetition of Wilson's editing proposed by Tomek [23], the all k-NN editing method [23], depuration [1], proximity graph-based editing [20], and the k-NCN editing approach [8].

Wilson's editing

This corresponds to the first proposal to edit the NN rule. In a few words, it consists of applying the k-NN classifier to estimate the class label of all prototypes in the TS and discard those samples whose class label does not agree with the class associated with the largest number of the k neighbors.

Thus, the Wilson's editing procedure can be written as follows:

- Let $S = X$. (X is the original TS, and S will be the resulting or edited TS)
- For each x_i in X do:
  a) $T = X - \{x_i\}$.
  b) Find the k-NN of x_i in T.
  c) Discard x_i from S if there is a majority of NNs from a different class.

Tomek's editing

This proposal consists of editing the already edited TS and so on, while prototypes liable to be edited remain in the TS.

According to the author (and also to a large number of experimental evaluations), the procedure is always stopped after a finite number of iterations, because after a certain number of repetitions, the TS becomes immune to further applications.

All k-NN editing

This corresponds to an application of Wilson's editing using all the l-NN rules, with l ranging from 1 through k, for a predetermined value of k.

- Let $S = X$.
- For each x_i in X do:
  a) $T = X - \{x_i\}$
  b) Set *flag* = 1 and $l = 1$
  c) While *flag* = 1 and $l < k+1$ do:
     1. Find the l-NN of x_i in T.
     2. Discard x_i from S if there is a majority of NNs from a different class, and set *flag* = 0.
     3. Set $l = l + 1$

Depuration of TS

This is a methodology that can be regarded as a cleaning process removing some suspicious instances of the TS and correcting the class labels of some others while retaining them and accordingly, it is designed to cope with all types of imperfections of the training instances (mislabeled, noisy and atypical or exceptional cases).

The method involves the application, only several times, of the *Generalized* editing [15] and afterwards, the employment of Wilson's technique, perhaps also reiterated. In Generalized editing, two parameters must be defined: k and k', in such a way that $(k + 1) / 2 \leq k' \leq k$.

For each prototype x_i in the TS, its k NNs are searched in the remainder of the TS. If a particular class has at least k' representatives among those k neighbors, then x_i is labeled according to that class, independently of its original label. Otherwise, x_i is edited (i.e., removed) from the TS. In short, the procedure looks for modifications of the training sample structure through changes of the labels of some training patterns and removal of some others.

Proximity graph-based editing

In [20] the Gabriel Graph (GG) and the Relative Neighborhood Graph (RNG) are used for editing the NN classification rule. These are two well-known examples of proximity graphs that establish a geometrical relation between a sample and some of its neighbors. In brief, the simplest editing scheme can be defined as follows:

- Compute the proximity graph (GG or RNG) corresponding to the given TS.
- Discard those prototypes that are misclassified by their graph neighbors (that is, pairs of samples that define the set of edges in the resulting proximity graph).

***k*-NCN editing**

This is a way of Wilson's algorithm particularized for the case of using the *k*-NCN classification rule [21] to estimate the class label of prototypes.

It is worth mentioning that the *k*-NCN classifier is thought to obtain a more accurate information about prototypes and more specially, for those close to decision boundaries. In general, this results in a practical improvement of the corresponding editing preocedure. The *k*-NCN editing algorithm consists of the following steps:

- Let $S = X$.
- For each x_i in X do:
  a) $T = X - \{x_i\}$.
  b) Find the *k*-NCN of x_i in T.
  c) Discard x_i from S if there is a majority of NCNs from a different class.

3 A new editing approach

A heuristic alternative to editing the NN rule is here proposed. This scheme tries to jointly use information about proximity as well as about the spatial distribution of prototypes around a given sample. The aim of this approach is to obtain a balanced trade-off between the classical nearest neighborhood and the surrounding neighborhood represented by the NCNs.

