Using Maximum Similarity Graphs to Edit Nearest Neighbor Classifiers

Milton García-Borroto[1,3], Yenny Villuendas-Rey[2], Jesús Ariel Carrasco-Ochoa[3], and José Fco. Martínez-Trinidad[3]

[1] Bioplantas Center, UNICA, Carretera a Morón km 9 ½, C. de Ávila, Cuba
mil@bioplantas.cu
http://www.bioplantas.cu
[2] Ciego de Ávila University UNICA, Carretera a Morón km 9 ½, C. de Ávila, Cuba
yennyv@bioplantas.cu
http://www.unica.cu
[3] National Institute of Astrophysics, Optics and Electronics, Puebla, México
{ariel,fmartine}@inaoep.mx
http://www.inaoep.mx

Abstract. The Nearest Neighbor classifier is a simple but powerful non-parametric technique for supervised classification. However, it is very sensitive to noise and outliers, which could decrease the classifier accuracy. To overcome this problem, we propose two new editing methods based on maximum similarity graphs. Numerical experiments in several databases show the high quality performance of our methods according to classifier accuracy.

Keywords: nearest neighbor, error-based editing, prototype selection.

1 Introduction

One of the most popular non-parametric classifiers is the Nearest Neighbor (NN). This classifier combines the simplicity with a classification error bounded by twice the Bayes error [1]. For classifying a new object, the NN classifier compares it against all the objects in the training set, and assigns the new object to the class of its nearest object.

An important drawback of the NN classifier is its sensitivity to noisy and mislabeled objects [2]. Since NN introduction in 1967, there is a constant research interest in the Pattern Recognition community to overcome this drawback [3-5]. Editing algorithms aim at improving classifier accuracy by deleting noisy or mislabeled objects.

In this paper, we address the problem of improving NN accuracy by smoothing classification boundaries. We use maximum similarity graphs to determine border objects, and delete those noisy or mislabeled objects that most likely could affect the classifier accuracy.

This paper is organized as follows: section two describes some previous works about NN error-based editing and their drawbacks, section three introduces our proposals, section four shows the results of the experiments and section five offers some conclusions.

E. Bayro-Corrochano and J.-O. Eklundh (Eds.): CIARP 2009, LNCS 5856, pp. 489–496, 2009.

2 Previous Works on Error-Based Editing

Several authors divide the algorithms to improve a training set for NN classifiers in two main categories: condensing algorithms and error-based editing algorithms (or just editing algorithms) [6]. Condensing algorithms aim at reducing the NN computational cost by obtaining a small subset of the training set, maintaining the accuracy as high as possible, while editing algorithms aim at improving classifier accuracy by deleting noisy or mislabeled objects. In this work, we focus on editing algorithms.

The first editing algorithm was the Edited Nearest Neighbor (ENN), proposed by Wilson in 1972 [3]. The ENN algorithm deletes all objects misclassified by a *k*-NN classifier, where k is a user-defined parameter, usually $k = 3$. The results of ENN strongly depend on the value of k, and there is not a simple procedure to find this value *a priori.* Another weak point is the use of a unique value of k for the entire database; without taking into account the object densities in different regions. Finally, a major drawback of ENN comes from the behavior you can see in Fig. 1. Note that the border object A will be removed using even a low value of $k = 3$, no matter it is very similar to other objects of its own class.

Fig. 1. ENN can erroneously remove objects in the class boundaries

In 1976, Tomek introduced the All-KNN algorithm[4]. All-KNN deletes an object if a k-NN classifier misclassifies it, with k in the range $1 \leq k \leq kMax$ where $kMax$ is a user-defined parameter, usually $kMax = 7$ or $kMax = 9$. The use of several values for k in All-KNN makes the algorithm to do more deletions than ENN. Nevertheless, in many cases, like that showed in Fig. 1, it produces an undesired behavior. All-KNN keeps the same drawbacks than ENN.

Another classical editing method is MULTIEDIT, proposed by Devijver and Kittler in 1980 [5]. First, MULTIEDIT randomly divides the training set in ns partitions. On each partition, it applies the ENN method using a 1-NN classifier trained with the next partition. After each iteration, MULTIEDIT joins the remaining objects in each partition and it repeats the process until no change is achieved in ni successive iterations. Both ns and ni are user-defined parameters. Usually $ns = 3$ and $ni = 2$. This method can successfully purge noisy objects and outliers, but if two classes are very close, it can completely remove one or both of them (Fig. 2). Also, the strong random characteristic of MULTIEDIT could make the result of two consecutive executions over the same training set to be completely different, as you can also see in Fig. 2.

