On the use of different classification rules in an editing task

L. Micó[1], F. Moreno-Seco[1], J.S. Sánchez[2], J.M. Sotoca[2], and R.A. Mollineda[2]

[1] Dept. Llenguatges i Sistemes Informàtics, Universitat d'Alacant
E-03071 Alacant (Spain)
[2] Dept. Llenguatges i Sistemes Informàtics, Universitat Jaume I
Av. Sos Baynat s/n, E-12071 Castelló de la Plana (Spain)

Abstract. Editing allows the selection of a representative subset of prototypes among the training sample to improve the performance of a classification task. The Wilson's editing algorithm was the first proposal and then a great variety of new editing techniques have been proposed based on it. This algorithm consists on the elimination of prototypes in the training set that are misclassified using the k-NN rule. From such editing scheme, a general editing procedure can be straightforward derived, where any classifier beyond k-NN can be used. In this paper, we analyze the behavior of this general editing procedure combined with 3 different neighborhood-based classification rules, including k-NN. The results reveal better performances of the 2 other techniques with respect to k-NN in most of cases.

Keywords: Pattern recognition, classification, nearest neighbor, prototype selection, editing.

1 Introduction

The k-Nearest Neighbor (k-NN) rule is a well known non-parametric classification approach. This rule classifies an unknown sample into the class most represented among its k nearest neighbors according to some metric [5]. Although it is mainly used for classification, the k-NN rule is widely used also for *edition*.

Given a set $\mathcal{T}$ of prototypes, an editing technique consists on the selection of a subset, $\mathcal{S} \subseteq \mathcal{T}$ where the overlapping among different classes has been reduced. The removed prototypes can be either those which belongs to overlapping regions, those erroneously labeled or atypical prototypes (*outliers*). The use of this technique improves the performance of the 1-NN classifier.

The Wilson's editing algorithm [2] was the first proposal related with the elimination of misleading prototypes from the training set $\mathcal{T}$. This technique retains in $\mathcal{T}$ only the correctly classified samples by a *leaving one out* strategy with a k-NN classifier. However, a more general editing scheme can be derived from Wilson's, by considering the error estimation strategy and the classification rule as editing scheme parameters.

In this work, an exhaustive evaluation of such general editing scheme based on three different neighborhood-based classification rules has been done. The

main purpose is to compare the performances of these three classifiers in an editing task over a wide variety of known datasets. The three neighborhood-based rules are the well-known k-NN rule [3, 7], the k-NCN rule [4] and the new k-NSN rule [9].

The k-NN rule classifies a sample into the majority class among its k nearest neighbors in $\mathcal{T}$. The k-NCN rule classifies a sample in the class most represented among the k neighbors whose centroid is the closest to the sample. These k neighbors are not usually the k nearest neighbors. The results achieved by the k-NCN rule are very interesting, outperforming the k-NN rule in many cases, specially with small training sets (which is what usually happens in practice). Finally, the k-NSN rule considers the k best neighbors selected by fast NN search algorithms when looking for the NN.

The structure of the paper is as follows. Section 2 presents the general editing scheme. In section 3 we shall briefly describe the distance-based rules that have been considered, and some details of their uses for edition. Section 4 consists of exhaustive experiments with 12 datasets and a discussion of results. Finally, we will conclude and outline some future work in section 5.

2 A General Editing Scheme

The classification accuracy of the NN rule can be improve by eliminating outliers and cleaning possible overlapping among classes in the original training set. This is the main goal of any editing technique.

A general editing procedure can be straightforward derived from Wilson's scheme. Given a training set $\mathcal{T}$, an error estimation strategy ξ, and a classification rule δ, let $\mathcal{R} \subseteq \mathcal{T}$ be the subset of samples incorrectly classified by δ using ξ. The edited subset $\mathcal{S}$ is obtained by removing from $\mathcal{T}$ those samples in $\mathcal{R}$. This process can be repeated until a certain condition η is satisfied. Figure 1 illustrates an schematic description of such procedure. Once the training set has been edited, the 1-NN rule is used to classify new samples.

In the experiments, this editing procedure is combined with 3 neighborhood-based classification rules, that is, rules which take into account the distances to a number of close neighbors and their classes to decide the class of a new sample. These classifiers are the plain k-NN rule [3], and two other related decision rules named the k-NCN rule [4] and the new k-NSN rule [9], respectively. Next section describes in details these techniques.