After computing the k NCNs of a given sample in the TS, only those whose NN among all prototypes in the TS has their same class label assigned are considered for editing. The procedure can be summarized as follows:

- Let $S = X$.
- For each x_i in X do:
  a) $T = X - \{x_i\}$.
  b) Find the *k*-NCN of x_i in T.
  c) Select the $j \leq k$ NCNs correctly labeled according to the NN rule.
  d) Discard x_i from S if there is a majority of those j NCNs belonging to a different class.

Alternatively, one might use the general *k*-NN decision rule (instead of the particular 1-NN) for editing the set of the NCNs (Step 2.c of the algorithm). Nevertheless, it is worth noting that, in general, the use of the *k*-NN scheme in the editing procedures does not lead to a significantly better behaviour than the simple NN classifier.

4 Experiments

Experiments over nine standard benchmark data sets taken from the UCI Repository [17] have been carried out to compare the performance of the editing algorithms previously introduced. A short description of the corresponding databases is given in Table 1.

Data set	No. classes	No. features	TS size	Test set size
Glass	6	9	174	40
Iris	3	4	120	30
Liver	2	6	276	69
Pima	2	6	615	153
Vehicle	4	18	678	168
Vowel	11	10	429	99
Cancer	2	9	546	137
Heart	2	13	216	54
Wine	3	13	144	34

Table 1 Description of experimental data sets.

Since all these data sets are small, five-fold cross-validation has been applied to obtain the performance results. Each initial database has randomly been divided into training and test samples, as shown in Table 1. Thus the experiment consists of applying the NN rule to each of the test sets, where the training portion has been preprocessed by using the different editing schemes.

5 Results

Two main aspects are of interest in this section. From each trial, the reduction in the TS size and the classification accuracy are calculated. The percentage reduction of training samples gives a direct measure of the amount of computational savings due to the number of prototypes resulting from each editing technique. On the other hand,

the recognition accuracy provides a check on the ability of the algorithms to select the most "efficient" prototypes (for example, high classification accuracy may result from retaining more prototypes instead of selecting fewer efficient samples).

In order to balance these two competing goals, a normalized Euclidean distance between each (reduction, recognition) pair and the origin (0% reduction, 0% recognition) has been calculated. Using this measure, the "best" technique is deemed as the one that produces the largest distance. Another way of visualizing this is to plot the recognition accuracy versus the reduction percentage and look for the point that is closest to the (100%, 100%) corner.

The k-NCN editing has been tried with increasing values of the neighborhood k (ranging from 3 through 9), and the ones leading to the best performance have finally been included in the tables. On the other hand, the editing approach proposed here, hereafter namely Edited k-NCN, has been run with only two different values of the parameter k (that is, 3 and 5). Finally, the Wilson's editing technique and its iterative algorithm (Tomek's editing) along with the all k-NN scheme have been used with $k = 3$ in all experiments.

Table 2 reports the average classification accuracy obtained by the NN rule applied over the resulting edited sets. The recognition rate for each entire original TS (i.e., no editing) has also been included for comparison purposes. Values in brackets correspond to the standard deviation. Highligh indicates the best method for each experimental database. Analogously, Table 3 represents the resulting TS size after applying each editing procedure.

Results in Table 2 show that in general, the best alternatives for these particular data sets correspond to the unlimited repetition of Wilson's technique, k-NCN editing, depuration and also RNG-based editing. On the other hand, it seems that the GG-based edited sets suffer from an important degradation in performance almost without exception under all the cases, but specially in the Vowel database. With respect to the recognition accuracy achieved by the editing algorithm here proposed, it should be noted that it is very similar to that of the best option in each database.

Examining the other critical factor of interest (see Table 3), namely edited set size, the results show that, as is to be expected, the Tomek's editing generally obtains the highest reduction rate: the averaged percentage of prototypes discarded from the initial TS is about 45%. On the other hand, the results also indicate that the other methods are very similar in terms of resulting edited set size: in particular, all of them provide about 15% - 25% less reduction than the iterative version of Wilson's technique.