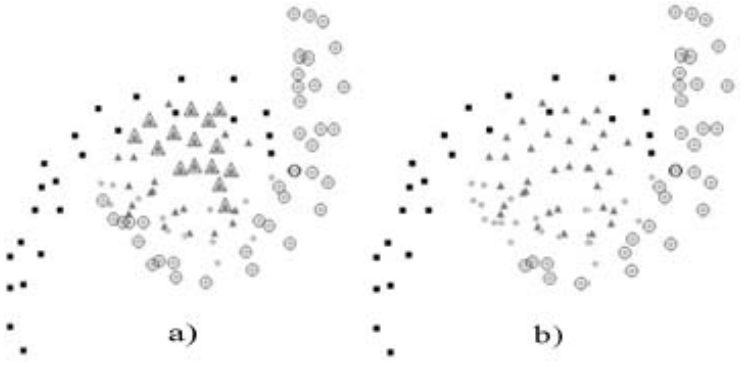

Fig. 2. Example of two MULTIEDIT executions: (a) one class deletion and (b) two class deletion. ■ ● ▲ are the objects in the training matrix; □ ○ △ highlight the selected objects.

Hattori and Takahashi in 2000 [7] proposed a new editing method, referred by us as NENN. The method computes the k nearest neighbors for each object. If an object has at least one of its neighbors in another class, NENN deletes this object from the training set. This condition is stronger than the one used in ENN, and could produce more dramatic object deletion. Unfortunately, if the boundaries between classes are close together, it can delete many important boundary objects, like the whole " ° " class in Fig. 1.

In 2002, Toussaint used the Relative Neighborhood proximity graphs to edit nearest neighbors [8]. The RNG-E algorithm computes the Relative Neighborhood graph of the training set, and deletes all objects misclassified by its neighbors in the graph. The Relative Neighborhood graph is a proximity graph with the set of edges defined as $ges = \{(p,q): \Lambda_{p,q} \cap T = \emptyset\}$, where p and q are vertexes, T is the training set, and $\Lambda_{p,q}$ is the intersection between the hyper-spheres centered in p and q respectively, with radius $\|p - q\|$.

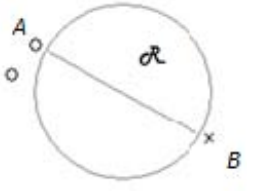

Fig. 3. A common problem of RNG-E algorithm

A common problem with RNG-E is that the proximity graph can connect faraway objects, because RNG-E depends on the configuration of other objects in the graph. For example, in Fig. 3 you can see that objects *A* and *B* are neighbors because no other object exists in the region $\mathcal{R}$ even though they are distant.

Caballero *et al.*, introduced the editing methods EditRS1 and EditRS2 in 2007 [9]. They used elements of the Rough Set theory to obtain lower and upper approximations of the training set for computing the limit regions of each class. The EditRS1

algorithm computes the lower approximation of each class and deletes the objects not included in the lower approximation. The EditRS2 also computes the lower approximation of the classes, and the limit region or border of each class. For each limit region, EditRS2 applies the Generalized Editing method [10]. Finally, the method deletes such objects not included in the lower approximations or in the edited limit region. Although both methods are supported by a well-founded theory, in most of the tested databases they were unable to remove any object.

The analysis of previous editing methods reveals that improving the k-NN accuracy by editing the training set is still an open problem. Previous methods cannot accurately deal with some cases, which is the basic motivation of the methods introduced in this paper.

3 Editing Based on Maximum Similarity Graphs

In this paper, we introduce MSEditA and MSEditB, two new editing methods based on maximum similarity graphs (MSG). A maximum similarity graph [11] is a directed graph, where each object is connected to its most similar neighbor. Formally, let $G = (T, \theta)$ be a maximum similarity graph for a training set T, with arcs θ. Two objects $x, y \in T$ form an arc $(x, y) \in \theta$ if

$$\max_{o \in T} \{sim(x, o)\} = sim(x, y)$$

where $sim(x, y)$ is a similarity function, usually defined in terms of the normalized Euclidean distance as $1 - d(x, y)$, but in general it can be any similarity or dissimilarity. This way we have an arc between an object and its most similar neighbor. If we have ties, we insert arcs for all the most similar neighbors.

A MSG is very useful for object selection because it can catch the similarity relations between any object and those that are on its neighborhood. It is also immune to configurations like that appearing in Fig. 1, which makes most of other editing methods to fail. Although in theory MSG can connect faraway objects, it is not a common behavior, because both objects have to be in complete isolation. In the example in Fig. 3, for example, the MSG does not connect the objects A and B.