3 Neighborhood-Based Classification Rules

3.1 The k-NN rule

One of the most widely studied non-parametric classification approaches corresponds to the k-NN rule. Given a set of n previously labeled prototypes or training set (TS), the k-NN classifier [7] consists of assigning an input sample to the class most frequently represented among the k closest prototypes in the

Input: a training set of labeled samples $\mathcal{T} = \{(x_i, c_i)\}$
a classification rule δ
an error estimation strategy ξ
an ending condition η
Output: a final edited subset $\mathcal{S}$
Method:
repeat
let $\mathcal{R} \subseteq \mathcal{T}$ be the subset of samples incorrectly classified by ξ, δ
$\mathcal{S} \longleftarrow \mathcal{T} - \mathcal{R}$
$\mathcal{T} \longleftarrow \mathcal{S}$
until η is true

Fig. 1. A general editing scheme.

TS, according to a certain dissimilarity measure. A particular case of this rule is when $k = 1$, in which each input sample is assigned to the class indicated by its closest neighbor.
The asymptotic classification error of the k-NN rule (that is, when n grows to infinity) tends to the optimal Bayes error rate as $k \to \infty$ and $k/n \to 0$. Moreover, if $k = 1$, the error is bounded by approximately twice the Bayes error [8]. This behavior in asymptotic classification performance combines with a conceptual and implementational simplicity, which makes it a powerful classification technique capable of dealing with arbitrarily complex problems, provided there is a large enough TS available.
However, in many practical settings, this theoretical behavior can hardly be achieved because of certain inherent weaknesses that significantly reduce the applicability of k-NN classifiers in real-world tasks. For example, the performance of these rules, as with any non-parametric approach, is extremely sensitive to incorrectness or imperfections in the TS.
That is the reason why a considerable amount of works have been devoted to improve the NN classification accuracy by eliminating outliers from the original TS and also cleaning possible overlapping among classes. This strategy has generally been referred as to *editing* [8], whereas the corresponding classifier has been called *edited NN rule*.

3.2 The *k*-NCN rule

The nearest centroid neighborhood [6] refers to a concept in which neighborhood is defined taking into account not only the proximity of prototypes to a given input sample but also their *symmetrical distribution* around it. From this general

idea, the corresponding classification rule, the k-nearest centroid neighbors (k-NCN) [4], has been proven to overcome the traditional k-NN classifier in many practical situations.
Now the editing approach presented here corresponds to a slight modification of the original work of Wilson and basically consists of using the *leaving one out* error estimate with the k-NCN classification rule.

3.3 The k-NSN rule

Recently, a new distance-based classification rule, the k *nearest selected neighbor* rule (k-NSN) has been proposed. The k-NSN rule is based on a class of fast NN search algorithms, those who search iteratively for the nearest neighbor: in each step, these algorithms select a candidate to nearest neighbor, then compute its distance to the sample, update the current nearest neighbor, prune the training set, and look for a new candidate. This process is repeated until no new candidates may be found. The k-NSN rule classifies the sample using the k nearest candidates selected while looking for the nearest neighbor, the so called k nearest selected neighbors. Of course, the performance of the k-NSN rule depends highly on the underlying fast NN search algorithms, and usually the fastest algorithm is the one with which the k-NSN results are the poorest, and vice versa. When the training set is large and/or the dimensionality of the data is high, the k-NSN rule obtains results that are similar to those of the k-NN rule (in fact, the k-NSN rule uses in these cases almost all of the k-NN for classification), but without the extra computational effort of finding exactly the k-NN.

Wilson's editing with k-NSN The fast NN search algorithms in which is based the k-NSN rule all require a certain data structure (usually a tree) to be built during the training phase, prior to classification. When using a *leaving one out* scheme for error estimation or for Wilson editing, these algorithms may need to rebuild its data structures many times, and this is usually a very time consuming step. In this work the k-NSN rule has been used only with one fast NN algorithm, the LAESA [10] algorithm, which is one of the simplest algorithms with which the k-NSN rule has been tested.

The LAESA algorithm uses a reduced matrix of distances between a subset of *base prototypes* and the rest of the prototypes in the training set. As the number of base prototypes required depends on the dimensionality of the data and not on the size of the training set, the spatial complexity is lineal on that size. The base prototypes are selected in the training phase as those that are maximally separated, and then the reduced matrix is computed.

In a *leaving one out* procedure, the algorithm should recompute the base prototypes and the reduced matrix each time the training set changes (i.e. each time a prototype is left out). However, in many cases the base prototypes would be the same as for the whole training set, so the matrix would be (almost) the same. Only in a few cases the result would differ. The set of base prototypes affects the number of distances computed, and thus the number of selected neighbors,

so the performance of the k-NSN rule may be slightly different, but not too much. The simplest way to avoid recomputing each time the base prototypes set and the reduced matrix is to compute the set and the matrix for the whole training set, and then, if the prototype left out is one of the base prototypes, simply ignore the row corresponding to that prototype in the matrix for further computations.[3]

4 Experiments and Discussions

Experiments involved 12 datasets from the UCI Machine Learning Repository (http://www.ics.uci.edu/∼mlearn). Table 1 summarizes the main characteristics of each data set: number of classes, attributes, and prototypes.