	Glass	Iris	Liver	Pima	Vehicle	Vowel	Cancer	Heart	Wine
Original TS	70.0	96.7	65.2	63.9	64.3	97.6	95.6	58.2	72.3
	(5.30)	(1.52)	(4.82)	(5.70)	(1.79)	(1.71)	(2.49)	(6.23)	(3.37)
Best k-NCN	**68.0**	96.0	70.1	74.1	62.7	89.7	96.0	**69.6**	71.8
	(5.79)	(1.34)	(6.71)	(2.64)	(2.16)	(4.21)	(1.90)	(3.63)	(3.99)
Wilson ($k = 3$)	63.0	**96.7**	69.3	72.0	59.6	86.67	96.0	64.4	71.8
	(6.20)	(2.11)	(6.24)	(2.59)	(3.16)	(6.46)	(1.90)	(1.39)	(8.02)
Repetition ($k = 3$)	64.5	**96.7**	**71.4**	73.2	60.0	**95.0**	96.5	66.3	**72.9**
	(7.58)	(2.37)	(8.50)	(3.67)	(4.20)	(11.55)	(2.27)	(3.04)	(7.61)
Edited k-NCN ($k = 3$)	67.0	**96.7**	68.1	72.2	61.4	89.7	96.3	67.8	70.0
	(5.34)	(2.11)	(4.85)	(3.47)	(3.24)	(4.21)	(2.13)	(2.22)	(6.81)
Edited k-NCN ($k = 5$)	65.0	**96.7**	66.4	73.9	62.5	81.0	96.2	67.4	70.0
	(7.07)	(2.11)	(6.94)	(2.45)	(1.00)	(7.70)	(2.45)	(1.89)	(9.00)
GG	67.0	95.3	69.3	74.1	59.6	61.4	95.9	67.8	70.0
	(5.79)	(1.64)	(4.71)	(3.27)	(5.76)	(3.91)	(2.00)	(5.57)	(5.39)
RNG	67.5	90.2	68.1	72.0	**63.2**	92.9	96.3	65.9	68.8
	(6.52)	(1.34)	(4.85)	(2.35)	(2.79)	(3.38)	(1.67)	(3.44)	(6.86)
All k-NN ($k = 3$)	64.2	**96.7**	68.1	71.7	59.6	86.7	96.3	65.2	67.7
	(6.29)	(2.36)	(7.39)	(3.84)	(2.36)	(7.22)	(2.28)	(2.41)	(5.5)
Depuration	67.0	**96.7**	70.3	**75.9**	61.3	83.2	**96.7**	69.3	70.6
($k = 3, k' = 2$)	(5.12)	(1.52)	(7.15)	(2.58)	(5.54)	(7.91)	(2.34)	(5.32)	(11.76)

Table 2 Classification rates of the NN rule applied to different edited sets.