3.1 Proposed Methods

Both MSEditA and MSEditB methods first compute the maximum similarity graph of the training set T, and then decide which objects to delete. In MSEditA, an object in the graph having a most similar neighbor from different class indicates a level of uncertainty of the correct class for the object. The idea followed in MSEditA consists in deleting an object if it has any most similar neighbor (successors) with different class. This method is different from ENN (using $k = 1$), because if the most similar object is not unique, ENN randomly selects one of them. In this case, MSEditA discard the object if any of the neighbors belongs to a different class.

On the other hand, MSEditB removes an object if the majority of its neighbors belong to a different class. We count as neighbors both successors and predecessors of an object in the graph, which are respectively its most similar objects and the objects for which the evaluated object is the most similar.

Pseudocode of MSEditA

1. $Edited \leftarrow T$
2. Compute maximum similarity graph of T.
3. For each object in $Edited$
4. If the object has at least one successor of different class in the maximum similarity class, delete the object.
5. Return $Edited$.

Pseudocode of MSEditB

1. $Edited \leftarrow T$
2. Compute maximum similarity graph of T
3. For each object o in $Edited$
4. Let be S_o and A_o the successors and antecessors of o in the graph.
5. $N \leftarrow S_o \cup A_o$
6. If the majority of the objects in N are not of the same class of o, delete o.
7. Return $Edited$

We can see the differences between MSEditA and MSEditB in Fig. 4.

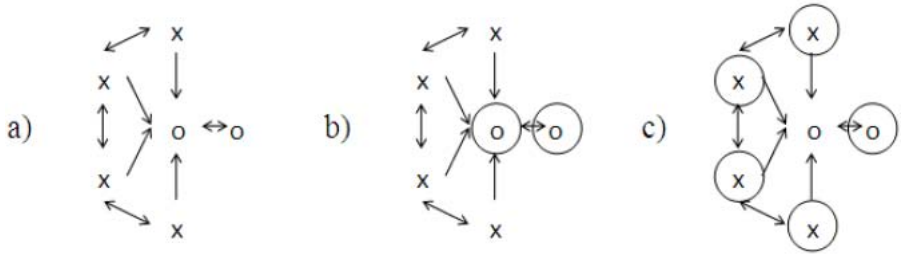

Fig. 4. Diferences (circled) between MSEditA (b) and MSEditB (c) on example MSG (a)

There are six objects from two classes: circle and cross. Each arrow represents an arc, so a single arrow from *A* to *B* means the successor *B* is the most similar object to the ancestor *A*, and a double arrow means they are simultaneously the most similar object of each other.

MSEditA deletes all crosses, because they have one of their successors in a different class, while MSEditB only deletes the central object, because most of its neighbors (successors and ancestors) are in a different class. It is important to notice that the results of these methods are different, and it is easy to prove that, in general, the result of a method is not contained in the other.

4 Experimental Results

In order to compare the performance of the proposed methods, we carried out some numerical experiments in a wide-range of databases from the UCI repository of Machine Learning [12]. The description of the tested databases appears in Table 1. We perform 10-fold cross validation among all databases, averaging the results. Due to NN classifier is dependant on the function used for comparing objects, in our experiments we use two functions: HEOM and HVDM [13].

We select several classic, state of the art and recent methods, which are frequently used for comparisons in most papers about the topic. For determining the best method, we made a two-tailed T-Test [14] (with significance of 0.05) of every method with respect to the lowest error result. If no significant difference exists, the result is also considered as lowest error. We compute how many databases each method attains the lowest error. The object retention ratio is calculated as the ratio between the amount of objects in the edited sample and the amount of objects in the training sample. Table 2 summarizes the results with both HEOM and HVDM functions.

Table 1. Databases description

Database	non-num/ num feats	Objects	Database	non-num/ num feats	Objects
anneal	29/9	798	hepatitis	13/6	155
autos	10/16	205	iris	0/4	150
balance-scale	0/4	625	labor	6/8	57
breast-cancer	9/0	286	lymph	15/3	148
breast-w	0/9	299	post-pat-data	7/1	90
cmc	7/2	1473	sonar	0/60	208
colic	15/7	368	tae	2/3	151
credit-a	9/6	690	trains	29/4	10
cylinder-bands	20/20	512	vehicle	0/18	846
dermatology	1/33	366	vote	16/0	435
glass	0/10	214	vowel	3/9	990
heart-c	7/6	303	zoo	16/1	101
heart-h	7/6	294			

In *tae* and *vowel* databases, using both HEOM and HVDM functions, all methods produce a significant degradation of the accuracy, which implies that all objects are very important to maintain classification accuracy. The same situation occurred in *cylinder-bands* database for the HEOM function. Both MSEditA and MSEditB have the best result, having the lowest error, in 22 of 25 databases. In general, usually one of the methods (not always the same) gets better results than the other gets, but we need further research to allow selecting *a priori* the best method for a database. It is important to highlight that some methods tie in most databases, so there is no categorical winner.