Data set	No. Classes	No. Features	Size
Cancer	2	9	685
Clouds	2	2	5002
Concentric	2	2	2501
Diabetes	2	8	770
Gauss	2	2	5002
German	2	24	1002
Glass	6	9	216
Heart	2	13	272
Liver	2	6	347
Phoneme	2	5	5406
Sonar	2	60	210
Waveform21	3	21	5001

Table 1. A brief summary of the UCI databases.

To guarantee the statistical significance of results, all classification tasks were designed following a 5-fold cross validation. The 5 training partitions derived from each dataset were edited with the 3 editing techniques resulting from the combination of the general procedure of section 2 with the 3 distance-based classification rules (k-NN, k-NCN, k-NSN) described in section 3. In the case of k-NSN, the LAESA algorithm [10, 9] was used as the fast NN search algorithm. Only the parameter $k = 3$ was used for all editing.

Then, resulting edited partitions were used to classify with the 1-NN rule their corresponding test partitions for each dataset. An additional baseline 1-NN clasification task was performed with the original training partitions (without any edition) and their corresponding test partitions (called **nedit**).

[3] The matrix is used to compute a lower bound of the distance of each prototype to the sample, so the candidate to nearest neighbor is selected as that whose lower bound is the lowest.

For each pair of dataset and editing technique, the average size of edited partitions and the average 1-NN classification accuracy on test partitions were collected. Table 2 provides a summary of these results.

In all datasets, edited partitions improve the 1-NN classification results of the corresponding original partitions (**NOEDIT**). Note that in all those cases the number of prototypes of edited partitions is lower than the size of original training partitions. These relations between classification accuracies and sizes are really common, but do not necessarily occur for all datasets. Their presence denote that there is some overlapping that can be removed by edition. Therefore, these datasets can better illustrate the behavior of editing techniques.

With respect to the comparison among the three different editions, it can be observed that they produce similar results both in the number of prototypes removed and in 1-NN classification accuracies. But, in most of cases, the editing scheme derived from k-NCN leads to better accuracies than the other 2 rules and, specifically, than the k-NN. These differences are more notable in the datasets with small number of prototypes (Glass, Heart, Liver). The importance of this observation is that small size datasets are very frequent in real world problems and they are usually a challenge for researchers. In addition, the use of the k-NSN rule also produces good edited partitions, with the lowest number of prototypes in many cases and with a similar accuracy in most of situations. These results confirm the applicability of this new rule for editing tasks.

5 Conclusions, Discussions, and Future Work

An editing process consists basically of removing from a training set those samples which may disorient a classifier training (samples in overlapping regions and outliers). This paper focuses on the comparison of 3 editing methods, which are particular instances of a general editing procedure directly derived from Wilson's scheme. Given that this general procedure allows a classifier as a parameter, each specific editing method is defined by a neighborhood-based classification rule. The 3 classifiers considered are the plain k-NN [3, 7] (the original classifier of the Wilson's scheme), the k-NCN [4] and a more recent k-NSN [9]. The k-NCN searches those k neighbors whose centroid is closest to a given sample, while k-NSN uses a fast NN search algorithm to find the k reference neighbors.

Exhaustive experiments were conducted over 12 datasets to compare the performances of these rules when used in editing tasks. A 5-fold cross validation strategy was defined for classifier evaluation. Editing methods were applied on training partitions and resulting edited partitions were used for 1-NN classification of their corresponding test partitions. Although average results were similar among editing methods, the k-NCN was better than k-NN in most of cases, considering the 1-NN classification accuracy. The differences were more significant in datasets with a small number of samples, which is a very frequent situation. Finally, and in spite of its approximated strategy, the k-NSN produces good results both in sizes of edited subsets and in classification accuracies on test partitions.

The main conclusion of this paper is the appropriateness of k-NCN in classification tasks on small size problems with respect to k-NN. An interesting question arises from this fact. How related is this conclusion with samples density? This feature is probably the most important condition in the behavior of k-NN techniques, but depends not only on the number of samples but also on the volume where samples are distributed. So, small size datasets are not necessarily those with low density. Future analysis should involve some measure to evaluate density, and a methodology for relating density, size, and dimensionality with the use of each neighborhood-based classification rule.

Acknowledgments

This work has been supported in part by grant TIC2003–08496 from the Spanish CICYT (Ministerio de Ciencia y Tecnología), GV06/166 from Generalitat Valenciana, and the IST Programme of the European Community, undes the Pascal Network of Excellence, IST-2002-506778.