	Glass	Iris	Liver	Pima	Vehicle	Vowel	Cancer	Heart	Wine
Original TS	174	120	276	615	678	429	546	216	144
Best k-NCN	115.8	115.6	190.8	456.0	422.6	412.8	529.6	147.2	105.6
	(5.49)	(0.80)	(4.45)	(10.14)	(7.96)	(3.54)	(2.73)	(3.87)	(2.06)
Wilson (k = 3)	114.8	115.4	176.0	427.8	423.2	407.0	529.6	138.2	100.2
	(6.43)	(1.36)	(7.16)	(6.43)	(8.08)	(4.34)	(2.06)	(0.75)	(3.66)
Repetition (k = 3)	**76.8**	**33.0**	146.8	**338.2**	**340.2**	378.0	**321.8**	131.6	**86.2**
	(43.44)	(47.54)	(54.38)	(47.74)	(46.26)	(23.33)	(25.98)	(6.58)	(15.36)
Edited k-NCN (k = 3)	114.6	115.6	176.8	424.0	431.8	412.2	527.4	135.2	103.8
	(5.75)	(0.80)	(8.06)	(3.74)	(5.74)	(3.19)	(4.03)	(3.19)	(2.93)
Edited k-NCN (k = 5)	109.8	116.2	185.6	438.6	437.0	376.6	528.4	139.0	104.2
	(6.68)	(0.75)	(4.45)	(3.07)	(7.07)	(7.00)	(3.72)	(2.19)	(1.94)
GG	105.6	115.2	190.4	472.8	342.8	**119.4**	530.8	147.8	110.8
	(4.32)	(1.17)	(3.93)	(8.42)	(11.14)	(6.53)	(1.60)	(3.54)	(2.23)
RNG	132.2	115.0	195.2	474.6	469.0	397.4	532.0	152.4	118.0
	(6.18)	(0.63)	(6.73)	(10.07)	(7.64)	(3.26)	(2.76)	(4.18)	(2.61)
All k-NN (k = 3)	111.0	114.6	**145.4**	363.2	377.2	406.4	519.8	**115.0**	92.8
	(6.24)	(0.89)	(12.95)	(10.13)	(7.01)	(5.03)	(5.40)	(3.32)	(4.15)
Depuration	142.4	35.0	232.0	403.6	601.4	426.0	338.2	183.8	109.8
(k = 3, k' = 2)	(26.34)	(47.53)	(48.39)	(49.55)	(18.11)	(2.45)	(36.96)	(12.00)	(16.66)

Table 3 Size of edited sets.

Figures 1-a through 1-f visually illustrate the percentage reduction versus classification accuracy for Glass, Vehicle, Vowel, Wine, Pima and Liver databases, respectively. It is apparent that, in most cases, differences among the performance of the distinct editing algorithms are not significant: in fact, all of them achieve a high enough recognition accuracy, but also retain a considerable amount of prototypes. This is consistent with the fact that editing is aimed at improving the NN classification accuracy, while the TS size reduction can be understood only as a consequence of the methodology applied.

Perhaps the most surprising result from the experiments refers to Figure 1-c. As can be seen, for the Vowel database, GG editing eliminates a very large number of prototypes (72.23% reduction). Nevertheless, as it can be expected, it also suffers from an important drop in recognition accuracy (61.42%). On the other hand, for this particular database, all the other editing algorithms provide a very low reduction rate (5.65%) and a high enough classification accuracy (89.13%).

6 Conclusions and future extensions

When using a NN classifier, it is necessary to work with a sufficiently reduced number of prototypes due to practical reasons. Nevertheless, it is convenient a large TS in order to approach optimal recognition accuracy. On the other hand, the presence of mislabeled prototypes can strongly degrade the classification accuracy.

Thus, a certain trade-off between accuracy and TS size may constitute a good solution for both problems. Taking into account these two important issues, in the present paper, some experiments have been carried out over a number of real data sets using several standard prototype selection techniques.

Differences in recognition accuracy among the editing algorithms here tested are marginal. In general, on the other hand, almost all the schemes retain a very similar number of prototypes in most cases. Therefore, it is difficult to conclude the most efficient editing technique under these particular databases.

Future work includes investigation of the potential of editing and condensing tools for achieving even better performance (that is, reduction rate and recognition accuracy). On the other hand, another interesting issue to be further studied refers to the effect of applying different metrics to various prototype selection methods. This should help to draw more definitive conclusions with regard to the benefits of using distinct editing and condensing techniques.

Acknowledgements

This work has partially been supported by grant TIC2000-1703-C03-03 from the *Spanish CICYT* and by grants 32016-A and 744.99P from the *Mexican CONACyT* and *COSNET*, respectively.