Table 2. Number of databses (from 25 databases) each method had lower error (err) and object retention ratio (ret)

	ENN	**All-KNN**	**MUL-TIEDIT**	**NENN**	**RNG-E**	**EditRS1**	**EditRS2**	**MSEditA**	**MSEditB**	**Original**
err HEOM	21	19	15	18	21	21	21	**22**	21	**22**
err HVDM	22	20	14	17	22	22	22	22	**23**	22
Average err	21.5	19.5	14.5	17.5	21.5	21.5	21.5	**22**	**22**	**22**
ret HEOM	0.88	0.74	**0.55**	0.57	0.76	1.00	1.00	0.74	0.75	
ret HVDM	0.88	0.74	**0.57**	0.60	0.77	1.00	1.00	0.77	0.75	

According to the object retention ratio (see Table 2), we show the averaged result for both HEOM and HVDM functions. In *cmc* database, for both functions, the NENN method deleted all objects, so we did not average this result. The best method according to retention rate was MULTIEDIT, but it was the worst according to classification accuracy. Both EditRS1 and EditRS2 only reduce objects in *breast-cancer* and *tae* databases, in all other databases they did not delete any object of the training set.

5 Conclusions

In this paper, we introduce two new editing methods based on maximum similarity graphs. Both methods have the best results according to classifier accuracy in the tested databases, by deleting noisy and mislabeled objects. They attain the higher accuracy in 22 of 25 databases. MSEditA and MSEditB have object retention rate around 75%, comparable with other editing methods such as RNE-G and All-KNN. Although NN classifier depends on the function used for comparing objects, we did not found significant difference in the performance of the methods with both HEOM and HVDM functions.

As future work, we are going to study which are the characteristics of the database that makes one of our methods to overcome the other. This way, we will be able to create a combined method with even better behavior.

Acknowledgments

The first author wants to thank to the National Institute of Astrophysics, Optics and Electronics for its support on this research. This work is partly supported by National Council of Science and Technology of México under grant 252752.

References

1. Cover, T., Hart, P.E.: Nearest Neighbor pattern classification. IEEE Trans. on Information Theory 13, 21–27 (1967)
2. Dasarathy, B.D.: Nearest Neighbor (NN) Norms: NN Pattern Classification Techniques. IEEE Computer Society Press, Los Alamitos (1991)
3. Wilson, D.L.: Asymptotic properties of nearest neighbor rules using edited data. IEEE Transactions on Systems, Man and Cybernetics 2, 408–421 (1972)
4. Tomek, I.: An experiment with the Edited Nearest-Neighbor Rule. IEEE Transactions on Systems, Man and Cybernetics SMC-6, 448–452 (1976)
5. Devijver, P.A., Kittler, J.: On the edited neares neighbor rule. In: Press, I.C.S. (ed.) 5th International Conference on Pattern Recognition, Los Alamitos, California, pp. 72–80 (1980)
6. Kuncheva, L.I.: Combining pattern classifiers: methods and algorithms. Wiley-Interscience, Hoboken (2004)
7. Hattori, K., Takahashi, M.: A new edited k-nearest neighbor rule in the pattern classification problem. Pattern Recognition 33, 521–528 (2000)
8. Toussaint, G.: Proximity Graphs for Nearest Neighbor Decision Rules: Recent Progress. In: 34 Symposium on Computing and Statistics INTERFACE-2002, Montreal, Canada, pp. 1–20 (2002)
9. Caballero, Y., Bello, R., Salgado, Y., García, M.M.: A method to edit training set based on rough sets. International Journal of Computational Intelligence Research 3, 219–229 (2007)
10. Koplowitz, J.: On the relation of performance to editing in nearest neighbor rules. Pattern Recognit. 13, 251–255 (1981)
11. Pons-Porrata, A., Berlanga-Llavori, R., Ruiz-Shulcloper, J.: Topic discovery based on text mining techniques. Information Processing & Management 43, 752–768 (2007)
12. Merz, C.J., Murphy, P.M.: UCI Repository of Machine Learning Databases. University of California at Irvine, Department of Information and Computer Science, Irvine (1998)
13. Wilson, R.D., Martinez, T.R.: Improved Heterogeneous Distance Functions. Journal of Artificial Intelligence Research 6, 1–34 (1997)
14. Dietterich, T.G.: Approximate Statistical Tests for Comparing Supervised Classification Learning Algorithms, vol. 10, pp. 1895–1923. MIT Press, Cambridge (1998)