References

1. Bernardo, E., Ho, T.-K.: On classifier domain of competence, In: Proc. 17th. Int. Conf. on Pattern Recognition, Cambridge, UK (2004) 136–139.
2. Wilson, D.L.: Asymptotic properties of nearest neighbour rules using edited data, IEEE Trans. on Systems, Man and Cybernetics **2** (1972) 408–421.
3. Cover, T.M., Hart, P.E.: Nearest Neighbor Pattern Classification, IEEE Trans. on Information Theory **IT-13(1)** (1967) 21–27.
4. Sánchez, J.S. *et. al.*: Analysis of new techniques to obtain quality training sets, Pattern Recognition Letters **24** (2003) 1015–1022.
5. Duda, R., Hart, P.: Pattern Classification and Scene Analysis. Wiley (1973).
6. Chaudhuri, B.B.: A new definition of neighborhood of a point in multi-dimensional space, Pattern Recognition Letters **17** (1996) 11-17.
7. Dasarathy, B.V.: Nearest Neighbor Norms: NN Pattern Classification techniques. IEEE Computer Society Press (1991), Los Alamos, CA.
8. Devijver, P.A., Kittler, J.: Pattern Recognition: A Statistical Approach. Prentice Hall (1982), Englewood Cliffs, NJ.
9. Moreno-Seco, F., Micó, L., Oncina, J.: Extending fast nearest neighbour search algorithms for approximate k-NN classification, In: Lecture Notes in Artificial Intelligence **2652** (2003) 589–597.
10. Micó, L., Oncina, J., Vidal, E.: A new version of the nearest neighbour approximating and eliminating search algorithm (AESA) with linear preprocessing-time and memory requirements, Pattern Recognition Letters **15** (1994) 9–17.

	Cancer		**Clouds**	
scheme	*edit*	*accuracy*	*edit*	*accuracy*
NOEDIT	547	95.17(2.38)	4000	84.66(0.96)
k-NSN	530(2.94)	96.19(2.08)	3391(18.43)	87.66(0.66)
k-NN	528(3.83)	**96.34**(1.90)	3498(12.09)	**88.26**(0.55)
k-NCN	528(2.64)	95.61(2.44)	3504(13.00)	**88.26**(0.44)
	Concentric		**Diabetes**	
scheme	*edit*	*accuracy*	*edit*	*accuracy*
NOEDIT	1999	81.59(1.26)	614	67.32(4.15)
k-NSN	1978(4.71)	81.59(1.26)	428(3.31)	70.83(3.62)
k-NN	1976(2.79)	81.59(1.26)	425(4.17)	71.75(2.22)
k-NCN	1984(3.49)	81.59(1.26)	436(8.95)	**72.01**(2.12)
	Gauss		**German**	
scheme	*edit*	*accuracy*	*edit*	*accuracy*
NOEDIT	4000	64.94(0.90)	800	65.61(2.22)
k-NSN	2592(24.58)	**68.32**(0.90)	544(8.47)	68.41(1.79)
k-NN	2688(17.98)	64.72(0.48)	543(10.09)	69.30(1.02)
k-NCN	2708(16.13)	64.72(0.48)	563(7.47)	**70.61**(1.69)
	Glass		**Heart**	
scheme	*edit*	*accuracy*	*edit*	*accuracy*
NOEDIT	171	65.21(14.95)	216	58.17(5.31)
k-NSN	110(7.86)	60.61(12.59)	131(2.53)	65.91(1.25)
k-NN	109(7.24)	59.66(10.08)	139(1.33)	63.68(1.27)
k-NCN	111(8.26)	**67.11**(11.41)	139(2.58)	**66.25**(3.78)
	Liver		**Phoneme**	
scheme	*edit*	*accuracy*	*edit*	*accuracy*
NOEDIT	216	65.21(7.36)	4323	69.72(7.28)
k-NSN	115(5.38)	63.21(6.32)	3936(44.91)	72.24(6.44)
k-NN	116(7.73)	66.95(6.68)	3898(51.08)	**72.83**(6.29)
k-NCN	120(4.17)	**70.54**(6.26)	3937(52.62)	72.33(6.25)
	Sonar		**Waveform21**	
scheme	*edit*	*accuracy*	*edit*	*accuracy*
NOEDIT	166	52.11(10.75)	3999	77.96(2.58)
k-NSN	136(4.83)	56.49(12.73)	3249(19.81)	80.70(2.05)
k-NN	137(4.82)	**56.97**(13.03)	3250(19.63)	80.70(2.05)
k-NCN	140(4.49)	55.55(13.58)	3245(22.52)	**80.74**(2.00)

Table 2. Average size of the edited sets and average 1-NN classification accuracies on test partitions (standard deviations are in brackets). Values in bold type indicate the highest accuracy for each database.