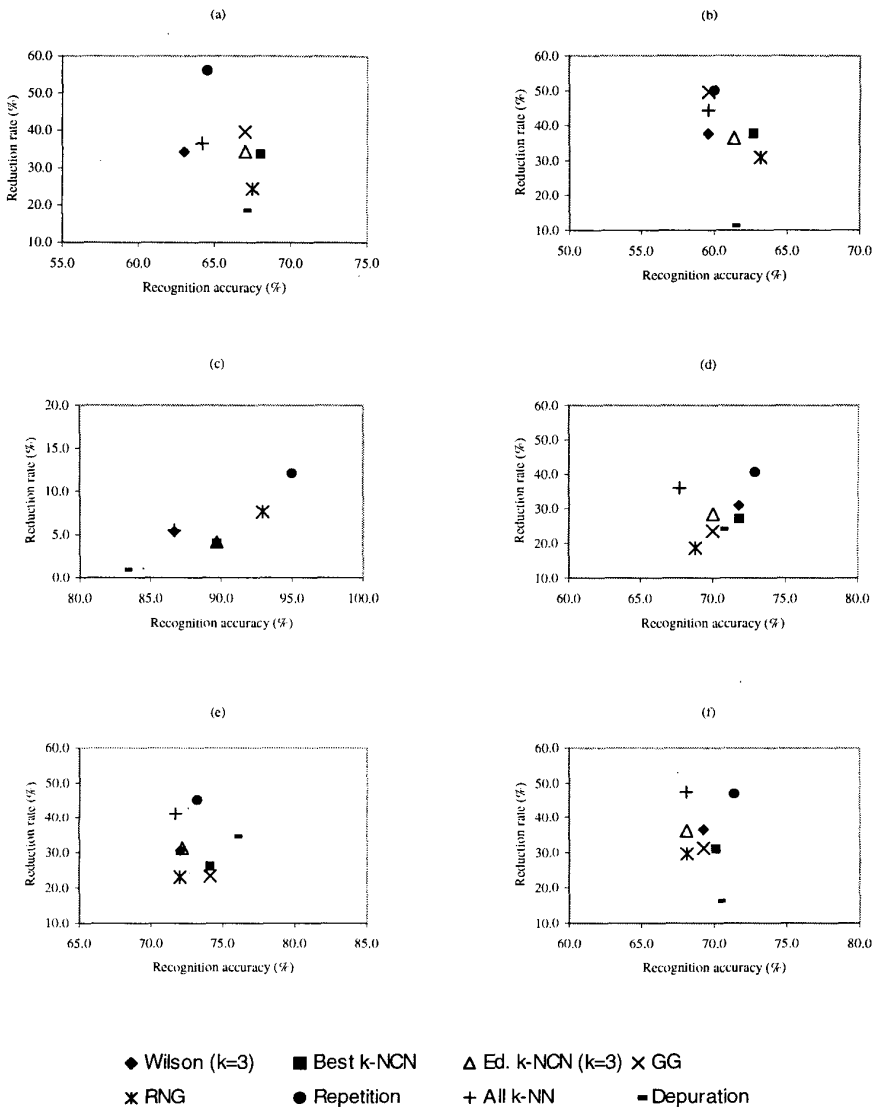

Figure 1 Recognition versus reduction for Glass, Vehicle, Vowel, Wine, Pima and Liver databases, respectively.

References

[1] R. Barandela, and E. Gasca, "Decontamination of training samples for supervised pattern recognition methods", In: *Advances in Pattern Recognition*, Lecture Notes in Computer Science 1876, Springer Verlag (2000), 621-630.

[2] C.E. Bordley and M.A. Friedl, "Identifying mislabeled training data", *Journal of Artificial Intelligence Research 11 (1999)*, 131-167.

[3] B.B. Chaudhuri, "A new definition of neighborhood of a point in multi-dimensional space", Pattern Recognition Letters 17 (1996), 11-17.

[4] B.V. Dasarathy, *Nearest Neighbor Norms: NN Pattern Classification techniques*, IEEE Computer Society Press, Los Alamos, CA, 1991.

[5] B.V. Dasarathy, "Minimal consistent subset (MCS) identification for optimal nearest neighbor decision systems design", *IEEE Trans. on Systems, Man, and Cybernetics* 24 (1994), 511-517.

[6] B.V. Dasarathy, J.S. Sánchez and S. Townsend, "Nearest neighbor editing and condensing tools - synergy exploitation", *Pattern Analysis & Applications* 3 (2000), 19-30.

[7] P.A. Devijver and J. Kittler, *Pattern Recognition: A Statistical Approach*, Prentice Hall, Englewood Cliffs, NJ, 1982.

[8] F.J. Ferri, J.S. Sánchez and F. Pla, "Editing prototypes in the finite sample size case using alternative neighbourhoods", In: *Advances in Pattern Recognition*, Lecture Notes in Computer Science 1451, Springer-Verlag (1998), 620-629.

[9] K. Fukunaga and P.M. Narendra, "A branch and bound algorithm for computing *k*-nearest neighbors", *IEEE Tans. on Computers* 24 (1975), 750-753.

[10] P.J. Grother, G.T. Candela and J.L. Blue, "Fast implementation of nearest neighbor classifiers", *Pattern Recognition* 30 (1997), 459-465.

[11] Y. Hamamoto, S. Uchimura and S. Tomita, "A bootstrap technique to nearest neighbor classifier design", *IEEE Trans. on Pattern Analysis and Machine Intelligence* 19 (1997), 73-79.

[12] P.E. Hart, "The condensed nearest neighbor rule", *IEEE Trans. on Information Theory* 14 (1968), 515-516.

[13] T. Hastie and R. Tobshirai, "Discriminant adaptive nearest neighbor classification", *IEEE Trans. on Pattern Analysis and Machine Intelligence* 18 (1996), 607-615.

[14] T. Kohonen, "The self-organizing map", *Proc. of IEEE* 78 (1990), 1464-1480.

[15] J. Koplowitz and T.A. Brown, "On the relation of performance to editing in nearest neighbor rules", In: Proc. of 4th Intl. Joint Conf. on Pattern Recognition, Japan (1978).

[16] C.L. Liu and M. Nakagawa, "Evaluation of prototype learning algorithms for nearest-neighbor classifier in application to handwritten character recognition", *Pattern Recognition* 34 (2001), 601-615.

[17] C.J. Merz and P.M. Murphy, *UCI Repository of Machine Learning Databases*, Dept. of Information and Computer Science, University of California, Irvine, CA, 1998.

[18] R. Muzzolini, Y.H. Yang and R. Pierson, "Classifier design with incomplete knowledge", *Pattern Recognition* 31 (1998), 345-369.

[19] F. Ricci and P. Avesani, "Data compression and local metrics for nearest neighbor classification", *IEEE Trans. on Pattern Analysis and Machine Intelligence* 21 (1999), 380-384.

[20] J.S. Sánchez, F. Pla and F.J. Ferri, "Prototype selection for the nearest neighbour rule through proximity graphs", *Pattern Recognition Letters* 18 (1997), 507-513.

[21] J.S. Sánchez, F. Pla and F.J. Ferri, "On the use of neighbourhood-based non-parametric classifiers", *Pattern Recognition Letters* 18 (1997), 1179-1186.

[22] R.D. Short and K. Fukunaga, "The optimal distance measure for nearest neighbor classification", *IEEE Trans. on Information Theory* 27 (1981), 622-627.

[23] I. Tomek, "An experiment with the edited nearest neighbor rule", *IEEE Trans. on Systems, Man and Cybernetics* 6 (1976), 448-452.

[24] S. Weinsberg, *Applied Linear Regression*, John Wiley and Sons, 1985.

[25] D.L. Wilson, "Asymptotic properties of nearest neighbor rules using edited data sets", *IEEE Trans. on Systems, Man and Cybernetics* 2 (1972), 408-421.

[26] D.R. Wilson and T.R. Martinez, "Reduction techniques for instance-based learning algorithms", *Machine Learning* 38 (2000), 257-286.